

David Doermann  
Karl Tombre  
*Editors*

# Handbook of Document Image Processing and Recognition



Springer Reference

---

# Handbook of Document Image Processing and Recognition



---

David Doermann • Karl Tombre  
Editors

# Handbook of Document Image Processing and Recognition

With 339 Figures and 98 Tables



Springer Reference

*Editors*

David Doermann  
University of Maryland  
College Park, MD, USA

Karl Tombre  
Université de Lorraine  
Nancy, France

ISBN 978-0-85729-858-4      ISBN 978-0-85729-859-1 (eBook)  
ISBN 978-0-85729-860-7 (print and electronic bundle)  
DOI 10.1007/978-0-85729-859-1  
Springer London Heidelberg New York Dordrecht

Library of Congress Control Number: 2014932686

© Springer-Verlag London 2014, corrected publication 2019

This work is subject to copyright. All rights are reserved by the Publisher, whether the whole or part of the material is concerned, specifically the rights of translation, reprinting, reuse of illustrations, recitation, broadcasting, reproduction on microfilms or in any other physical way, and transmission or information storage and retrieval, electronic adaptation, computer software, or by similar or dissimilar methodology now known or hereafter developed. Exempted from this legal reservation are brief excerpts in connection with reviews or scholarly analysis or material supplied specifically for the purpose of being entered and executed on a computer system, for exclusive use by the purchaser of the work. Duplication of this publication or parts thereof is permitted only under the provisions of the Copyright Law of the Publisher's location, in its current version, and permission for use must always be obtained from Springer. Permissions for use may be obtained through RightsLink at the Copyright Clearance Center. Violations are liable to prosecution under the respective Copyright Law.

The use of general descriptive names, registered names, trademarks, service marks, etc. in this publication does not imply, even in the absence of a specific statement, that such names are exempt from the relevant protective laws and regulations and therefore free for general use.

While the advice and information in this book are believed to be true and accurate at the date of publication, neither the authors nor the editors nor the publisher can accept any legal responsibility for any errors or omissions that may be made. The publisher makes no warranty, express or implied, with respect to the material contained herein.

Printed on acid-free paper

Springer is part of Springer Science+Business Media ([www.springer.com](http://www.springer.com))

---

## Foreword

In the beginning, there was only OCR. After some false starts, OCR became a competitive commercial enterprise in the 1950's. A decade later there were more than 50 manufacturers in the US alone. With the advent of microprocessors and inexpensive optical scanners, the price of OCR dropped from tens and hundreds of thousands of dollars to that of a bottle of wine. Software displaced the racks of electronics. By 1985 anybody could program and test their ideas on a PC, and then write a paper about it (and perhaps even patent it).

We know, however, very little about current commercial methods or in-house experimental results. Competitive industries have scarce motivation to publish (and their patents may only be part of their legal arsenal). The dearth of industrial authors in our publications is painfully obvious. Herbert Schantz's book, *The History of OCR*, was an exception: he traced the growth of REI, which was one of the major success stories of the 1960's and 1970's. He also told the story, widely mirrored in sundry wikis and treatises on OCR, of the previous fifty years' attempts to mechanize reading. Among other manufacturers of the period, IBM may have stood alone in publishing detailed (though often delayed) information about its products.

Of the 4000-8000 articles published since 1900 on character recognition (my estimate), at most a few hundred really bear on OCR (construed as *machinery - now software - that converts visible language to a searchable digital format*). The rest treat character recognition as a prototypical classification problem. It is, of course, researchers' universal familiarity with at least some script that turned character recognition into the pre-eminent vehicle for demonstrating and illustrating new ideas in pattern recognition. Even though some of us cannot tell an azalea from a begonia, a sharp sign from a clef, a loop from a tented arch, an erythrocyte from a leukocyte, or an alluvium from an anticline, all of us know how to read.

Until about 30 years ago, OCR meant recognizing mono-spaced OCR fonts and typewritten scripts one character at a time – eventually at the rate of several thousand characters per second. Word recognition followed for reading difficult-to-segment typeset matter. The value of language models more elaborate than letter n-gram frequencies and lexicons without word frequencies gradually became clear. Because more than half of the world population is polyglot, OCR too became multilingual (as Henry Baird predicted that it must). This triggered a movement to post all the cultural relics of the past on the Web. Much of the material awaiting conversion,

ancient and modern, stretches the limits of human readability. Like humans, OCR must take full advantage of syntax, style, context, and semantics.

Although many academic researchers are aware that OCR is much more than classification, they have yet to develop a viable, broad-range, end-to-end OCR system (but they may be getting close). A complete OCR system, with language and script recognition, colored print capability, column and line layout analysis, accurate character/word, numeric, symbol and punctuation recognition, language models, document-wide consistency, tuneability and adaptability, graphics subsystems, effectively embedded interactive error correction, and multiple output formats, is far more than the sum of its parts. Furthermore, specialized systems - for postal address reading, check reading, litigation, and bureaucratic forms processing - also require high throughput and different error-reject trade-offs. Real OCR simply isn't an appropriate PhD dissertation project.

I never know whether to call hand print recognition and handwriting recognition "OCR." but abhor *intelligent* as a qualifier for the latest wrinkle. No matter: they are here to stay until tracing glyphs with a stylus goes the way of the quill. Both human and machine legibility of manuscripts depend significantly on the motivation of the writer: a hand-printed income tax return requesting a refund is likely to be more legible than one reporting an underpayment. Immediate feedback, the main advantage of on-line recognition, is a powerful form of motivation. Humans still learn better than machines.

Document Image Analysis (DIA) is a superset of OCR, but many of its other popular subfields require OCR. Almost all line drawings contain text. An E-sized telephone company drawing, for instance, has about 3000 words and numbers (including revision notices). Music scores contain numerals and instructions like *pianissimo*. A map without place names and elevations would have limited use. Mathematical expressions abound in digits and alphabetic fragments like *log*, *limit*, *tan* or *argmin*. Good lettering used to be a prime job qualification for the draftsmen who drew the legacy drawings that we are now converting to CAD. Unfortunately, commercial OCR systems, tuned to paragraph-length segments of text, do poorly on the alphanumeric fragments typical of such applications. When Open Source OCR matures, it will provide a fine opportunity for customization to specialized applications that have not yet attracted heavy-weight developers. In the meantime, the conversion of documents containing a mix of text and line art has given rise to distinct sub-disciplines with their own conference sessions and workshops that target graphics techniques like vectorization and complex symbol configurations.

Another subfield of DIA investigates what to do with automatically or manually transcribed books, technical journals, magazines and newspapers. Although Information Retrieval (IR) is not generally considered part of DIA or vice-versa, the overlap between them includes "logical" document segmentation, extraction of tables of content, linking figures and illustrations to textual references, and word spotting. A recurring topic is assessing the effect of OCR errors on downstream applications. One factor that keeps the two disciplines apart is that IR experiments (e.g., TREC) typically involve orders of magnitude more documents than DIA

experiments because the number of characters in any collection is far smaller than the number of pixels.

Computer vision used to be easily distinguished from the image processing aspects of DIA by its emphasis on illumination and camera position. The border is blurring because even cellphone cameras now offer sufficient spatial resolution for document image capture at several hundred dpi as well as for legible text in large scene images. The correction of the contrast and geometric distortions in the resulting images goes well beyond what is required for scanned documents.

This collection suggests that we are still far from a unified theory of DIA or even OCR. The Handbook is all the more useful because we have no choice except to rely on heuristics or algorithms based on questionable assumptions. The most useful methods available to us were all *invented* rather than derived from prime principles. When the time is ripe, many alternative methods are invented to fill the same need. They all remain entrenched candidates for “best practice”. This Handbook presents them fairly, but generally avoids picking winners and losers.

“Noise” appears to be the principal obstacle to better results. This is all the more irritating because many types of noise (e.g. skew, bleed-through, underscore) barely slow down human readers. We have not yet succeeded in characterizing and quantifying signal and noise to the extent that communications science has. Although OCR and DIA are prime examples of information transfer, information-theoretic concepts are seldom invoked. Are we moving in the right direction by accumulating empirical midstream comparisons – often on synthetic data – from contests organized by individual research groups in conjunction with our conferences?

Be that as it may, as one is getting increasingly forgetful it is reassuring to have most of the elusive information about one’s favorite topics at arm’s reach in a fat tome like this one. Much as on-line resources have improved over the past decade, I like to turn down the corner of the page and scribble a note in the margin. Younger folks, who prefer search-directed saccades to an old-fashioned linear presentation, may want the on-line version.

David Doermann and Karl Tombre were exceptionally well qualified to plan, select, solicit, and edit this compendium. Their contributions to DIA cover a broad swath and, as far as I know, they have never let the song of the sirens divert them from the muddy and winding channels of DIA. Their technical contributions are well referenced by the chapter authors and their voice is heard at the beginning of each section.

Dave is the co-founding-editor of *IJDAR*, which became our flagship journal when *PAMI* veered towards computer vision and machine learning. Along with the venerable *PR* and the high-speed, high-volume *PRL*, *IJDAR* has served us well with a mixture of special issues, surveys, experimental reports, and new theories. Even earlier, with the encouragement of Azriel Rosenfeld, Dave organized and directed the Language and Media Processing Laboratory, which has become a major resource of DIA data sets, code, bibliographies, and expertise.

Karl, another *IJDAR* co-founder, put Nancy on the map as one of the premier global centers of DIA research and development. Beginning with a sustained drive

to automate the conversion of legacy drawings to CAD formats (drawings for a bridge or a sewer line may have a lifetime of over a hundred years, and the plans for the still-flying Boeing 747 were drawn by hand), Karl brought together and expanded the horizons of University and INRIA researchers to form a critical mass of DIA.

Dave and Karl have also done more than their share to bring our research community together, find common terminology and data, create benchmarks, and advance the state of the art. These big patient men have long been a familiar sight at our conferences, always ready to resolve a conundrum, provide a missing piece of information, fill in for an absentee session chair or speaker, or introduce folks who should know each other.

The DIA community has every reason to be grateful to the editors and authors of this timely and comprehensive collection. Enjoy, and work hard to make a contribution to the next edition!

July 2013

George Nagy  
Professor Emeritus, RPI

---

## Preface

Optical Character Recognition was one of the very first applications addressed in computer science, once this field extended beyond the needs for simulation and scientific computing, which boosted its birth during World War II. However, although real applications were quickly available, it progressively became clear to everybody that beyond the ability for a computer to decipher characters, there was a very vast domain of open scientific problems and hard technical challenges to be able to build complete document analysis systems, to extract the appropriate information from a vast range of documents meant to be converted from their original paper form to the appropriate digital formats.

As scientists, we have been very lucky to have been a part of this great scientific and technological adventure, throughout several decades. We have seen the field of document analysis and recognition emerge and mature, we have seen the community organize itself, and create conferences, workshops, and journals dedicated to this field. We have also seen companies integrate the best of the scientific results into great technological products, which have helped people and companies merge the paper world and the digital world, and invent new concepts of what a document is and how it is handled, stored, exchanged, and searched for.

In 2010, nearly 20 years had already passed since the first International Conference on Document Analysis and Recognition was held in Saint-Malo, France. We felt that it was time to consolidate the vast knowledge accumulated in the field into a comprehensive handbook making the state of the art available in a single volume.

It took us 3 years of work to put this book together, with the great help of a set of dedicated authors representing the very best experts in the field. The field of document image processing and recognition is primarily defined by the fact that the objects on which processing, recognition, and analysis methods are applied, are *documents* – usually captured by some kind of imaging process. But it relies on a vast set of knowledge about image processing, pattern recognition, data classification, and information retrieval. The temptation was often there to tell the reader more about the basics of these scientific areas. However, we came to the conclusion that this would lead us to putting together a complete encyclopedia, not just a handbook! Therefore, we assume that our readers are reasonably familiar with the fundamental methodology and concepts of these fields, or that they will look them up in other reference books.

We hope that the reader will find the final result as comprehensive and useful as we feel it is, thanks to the vast expertise gathered from all the senior experts who have contribute to this book.

July 2013

David Doermann

Karl Tombre

---

## Acknowledgments

The journey that has now resulted in the first edition of the *Handbook of Document Image Processing and Recognition* began with a brief conversation with Wayne Wheeler almost 4 years ago. That meeting set the wheels turning on a project that has now involved dozens of people at all levels of the publication process. Certainly it would never have been possible to compile such a definitive collection of information, without the numerous experts that wrote the chapters. These researchers, some founders of the field, others the “next generation”, are all volunteers, yet they made the time and put forth the effort necessary to establish this unparalleled reference. The community is indebted to them for their help with this project.

There were also a number of members of our community that provided input at the conceptual stage and others that provided very constructive feedback beyond their own chapters. While some of you, for various reasons, did not end up participating directly as authors, your input was very important in shaping the final product.

Most importantly, we would like to thank the production team at Springer. We thank Wayne Wheeler for convincing us to undertake this project and for his continuous motivation stating the need for such a resource. We thank associate editors Marion Kraemer and Barbara Wolf who provided much of the infrastructure support when the handbook was just becoming a reality and provided continued support and advice to educate us along the way. Finally, we would like to thank editorial assistant Saskia Ellis who coordinated the day-to-day interactions between the editors, the authors, and the production departments. Without that level of commitment and dedication on her part, we would neither have had the bandwidth nor the patience to complete this book.

Coeditors  
David Doermann  
Karl Tombre



---

## About the Editors



**David Doermann** is a member of the research faculty at the University of Maryland College Park. He received a B.Sc. degree in computer science and mathematics from Bloomsburg University in 1987, and a M.Sc. degree in 1989 in the Department of Computer Science at the University of Maryland, College Park. He continued his studies in the Computer Vision Laboratory, where he earned a Ph.D. in 1993. Since then, he has served as co-director of the Laboratory for Language and Media Processing in the University of Maryland's Institute for Advanced Computer Studies and as an adjunct member of the graduate faculty.

His team of researchers focuses on topics related to document image analysis and multimedia information processing. Their recent intelligent document image analysis projects include page decomposition, structural analysis and classification, page segmentation, logo recognition, document image compression, duplicate document image detection, image-based retrieval, character recognition, generation of synthetic OCR data, and signature verification. In video processing, projects have centered on the segmentation of compressed domain video sequences, structural representation and classification of video, detection of reformatted video sequences, and the performance evaluation of automated video analysis algorithms.

In 2002, Dr. Doermann received an Honorary Doctorate of Technology Sciences from the University of Oulu for his contributions to digital media processing and document analysis research. He is a founding co-editor of the *International Journal on Document Analysis and Recognition*, has the general chair or co-chair of over a half dozen international conferences and workshops, and is the general chair of the International Conference on Document Analysis and Recognition (ICDAR) to be held in Washington DC in 2013. He has over 30 journal publications and over 160 refereed conference papers.



**Karl Tombre**, born in Trondheim (Norway) in 1961, received a French *doctorat* (Ph.D. degree) in computer science in 1987. From 1987 to 1998, he was *chargé de recherche* (senior researcher) at INRIA, a French national institute devoted to research in computer science and applied mathematics. In September 1998, he joined *École des Mines de Nancy* at *Institut National Polytechnique de Lorraine* as full professor, and was head of its Computer Science Department from 2001 to 2007. From 2007 to 2012, he returned on secondment to INRIA, to hold the position of director of the Inria Nancy – Grand Est research center, a large research center (about 450 staff members) for computer science and applied mathematics. Since September 2012, he is vice-president of *Université de Lorraine*, one of the largest universities in France, resulting from the merger of four universities in the Lorraine region, and is in charge of economic partnerships and of international affairs.

Prof. Tombre's scientific expertise is in document image analysis and recognition, with a major focus on graphics recognition. In 2002, he founded a research group on graphics recognition and led this group until September 2007. He is also at the origin of a free software package for graphics recognition. In the period 2006–2008, he was the president of the International Association for Pattern Recognition (IAPR). Prof. Tombre is co-founder of the series of international workshops on graphics recognition, and co-chair of the two first workshops, in 1995 and 1997. A co-founder of the *International Journal on Document Analysis and Recognition* (Springer Verlag), he was one its editors-in-chiefs from its creation to August 2013. Karl Tombre is also member of several other editorial boards and has been invited to serve on numerous international conference committees.

---

# Contents

## Volume 1

<b>Part A Introduction, Background, Fundamentals .....</b>	<b>1</b>
<b>1 A Brief History of Documents and Writing Systems .....</b>	<b>3</b>
Henry S. Baird	
<b>2 Document Creation, Image Acquisition and Document Quality.....</b>	<b>11</b>
Elisa H. Barney Smith	
<b>3 The Evolution of Document Image Analysis .....</b>	<b>63</b>
Henry S. Baird and Karl Tombre	
<b>4 Imaging Techniques in Document Analysis Processes .....</b>	<b>73</b>
Basilis G. Gatos	
<b>Part B Page Analysis.....</b>	<b>133</b>
<b>5 Page Segmentation Techniques in Document Analysis .....</b>	<b>135</b>
Koichi Kise	
<b>6 Analysis of the Logical Layout of Documents.....</b>	<b>177</b>
Andreas Dengel and Faisal Shafait	
<b>7 Page Similarity and Classification.....</b>	<b>223</b>
Simone Marinai	
<b>Part C Text Recognition .....</b>	<b>255</b>
<b>8 Text Segmentation for Document Recognition.....</b>	<b>257</b>
Nicola Nobile and Ching Y. Suen	
<b>9 Language, Script, and Font Recognition .....</b>	<b>291</b>
Umapada Pal and Niladri Sekhar Dash	

<b>10</b>	<b>Machine-Printed Character Recognition</b>	331
	Huaigu Cao and Prem Natarajan	
<b>11</b>	<b>Handprinted Character and Word Recognition</b>	359
	Sergey Tulyakov and Venu Govindaraju	
<b>12</b>	<b>Continuous Handwritten Script Recognition</b>	391
	Volkmar Frinken and Horst Bunke	
<b>13</b>	<b>Middle Eastern Character Recognition</b>	427
	Abdel Belaïd and Mohamed Imran Razzak	
<b>14</b>	<b>Asian Character Recognition</b>	459
	Srirangaraj Setlur and Zhixin Shi	

## Volume 2

<b>Part D</b>	<b>Processing of Non-textual Information</b>	487
<b>15</b>	<b>Graphics Recognition Techniques</b>	489
	Josep Lladós and Marçal Rusiñol	
<b>16</b>	<b>An Overview of Symbol Recognition</b>	523
	Salvatore Tabbone and Oriol Ramos Terrades	
<b>17</b>	<b>Analysis and Interpretation of Graphical Documents</b>	553
	Bart Lamiroy and Jean-Marc Ogier	
<b>18</b>	<b>Logo and Trademark Recognition</b>	591
	Anastasios Kesisidis and Dimosthenis Karatzas	
<b>19</b>	<b>Recognition of Tables and Forms</b>	647
	Bertrand Coüasnon and Aurélie Lemaitre	
<b>20</b>	<b>Processing Mathematical Notation</b>	679
	Dorothea Blostein and Richard Zanibbi	
<b>Part E</b>	<b>Applications</b>	703
<b>21</b>	<b>Document Analysis in Postal Applications and Check Processing</b>	705
	Michel Gilloux	
<b>22</b>	<b>Analysis and Recognition of Music Scores</b>	749
	Alicia Fornés and Gemma Sánchez	
<b>23</b>	<b>Analysis of Documents Born Digital</b>	775
	Jianying Hu and Ying Liu	

<b>24</b>	<b>Image Based Retrieval and Keyword Spotting in Documents .....</b>	805
	Chew Lim Tan, Xi Zhang, and Linlin Li	
<b>25</b>	<b>Text Localization and Recognition in Images and Video .....</b>	843
	Seiichi Uchida	
	<b>Part F Analysis of Online Data.....</b>	<b>885</b>
<b>26</b>	<b>Online Handwriting Recognition.....</b>	887
	Jin Hyung Kim and Bong-Kee Sin	
<b>27</b>	<b>Online Signature Verification.....</b>	917
	Réjean Plamondon, Giuseppe Pirlo, and Donato Impedovo	
<b>28</b>	<b>Sketching Interfaces .....</b>	949
	Tong Lu and Liu Wenyin	
	<b>Part G Evaluation and Benchmarking .....</b>	<b>981</b>
<b>29</b>	<b>Datasets and Annotations for Document Analysis and Recognition .....</b>	983
	Ernest Valveny	
<b>30</b>	<b>Tools and Metrics for Document Analysis Systems Evaluation .....</b>	1011
	Volker Märgner and Haikal El Abed	
	<b>Correction to: Language, Script, and Font Recognition .....</b>	C1
	Umapada Pal and Niladri Sekhar Dash	
	<b>Index.....</b>	<b>1037</b>



---

## Contributors

**Henry S. Baird** Computer Science and Engineering Department, Lehigh University, Bethlehem, PA, USA

**Elisa H. Barney Smith** Electrical & Computer Engineering Department, Boise State University, Boise, ID, USA

**Abdel Belaïd** Université de Lorraine – LORIA, Vandœuvre-lès-Nancy, France

**Dorothea Blostein** School of Computing, Queen's University, Kingston, Canada

**Horst Bunke** Research Group on Computer Vision and Artificial Intelligence (FKI), Institute of Computer Science and Applied Mathematics, University of Bern, Bern, Switzerland

**Huagu Cao** Raytheon BBN Technologies, Cambridge, MA, USA

**Bertrand Coëtiasnon** IRISA/INSA de Rennes, Rennes Cedex, France

**Niladri Sekhar Dash** Linguistic Research Unit, Indian Statistical Institute, Kolkata, India

**Andreas Dengel** German Research Center for Artificial Intelligence (DFKI GmbH), Kaiserslautern, Germany

**David Doermann** Institute for Advanced Computer Studies, University of Maryland, College Park, MD, USA

**Haikal El Abed** Institute for Communications Technology, Technische Universität Braunschweig, Braunschweig, Germany

**Alicia Fornés** Computer Vision Center & Computer Science Department, Universitat Autònoma de Barcelona, Bellaterra, Spain

**Volkmar Frinken** Computer Vision Center, Autonomous University of Barcelona, Bellaterra, Spain

**Basilis G. Gatos** Institute of Informatics and Telecommunications, National Center for Scientific Research “Demokritos”, Agia Paraskevi, Athens, Greece

**Michel Gilloux** Seres/Docapost EBS (Groupe La Poste), Nantes, France

**Venu Govindaraju** Department of Computer Science & Engineering, Center for Unified Biometrics and Sensors, University at Buffalo, Amherst, NY, USA

**Jianying Hu** IBM T.J. Watson Research Center, Yorktown Heights, NY, USA

**Donato Impedovo** Politecnico di Bari, Bari, Italy

**Dimosthenis Karatzas** Computer Vision Center, Universitat Autònoma de Barcelona, Bellaterra, Spain

**Anastasios Kesisidis** Department of Surveying Engineering, Technological Educational Institution of Athens, Aigaleo, Athens, Greece

**Jin Hyung Kim** Department of Computer Science, Korea Advanced Institute of Science and Technology, Yuseong-gu, Daejeon, Korea

**Koichi Kise** Department of Computer Science and Intelligent Systems, Graduate School of Engineering, Osaka Prefecture University, Sakai, Osaka, Japan

**Bart Lamiroy** Université de Lorraine – LORIA, Vandœuvre-lès-Nancy Cedex, France

**Aurélie Lemaitre** IRISA/Université Rennes 2, Rennes Cedex, France

**Linlin Li** National University of Singapore, Singapore, Singapore

**Ying Liu** Korea Advanced Institute of Science and Technology (KAIST), Yuseong-gu, Daejeon, Republic of Korea

**Josep Lladós** Computer Vision Center & Computer Science Department, Universitat Autònoma de Barcelona, Bellaterra, Spain

**Tong Lu** Department of Computer Science and Technology, State Key Laboratory for Novel Software Technology at Nanjing University, Nanjing, China

**Volker Märgner** Institute for Communications Technology, Technische Universität Braunschweig, Braunschweig, Germany

**Simone Marinai** Dipartimento di Ingegneria dell'Informazione, Università degli studi di Firenze, Firenze, Italy

**Prem Natarajan** Information Sciences Institute, University of Southern California, Marina Del Rey, CA, USA

**Nicola Nobile** Centre for Pattern Recognition and Machine Intelligence (CENPARMI), Concordia University, Montréal, QC, Canada

**Jean-Marc Ogier** Université de La Rochelle, La Rochelle Cédex 1, France

**Umapada Pal** Computer Vision And Pattern Recognition Unit, Indian Statistical Institute, Kolkata, India

**Giuseppe Pirlo** Dipartimento di Informatica, Università degli Studi di Bari, Bari, Italy

**Réjean Plamondon** Département de génie électrique, Ecole Polytechnique de Montréal, Montréal, QC, Canada

**Oriol Ramos Terrades** Universitat Autònoma de Barcelona – Computer Vision Center, Bellaterra, Spain

---

**Mohamed Imran Razzak** Department of Computer Science and Engineering, Air University, Pakistan

College of Public Health and Health Informatics, King Saud bin Abdulaziz University for Health Sciences, Riyadh, Saudi Arabia

**Marçal Rusiñol** Computer Vision Center & Computer Science Department, Universitat Autònoma de Barcelona, Bellaterra, Spain

**Gemma Sánchez** Computer Vision Center & Computer Science Department, Universitat Autònoma de Barcelona, Bellaterra, Spain

**Srirangaraj Setlur** Department of Computer Science and Engineering, University at Buffalo, The State University of New York, Buffalo, NY, USA

**Faisal Shafait** Multimedia Analysis and Data Mining Competence Center, German Research Center for Artificial Intelligence (DFKI GmbH), Kaiserslautern, Germany  
School of Computer Science and Software Engineering, The University of Western Australia, Perth, Australia

**Zhixin Shi** Department of Computer Science and Engineering, University at Buffalo, The State University of New York, Buffalo, NY, USA

**Bong-Kee Sin** Department of IT Convergence and Applications Engineering, Pukyong National University, Namgu, Busan, Korea

**Ching Y. Suen** Department of Computer Science and Software Engineering, Centre for Pattern Recognition and Machine Intelligence (CENPARMI), Concordia University, Montréal, QC, Canada

**Salvatore Tabbone** Université de Lorraine – LORIA, Vandœuvre-lès-Nancy Cedex, France

**Chew Lim Tan** Department of Computer Science, National University of Singapore, Singapore, Singapore

**Karl Tombre** Université de Lorraine, Nancy, France

**Sergey Tulyakov** Center for Unified Biometrics and Sensors, University at Buffalo, Amherst, NY, USA

**Seiichi Uchida** Department of Advanced Information Technology, Kyushu University, Nishi-ku, Fukuoka, Japan

**Ernest Valveny** Departament de Ciències de la Computació, Computer Vision Center, Universitat Autònoma de Barcelona, Bellaterra, Spain

**Liu Wenyin** College of Computer Science and Technology, Shanghai University of Electric Power, Shanghai, China

**Richard Zanibbi** Department of Computer Science, Rochester Institute of Technology, Rochester, NY, USA

**Xi Zhang** National University of Singapore, Singapore, Singapore

---

## Part A

### Introduction, Background, Fundamentals

Suzanne Briet (1894–1989) is recognized for her pioneer role in laying the foundations of modern information science. In her manifesto titled *Qu'est-ce que la documentation?* (“What is documentation?”), she defines a document as *evidence in support of a fact; a physical or symbolic sign, preserved or recorded, for representing, reconstructing or demonstrating a physical or a conceptual phenomenon*. Thus, a document *documents* something, it is a representation serving as evidence for some purpose. We can also say that the information it provides is presented in such a way that the producer of the document becomes able to convey this information to its users/readers, in an intelligible way. This again means that producer and reader use shared representation rules for the information.

Compared to oral communication, the main difference is the preservation of the representation, so that the consumer of the information does not need to communicate directly and immediately with its producer. The most ancient and well-known document type is of course the written document; the introduction of writing was of such importance for mankind that historians usually consider that history starts at that moment, when knowledge of the past can be recovered from written records.

In ►[Chap. 1](#) (A Brief History of Documents and Writing Systems), Henry Baird introduces us to writing systems and briefly reviews the history of written documents. Throughout the centuries, these documents have been physically produced on various materials, using many different writing or printing processes, until this century's documents which are very often originally available in electronic format. Elisa Barney Smith reviews the evolution of these creation processes and equipment in ►[Chap. 2](#) (Document Creation, Image Acquisition and Document Quality). She also introduces us to the numerous acquisition processes for converting physical documents into images for further processing by appropriate software tools.

When people hear about document image processing, they probably think first and foremost of optical character recognition (OCR). But in the same way as children do not only learn to decipher characters, but progressively learn to understand complete documents, document analysis systems also have gone much further than just OCR. This evolution from character to full and complex documents

is presented by Henry Baird and Karl Tombre in ►[Chap. 3](#) (The Evolution of Document Image Analysis).

A common, fundamental toolbox for all these document analysis systems is that of image processing algorithms applied to document images. Image processing as a whole would deserve a full handbook on its own, but in ►[Chap. 4](#) (Imaging Techniques in Document Analysis Processes), Basilis Gatos presents an overview of the most fundamental image processing methods needed in any document processing and recognition system.

---

# A Brief History of Documents and Writing Systems

1

Henry S. Baird

## Contents

Introduction.....	4
The Origins of Writing.....	4
Writing System Terminology.....	4
Reading Order and Segmentation.....	5
Types of Writing Systems.....	7
Origins of Writing Media.....	8
Punctuation.....	9
Conclusion.....	9
Cross-References.....	10
References.....	10
Further Reading.....	10

---

### Abstract

This chapter provides a review of the history of written language, with emphasis on the origins and evolution of characteristics which have been found to affect – and in some cases continue to challenge – the automated recognition and processing of document images.

---

### Keywords

Alphabet • Document • Font • Glyph • Graph • Grapheme • Image quality • Language • Layout style • Punctuation • Reading order • Script • Style • Syllabary • Text blocks • Text lines • Typeface • Writing system • Words

---

H.S. Baird

Computer Science and Engineering Department, Lehigh University, Bethlehem, PA, USA  
e-mail: [baird@cse.lehigh.edu](mailto:baird@cse.lehigh.edu)

## Introduction

Gaur's profusely illustrated history of writing [2] makes clear that in addition to characters handwritten or machine-printed on flat surfaces, human communication embraces rock paintings (by many prehistoric people), message sticks (Aboriginal Australians), beans marked with dots and lines (Incas), *quipu* (knotted cords of hair or cotton; Inca plus China, Africa, Polynesia, etc.), wampum belts (North American Natives), sets of cowrie shells (Yoruba of Nigeria), and strings of tally sticks (Torres Strait Islands). Although none of the above has yet to be read automatically by computers, it is conceivable that the document image analysis R&D community will someday attempt them.

---

## The Origins of Writing

Gaur distinguishes between "thought writing" (which "transmits an idea directly," e.g., "the drawing of a tree means "tree") and "sound writing" in which the "phonetics" of speech is made visible by a conventional set of signs (Sampson calls this a "glottographic" system [7]). Parkes [6] describes the early relationship between speech and writing in the West as follows:

In Antiquity the written word was regarded as a record of the spoken word, and texts were usually read aloud. But from the sixth century onwards attitudes to the written word changed: writing came to be regarded as conveying information directly to the mind through the eye, and Isidore of Seville (c. 560–636 CE) could state a preference for silent reading which subsequently became established as the norm.

Daniels and Bright's [1] 1996 survey of the world's writing systems lists over 50 major families, some with a dozen or more subfamilies. The vast diversity of (especially phonetic) writing conventions suggests that many are largely arbitrary cultural inventions: this variety remains of course the single most confounding technical challenge to automatic recognition. The key stages of the evolution of writing systems are also imperfectly understood and may always remain so. Perhaps as a result, the present state of writing systems appears largely chaotic: few broadly applicable rules are evident. Even systems with long historical records – notably Chinese – tend to defy completely systematic analysis. Similar pessimism, qualified no doubt by progress in modern linguistics, might be extended to the thousands of languages known, only a small fraction of which enjoy a writing system. (In 2009, the SIL Ethnologue [4] listed 6,909 living human languages and estimated that 7,000–10,000 more or less distinct living languages exist.)

---

## Writing System Terminology

First, a brief review of terms used to describe the appearance of writing systems, starting, as modern document image analysis systems usually do, with an image of a single sheet of paper (a "page") with a message inked on it. This image may contain

a mixture of text and non-text regions. Textual regions typically contain blocks (or “columns”) of text organized into text “lines,” which (according to the language) may run horizontally or vertically (very rarely, in a spiral). Within a block, text lines are read typically top to bottom (for horizontal lines) and left to right (for vertical lines); this choice, which may seem arbitrary, nevertheless holds, interestingly, for many ancient texts as well as for virtually all modern texts. The order of reading within text lines also varies by language; in some ancient texts, the order switches from line to line so that if one line was read left to right, the next was read right to left (a technical term for this is *boustrophedonic*, from Greek “as the ox plows”). Text lines contain symbolic images of words from the language (and punctuation, discussed below). Almost universally, the words are written in the same order that they would be spoken.

Resuming the review of basic terminology, into what smaller elements should a text-line image be segmented? In all Western European (and many other) writing systems, a “word-space” convention assists breaking text lines into “word” images; although, these may contain punctuation and thus not map directly onto linguistic words. Even in these systems, automatic segmentation can be difficult to achieve reliably using purely “geometric” cues (such as the distribution of horizontal spaces separating characters, scaled by estimates of local type size); ambiguities often require assistance from symbol recognition and even from higher levels of interpretation.

---

## Reading Order and Segmentation

Since in spoken language words occur in time, in sequence one after the other, almost all phonetic writing is also arranged linearly in space. By contrast, most writing media are two dimensional (at least), but the linear convention copied from speech seldom exploits these extra dimensions. Some “primitive” writing such as the Yukaghir messages discussed by Sampson does not encode any fixed word order and so can be “read out loud” in a multiplicity of narratives; and some modern “super-textual” writing, such as mathematics and music, expands beyond one dimension (more on this later in the book). The fact that once a page of text has been decomposed (“segmented”) into blocks and text lines and the intended reading order inferred, recognition faces a decisively simpler class of linearized problems.

Independently of the document analysis community, the speech recognition (and more generally the computational linguistics) R&D community had, beginning in the 1970s, discovered the power of a class of dynamic programming optimization algorithms in analyzing times series problems. Methods that depend for efficiency on linear ordering include grammars, Markov models, Hidden Markov models, dynamic time warping, finite state transducers, and so forth. These algorithmic advances, which revolutionized other fields, were slow to penetrate the document analysis community until the early 1990s; but this process has now taken hold. It is not widely appreciated that most of these dynamic programming methods depend for their efficiency on special properties (often called “optimal substructure,” also known, earlier, as “the principle of optimality”) that apply to many one-dimensional

problems but which rarely generalize to higher dimensions. Many two (and higher)-dimensional optimization problems seem to be intrinsically harder in this sense: this may account, in part, for the relatively slower evolution of layout analysis methods compared to text recognition methods.

Other languages, notably the major modern East Asian languages, lack a word-space convention, and so the next level of segmentation must be directly related to individual symbols. In some writing systems, such as Arabic, a linguistic word is written as a sequence of spaced-out connected groups of symbols (“subwords”): true word spaces exist alongside interword breaks and thus complicate segmentation. Much handwriting is cursive, in which many (or all) symbols within a word are connected. Even in some machine-printed systems, such as Arabic, calligraphic influences strongly survive, and typefaces are designed to imitate careful but still cursive handwriting.

It is natural to suppose that all these language and writing system dependent policies have been modeled and implemented as a segmentation algorithm able to detect and isolate, from an image of a text line, each individual *symbol*, which normally will be a fundamental unit of the written language, such as a character shape permitted by the alphabet or syllabary.

However, exceptions to a straightforward one-to-one mapping between linguistic symbols and images of symbols are surprisingly frequent. Ligatures (and digraphs, and contractions generally) merge two or more linguistic symbols into a single written character. In some writing systems, e.g., Medieval manuscripts, the number of contractions permitted can outnumber the letters of the underlying alphabet. The impact on document recognition engineering can be daunting: in effect, for purposes of image recognition, the alphabet has expanded, perhaps by a large factor, increasing the effort of collecting labeled samples for each class. More seriously perhaps, the set of “characters” to be found in the document image may be unknown at the outset; variations may be discovered on the fly; what is a legitimate variation – not a typo or misinterpretation – may not be clear; and professional historians may need to be consulted. In this sense, many calligraphic writing systems, even in the West, are “open,” lacking a fixed set of conventional letterforms.

An image of a correctly isolated symbol is called a *graph* (some authorities prefer the term *graphemes* for what are here called symbols). Now consider the set of all graphs segmented from a document image; the task of a *character classifier* is to assign to each graph its correct linguistic character label (in the case of contractions, the correct output is a string of linguistic labels). Now graphs for the same symbol can be expected to differ in detail, due to the vagaries of printing (e.g., text size, inking, paper quality), handwriting, imaging (point-spread function, scanning resolution, etc.), and variations even in segmentation style. Of course such variations are a principal technical challenge of text-image classifier design.

But there are deeper challenges, due to variations of other kinds. In some writing systems, more than one shape is allowed to be used to represent a single symbol: such a set of visually dissimilar but linguistically identical character shapes are sometimes called *allographs*. The underlying shapes may be so different that they must, as a practical matter of classifier training, be separated into distinct classes:

in this case, once again, the classes required for image recognition cannot be mapped one to one onto linguistic classes. But, from another point of view, an inability to generalize, during training, across dissimilar allographs can be judged a symptom of inadequacy in the trainable classifier technology, and if this criticism is justified, then couldn't the technology also fail to generalize across other variations such as extremes of image quality? Indeed, document recognition engineers often feel the necessity of making manual adjustments – which a linguist might regard as irrelevant and distracting interventions – to the labeling of training sets, splitting and combining classes, or organizing them into tree structures. The hope of minimizing this potentially open-ended manual “tuning” is one motive for trying classification trees (CARTs); unfortunately, training good trees has always been either computationally prohibitive or weakly heuristic. Note that these problems, due to certain “open-ended” characteristics of writing systems and typographic conventions, can arise even for modern languages in high-technology cultures.

Furthermore, there is the problem (and promise) of *style*: an individual's writing personality is an example, as are typefaces in machine-print; also image quality can be thought of helpfully as a kind of style (much more on this later).

---

## Types of Writing Systems

Harris' 1986 history of writing systems [3] attempted to classify the various types of symbols (he says “signs”) used in the writing systems of the world as follows:

*Alphabetic*: a set of symbols representing the complete set of consonants (e.g., “s”) and vowels (e.g., “a”) occurring in speech, as in English and most classical and modern Western scripts (perhaps “ultimately from the North Semitic alphabet of the 2nd half of the 2nd millennium B.C.”)

*Syllabic*: a set of symbols, one for each syllable (short consonant-vowel or consonant-vowel-consonant combination), e.g., “ka” (Japanese)

*Logographic*: a set of symbols “representing words but giving no indication of pronunciation,” as in “\$” to indicate “dollar,” and frequently throughout the Chinese Han system (used also in Japan and Korea)

*Pictographic*: symbols “which take the form of a simplified picture of what they represent,” as in “a circle with rays” to indicate the sun and arguably in certain Egyptian hieroglyphs

*Ideographic*: symbols “representing an idea of a message as a whole, rather than any particular formulation of it,” as in “an arrow sign” to indicate direction

Although this taxonomy is simplistic (and still somewhat controversial), it should be clear enough for the purposes of this chapter. The principal implications for document recognition are that (a) alphabetic, syllabic, and logographic systems dominate almost all modern (and many ancient) scripts; (b) the recognition of pictographic and ideographic writing systems has been relatively neglected by the OCR community (except for “logo” recognition in business documents), though this may change radically as the challenges of “cityscape scenes” are taken more seriously, including the problems of detection, isolation, recognition, and interpretation of traffic signs and the rapidly growing set of “international” signs and symbols; (c) alphabets tend to be much smaller than syllabaries which in turn

are very much smaller than logographic sets, with important implications for the engineering costs of supervised training; and (d) while alphabets and syllabaries are typically “closed” (complete and fixed), logographic systems tend to be “open” (incomplete, freely extensible).

It is difficult to generalize across all the variations exhibited in writing systems. However, one strong tendency, in virtually all phonetic writing systems is towards the use of compact “physical support” for individual symbol images: that is, they tend all to fit within small non-overlapping cells of approximately equal size.

The implications for document image recognition are daunting: in order to process a new language, several hurdles must be overcome, including: descriptions of all the graphemes used, collection of samples of glyphs (many for each grapheme, and more for each distinct style), analysis of page layout conventions, amassing dictionaries (lexica or morphological analyzers), at least. Some of these hurdles may require assistance from professional linguists.

---

## Origins of Writing Media

A wide range of materials have served for early writing: Gaur highlights stone, leaves, bark, wood generally, clay, skin, animal bones, ivory, bamboo, tortoiseshell, and many metals notably copper and bronze. Although relatively perishable, one Egyptian wooden writing board survives from about 2000 BCE. “Some of the earliest Chinese writing” survives on “oracle” bones from about 1700 BCE. Waxed writing tablets, conveniently reusable, originated as early as the 8th C. BCE, and were used pervasively by the ancient Greeks and Romans; Roman laws, however, were published by inscription on bronze tablets which were displayed on doors.

The scale of production of certain writing media grew remarkably, even in ancient times, starting with clay tablets in Mesopotamia and continuing with papyri in Egypt. In South and Southeast Asia, palm leaves were the dominant medium until modern times. Vast corpora of palm leaves survive, many containing Jain, Buddhist, and Hindu scriptures: these have already been the object of serious document image recognition research. The rapid growth of interest worldwide in preservation of and access to historical documents seems likely to leave few of these arcane document types untouched, and reveal many new technical challenges.

Note that each of the three writing cultures above amassed huge collections of documents which were apparently intended to be highly uniform in material, size, and appearance, including the order in which symbols were written and in the shapes of symbols. The evidence is strong for large cadres of professional scribes trained in uniform practices. The wide diversity – indeed profusion of creative variations – in modern writing styles, which one may be tempted today to take for granted, was not the norm in early societies, and it accelerated only with the industrial age. A significant technical trend in today’s document recognition research is interest in *style-conscious* methods which can exploit known (or merely guessed) uniformity on the input images. The older a written corpus is, the more likely it is to be

crafted in a uniform style: thus modern style-conscious methods may turn out to be particularly (even surprisingly) effective when applied to pre-modern documents.

Another important implication is that each medium could, and often did, affect the evolution of writing styles. For example, the introduction of serifs occurred in monumental classical inscriptions (such as a highly influential Trajan column), driven by technical constraints peculiar to carving (chiseling) marble. The survival of serifs into modern times has ostensibly been due to aesthetics, though one could argue that they also aid in legibility.

Some writing materials were (and remain) far more costly than others. The expense of relatively long-lasting media such as vellum drove the development of elaborate Medieval scribal conventions to save space, including an explosion of concise contractions and diacritical marks.

---

## Punctuation

Parkes' profusely illustrated 1993 study [6] showed that, in the West at least, by the Middle Ages,

punctuation became an essential component of written language. Its primary function is to resolve structural uncertainties in a text, and to signal nuances of semantic significance . . . .

Nevertheless the functions of punctuation have received relatively little attention by classical and even modern computational linguists. One exception is Sproat's 2000 formal theory of orthography [8] embracing several modern writing systems including Russian, Chinese, and Hangul (Korean): his principal aim is to analyze encoded text corpora in order to drive (control) an intelligible text-to-speech synthesis system; he shows that this requires finite-state models at both "shallow" and "deep" levels; and he suggests that complete models of this kind are unlikely to be learnable from training data using purely statistical inference. Nunberg's 1990 thoughtful study [5] revealed that punctuation rules in English are more complex than the regular expressions used in state-of-the-art OCR machines would suggest.

---

## Conclusion

Some clear trends in the history of writing systems that are potentially important to the document image analysis R&D community have not, as far as is known, received sustained scholarly attention of any kind. Careful studies of the reasons for the early and persistent dominance of monochrome (bilevel) documents are not known to the present writer. Although much is known about the evolution of certain, mostly Western and Asian, alphabets (and syllabaries, ideographic systems, etc.), details are often missing concerning the crucial shifts from open-ended symbol sets to finite and fixed sets. (It is interesting to contrast this fact with the persistent open-endedness of dictionaries in all living languages.) In most writing systems with long histories, the graphs of symbols have evolved steadily from complex

to relatively simplified forms. Within living memory, the Han writing system underwent a dramatic refinement to smaller symbol sets and simplified glyph forms. One such event, which now seems anomalous and even embarrassing, occurred when manufacturers of early OCR systems despaired of coping with naturally occurring printed text and invented “OCR fonts” such as OCR-A and OCR-B to make their problem simpler and then seriously (if ineffectively) proposed them for widespread commercial use.

Perhaps most of these events, all of which have tended to simplify the task of recognition, have been driven by a slow but inexorable broader impulse towards standardization and automation. But this explanation begs the question of whether or not simplification will continue unabated.

---

## Cross-References

- ▶ [Document Creation, Image Acquisition and Document Quality](#)
  - ▶ [The Evolution of Document Image Analysis](#)
- 

## References

1. Daniels PT, Bright W (eds) (1996) *The world's writing systems*. Oxford University Press, Oxford
2. Gaur A (1984) *A history of writing*. Cross River Press, New York
3. Harris R (1986) *The origin of writing*. Open Court, La Salle
4. Lewis MP (ed) (2009) *Ethnologue: languages of the world*, 16th edn. SIL International, Dallas. Online version: [www.ethnologue.com](http://www.ethnologue.com)
5. Nunberg G (1990) *The linguistics of punctuation*. CSLI: Center for the Study of Languages and Information, Menlo Park
6. Parkes MB (1993) *Pause and effect: an introduction to the history of punctuation in the West*. University of California Press, Berkeley
7. Sampson G (1985) *Writing systems*. Stanford University Press, Stanford
8. Sproat R (2000) *A computational theory of writing systems*. Cambridge University Press, Cambridge

## Further Reading

The voluminous references offered here may suggest interesting further reading.

---

# Document Creation, Image Acquisition and Document Quality

2

Elisa H. Barney Smith

## Contents

Introduction .....	12
Document Creation Materials .....	12
Writing Substrates .....	12
Inks .....	17
Writing and Printing Processes .....	20
HandHeld Writing Instruments .....	20
Machine Printing .....	23
Acquisition Methods .....	39
Flatbed Scanner and Fax Machine Acquisition .....	39
Cameras and Mobile Devices .....	44
Video .....	46
Other Specialty Modes .....	47
Document Quality .....	48
Factors Affecting Document Quality .....	48
Effects of Document Quality on Analysis and Recognition Results .....	50
Models of Document Degradations .....	51
Conclusion .....	59
References .....	60
Further Reading .....	60

---

### Abstract

A summary of materials used in creating documents, methods of creating the printed document, and methods to acquire a digital version of that document are presented. Current as well as historical methods, materials, and processes are presented. Along with this, a discussion of places where image degradations can enter the process is included. All this is related to how these aspects could affect document recognition ability.

---

E.H. Barney Smith

Electrical & Computer Engineering Department, Boise State University, Boise, ID, USA  
e-mail: [EBarneySmith@boisestate.edu](mailto:EBarneySmith@boisestate.edu)

**Keywords**

Acquisition methods • Document degradations • Document defects • Document quality • Ink • Paper • Printing • Scanning

---

## Introduction

Documents can be created by hand or by machine. In either case, several factors affect the final appearance, including the content, pigment, instrument transferring pigment to the paper, and the paper itself. How either people or machines perceive document appearance depends on how it is acquired. What is considered good quality on paper, when received directly by a human eye and processed by a human brain, is not always considered good quality when digitized and then viewed on a monitor. Likewise, what a person considers good perceptual quality on the original or digitized version is not always of a quality that can make the document content recognizable by a high precision machine.

To help explain the relationship between document sources and their quality, this chapter identifies junctures at which quality can decrease as it describes:

- Materials – materials, such as paper and ink; people: and machines use to create a document.
  - Processes – current and obsolete processes for creating printed text by hand or machine. Particular obsolete processes are noted for technologies archivists see in historical document collections.
  - Acquisition methods – methods for converting documents to digital form, facilitating automatic document image processing and recognition.
  - Models – document production models, quality measures, and how quality affects recognition results.
- 

## Document Creation Materials

This is an overview of some of the materials that have been used over time to create documents and which significantly shape their appearance. Materials include the substrate on which the document appears, usually considered a form of “paper,” and the ink that shows the written message. The choice of paper and ink is partially historical, set by available materials and technologies, and partially by the writing or printing process. Transferring the ink to the substrate can be done by hand with a writing instrument or by machine through a printing process. Figure 2.1 includes examples of different materials and methods for writing and printing and when they were first introduced. Each has introduced a new variable affecting the appearance of a final document.

## Writing Substrates

Writing substrates are writing surfaces. Surprisingly, while there have been many since the earliest humans first began writing, many substrate fundamentals have not

**Fig. 2.1** Timeline of document creation materials and methods

### History of Document Creation

Stone Clay	
Tree Bark	
Papyrus – 3500BC	450 AD – Letterpress
Carbon/India Ink – 2500BC	8 <sup>th</sup> C AD – Xylography
Parchment – 150BC	1040 – Moveable Type (China)
Paper – 100AD	1440 – Moveable Type (Gutenberg)
Iron Gall Ink – 400AD	1796 – Lithography
Quill Pens – 600AD	
Paper in Europe – 1100	
Pencils – 1564	
Mechanical Pencils – 1820s	1852 – Copper Plate Gravure
Metal Nib tip pens – 1822	1855 – Photolithography
Fountain Pens – 1827	1868 – Typewriter
Wood Paper – 1840	1870s – Hectograph
Ballpoint Pens – 1944	1875 – Offset Lithography
Felt tip Pens – 1964	1880 – Gravure Press
ePens – 1980s	1887 – Mimeograph
Gel Pens – 1984	
eReaders – 1998	1920 – Microfilm/fiche
	1923 – Ditto Machine
	1948 – Xerography
	1969 – Laser Printer
	1970 – Dot Matrix Printer
	1970 – Thermal Printers
	1970 – Inkjet Printer

changed all that much. The oldest writing available to study is preserved because it was written or carved on stone or because it was impressed into clay tablets. While these materials have longevity and in certain areas are plentiful, they are not particularly portable. Almost any portable substance that would retain the marks of a brush or pen was used as a writing substrate. This includes leaves, tree bark, boards, and cloth. In China, old writing has been found on bamboo sticks and in India writing on birch bark and palm leaves. The Mayan wrote on “paper” from the inner bark of the fig tree which was coated with a thin layer of a plaster like substance, and the text was painted onto the “paper”-like stucco painting. Not unlike today’s books, these were folded fanlike into a book form.

### From Papyrus to Parchment and Paper

The most well known of ancient writing substrates is *papyrus* from which the modern word paper derives. As early as 3500 BC, Egyptians used papyrus to form their paper. Papyrus is a type of reed called a sedge. The rind was removed to expose

the soft inner pith which was flattened, and strips were laid out in overlapping layers at right angles. This was cemented by beating the pith until the plant tissue ruptured, and the sap from the tissues formed a glue that would hold the strips together. The material was dried under pressure and polished on one side to form a smooth surface on which writing would occur. The standard writing unit or what we consider today to be a “page” evolved from the size of one of these units. Several of these units (on the order of 20) were combined by overlapping the edges from one unit to the next and cementing those in a similar fashion to form a roll, which became known as a *volumen* from the Latin word “to roll.” Each scroll contained about as much information as seven to ten pages of a modern handwritten book. The word *book* came from the name of the port, Byblos, through which the Greeks imported papyrus in the late Iron Age.

*Parchment* is a writing material made from stretched and untanned animal skins, particularly, calves, sheep, or goats. While leather was used for writing since 2000 BC, it didn’t store well and could only be written on one side. Parchment became commonly used for writing once a method was developed in the second century BC to allow both sides to be used for writing, although the interior side had a smoother surface. In Europe, it became the primary writing substrate from the fourth century AD until the Renaissance and the introduction of paper. Parchment made from fine skins from young calves or goats is called *vellum*. With the use of parchment, the writing substrate was no longer rolled but bound into codices, as we do with today’s printed books. To print, the Latin Bible required more than 500 calf hides. The quantity of hides necessary to create books made them quite expensive; thus, it became common to reuse parchment. The ink was scraped off the parchment or the writing was done at right angles to visually distinguish the new writing from the traces of the old. Books with this reuse are called *palimpsest*, from the Greek “to scrape again.” Even with this reuse, depending on parchment as a substrate limited book production.

*Paper* as we consider it today originated in China in the first century AD. The Chinese kept the process for creating paper a secret for many centuries, until they attacked the Arab city of Samarkand in the eighth century, and the Arabs took prisoner some of the Chinese skilled in making paper. Papermaking then moved west as the Arabs expanded their control in Europe. The first paper mill in Europe was founded in 1100 AD in Constantinople, and papermaking spread rapidly once in Europe until by the fourteenth century it was established throughout Europe. The introduction of paper in Europe led to an increase in the production of books and coincided with an increase in readers.

## **Paper Production**

Paper production begins by shredding and reducing vegetable fibers to a pulp in water. A thin layer of pulp is spread on a screen, and the water is drained off to form a felt. Pulp fibers are matted and dried. Process differences result in the variations among available papers. Paper was produced by manual methods for several centuries, but this limited the quantity or size of the sheet that could be

produced at any one running of the process. The first mechanized papermaking process was invented in 1798 by Nicolas Louis Robert and was made commercially practicable in 1805 by Henry and Sealy Fourdrinier.

Initially, the fibers used in making paper were made primarily from linen, jute, flax, and hemp. Fibers from cloth rags were a common source through the seventeenth century. Paper made from cotton fibers is called *rag paper*. These papers are used commonly today for high-quality documents and banknotes. Using fiber from straw was experimented with in the eighteenth century, and esparto grass was used frequently in England in the nineteenth century. A bleaching process was developed so white paper could be made from colored fibers. Starting in Saxony in the mid-1800s, most modern paper is made from tree cellulose, and the discovery that this was a suitable source greatly increased the paper supply. Wood is reduced to a pulp by either mechanical or chemical methods. The mechanical grinding of wood pulp between stone grindstones introduces many impurities, which decrease the paper's quality. Wood cell walls are constructed from a network of cellulose filled in with lignin. Mechanical grinding does not remove the lignin so the yield is higher, but the lignin over time yellows the paper and makes it brittle. An alternative was developed to add chemical reagents such as soda and sulfate, to break down the lignin that holds the cells together. If the chemical agent is caustic soda, soft fluffy fibers result that are good for cover and writing paper. Calcium bisulphate or magnesium bisulfate produces a stronger or harder fiber to create a paper better suited for printing. These acids can lead to deterioration of the paper and the inks over time, so sodium sulfate is an alternative that makes a very tough paper. The fibers will be longer in chemical pulping than in mechanical pulping, and thus, the paper will be stronger.

*Acid-free paper* has a neutral pH from being treated with a mild base (usually calcium or magnesium bicarbonate) to neutralize the natural acids occurring in wood pulp and in the production process. It is also lignin and sulfur free. It is suitable for archival purposes because it will not yellow or become brittle for a long period of time. If an adequate alkaline reserve is included, such paper will likely survive 1,000 years.

Until the late eighteenth century, paper was mostly *laid*. The fibers were laid on a chain surface with long parallel links, interrupted occasionally by a perpendicular gap. This pattern could be seen in the final paper. In the eighteenth century, *wove paper* was developed that was smoother for better printing. A woven wire mesh transports the pulp, and the grain of the paper is no longer prevalent. Today, wove paper is considered a cheaper paper. *Watermarks*, also known as papermarks, are intentional patterns pressed into the grain. They started appearing in paper in the thirteenth century to indicate origin.

Waste paper can be recycled and used instead of raw tree wood as the source of pulp. The paper must be shredded and then returned to a pulp state. Along the way the ink must be bleached out of it. As the paper is returned to pulp, the length of the fibers is reduced. This lowers the quality of the paper produced from this pulp. It is therefore usually mixed with virgin pulp.

## Finishing Procedures

The surface characteristics of the paper affect the visual characteristics of the writing trace. The ink can either sit on top of the paper fibers or be absorbed into them. Europeans were accustomed to using quill pens on parchment and needed a strong, scratch-resistant, non-absorbing paper. *Sizing* adds gelatin, rosin, starches, gums, or alum to the paper to make it harder and less absorbent and thus resistant to the water in water-based writing inks. Sizing can be done after the formation of sheets through *tub sizing* by placing the paper in a bath of gelatin essentially coating it with a thin layer of glue. Alternatively, with *engine sizing* the pulp is mixed with sizing materials during processing before the sheets are formed. Tub-sized paper is higher quality than engine sizing because the sizing material is located where it is most effective, but it is also more expensive. Sizing made paper durable enough that both sides of the paper could be used for printing.

There are several paper *finishes*. They are often *coatings* of pigments or vehicles (binders) such as calcium carbonate or china clay. Coatings can produce a *matte* (dull or muted), *semimatte*, or *glossy* finish. Paper was originally brush coated with a clay substance to produce a surface suitable for fine-screened halftones for use in the finest quality photographic reproduction. A *machine finish* will produce a smoother surface and is often used for magazines. *Coated* paper is usually white and in text weight. *Gloss* will lead to less dot gain when printing as the ink will not spread as much. Uncoated paper is found in white and colored versions. *Art paper* is paper glazed with a china clay coating then rolled to make it very smooth to better print halftones/screens for illustrated documents. However, the china clay coating reacts with the acids in the paper and makes the paper brittle so folds quickly become cracks. Coloring was first added to paper in 1687, and machine ruling lines first appeared in 1770.

*Calendering* is a finishing operation that passes the paper through a series of steel rolls to impart a glossy finish or increase the surface smoothness or opacity. Minimum calendering produces an eggshell or antique paper, which has a rougher texture and is very “non-glare,” which can increase readability. *Supercalendered* paper is given a smooth shiny finish by repeated rolling of the paper between hot and cold rollers. Machine finish papers have fairly extensive calendering and are used for magazines, because the finish enables the printing to reproduce very fine halftones.

## Paper Classifications, Uses, and Quality

Paper production materials and processes influence paper quality. The paper options influence their use, characteristics, and quality. There are three factors to consider when buying paper today: *grade*, *whiteness*, and *opacity*. A higher grade paper has a more refined smoothness, whiteness, and greater opacity than other papers. In addition, there are four basic paper classifications to consider: *bond*, *book*, *cover*, and *cardstock*. Bond paper – lower grade paper that is used in most offices for printing and photocopying – has a semihard finish, and book paper comes in a range of textures. Rough papers will likely have ink dropouts where ink never reached the

**Table 2.1** Paper characteristics, uses, and quality

Classification	Use	Grade	Whiteness	Opacity	Thickness	Finish
Bond	Stationery, office copies, manuals	Writing, copy bond, digital, virgin pulp or recycled	Average commodity whiteness	More with thickness	Varies	Smooth or textured
Book	Text pages of books and booklets	Varies by whiteness, opacity, thickness, and finish properties	Varies with grade	Made with or without opacity properties	Varies	Smooth, vellum, coated, or uncoated
Cover	Pamphlets, book covers	Same as book/text	Same as book/text	More than book/text due to thickness	Medium to heavy	Same as book/text
Cardstock	Postcards, business cards, misc.	Varies	Varies	More due to thickness	Medium to heavy	Smooth or vellum

paper during the initial printing process. Ink spreads according to the porosity. Filler material, such as white chalks, clays, and titanium dioxide, is often added to the pulp to give it better opacity and surface finish. Cover and cardstock are not often used for producing documents (Table 2.1).

Paper is graded by thickness. In North America and Great Britain, this is indicated by measuring the weight of a ream of paper cut to the basis size for that grade of paper. A ream has 500 pages, although in times past a ream had 480–520 sheets. The basis size for bond paper is 17 × 22 in. and for book paper is 25 × 38 in.; thus, 20 lb bond paper is equivalent in thickness to 50 lb book paper. In Europe, paper grading is much simpler and uses the weight in grams per square centimeter ( $\text{g}/\text{m}^2$ ), sometimes abbreviated as gsm. 20 lb bond paper is equivalent to 75.2 gsm paper. Paper thickness contributes to the likelihood of printed matter on the verso (back) side being visible on the recto (front) side. Calendering makes more dense paper. The choice of fillers also contributes. India paper is a very thin paper that is also opaque.

## Inks

Inks can be grouped into two categories, those used with handheld writing instruments and those used by mechanical printing processes. Inks are all made from *colorants* (pigments and dyes), *vehicles* (binders), *additives*, and *carrier substances* (solvents). The desired flow property depends on the printing or writing process that will be used as the ink has to match the transfer mechanism and drying or fixing process. Inks range from thin and watery to viscous and also exist in powders or solids. Ink must run freely but not spread. It must dry easily and not harm the paper or writing instrument.

Egyptians around 3000 BC used black ink made from carbon and red ink made from natural iron oxide suspended in water with gum or glue. Pictures of scribes and the hieroglyph for scribe always include a rectangle with two circles within it, , representing the wells for these two ink colors. At about the same time, the Chinese developed a similar black ink made from lamp or carbon black suspended in a dilute solution of water soluble gums. This kind of ink is called “India ink” as it was introduced to the west through India. This ink requires frequent stirring so the carbon remains in suspension. The carbon pigment remained on the paper surface instead of soaking into the paper. This ink is stable and shows minimal effects of age, but is water soluble.

Iron gall ink was invented in the fifth century AD and became the prominent writing material from the Middle Ages through the twentieth century. It was made from a mix of iron salts (often vitriol or iron(II) sulfate), tannin (an acid from oak galls from which gallotannins are extracted), and glue (gum Arabic, a vegetable gum from the acacia tree) to bind. Over time, the iron-tannin components would oxidize turning the ink black giving it the name “blue-black ink.” Eventually, this ink fades to a dull brown color. In the nineteenth century, indigo dye was first added to inks to produce a less acidic blue ink.

Colorants used in inks can be organic or inorganic pigments in soluble oil. Pigments have particle sizes  $0.1\text{--}2 \mu\text{m}$  and are held in suspension. They need a vehicle for binding them to the paper. Vehicles can also coat the pigments and protect against mechanical wear (abrasion) and are sometimes called varnishes. Pigments have a wide color absorption band. Dyes during application have higher color intensity, produce more luminous colors, and come in a wider range of colors. Dyes are organic compounds that are dissolved. Natural dyes were initially used for color but were replaced by aniline and synthetic dyes around 1900. Synthetic dyes are used almost exclusively today. Dyes can be transparent, and the particles are smaller than in pigment, but they are less light fast than pigments. Most printing methods use pigment, but inkjet printers predominantly use dyes.

Binders are usually resins dissolved in mineral oil. Additives depend on the printing process and influence drying time, flow behavior, and abrasion resistance. Carrier substances are thinning agents like mineral oil or solvents like toluene.

In the 1940s, the ball-point pen was commercially introduced which uses a viscous quick drying paste like ink. Ball-point pen ink colors come from synthetic dye and include methyl violet, Victoria blue, and luxol fast orange; nigrosine; copper phthalocyanine; and other organometallic dyes. Dyes and pigments compose about 25 % of the mass of a typical ball-point ink. The solvents or vehicles are made from a mixture of glycols such as ethylene glycol. Prior to 1950, oils such as linseed or mineral oil were used. The vehicle dissolves or suspends the dyes or pigments and promotes the smooth flow of the ink over the surface of the rotating ball. The vehicle dries quickly usually through evaporation leaving the color on the paper. Solvents make up 50 % of the mass of the ink. The remaining 25 % of the ink is resins which can be naturally occurring or synthetic materials and provide a viscosity to the ink.

In the 1970s and 1980s, felt-tip and roller writer pens were introduced which use a liquid ink that transfers through the tip and soaks the paper evenly. Fluid ink will

penetrate the paper fibers more than viscous inks. Gel pen inks introduced in the late 1980s are viscous, but not to the degree of ball-point ink. The gel is water based with biopolymers, such as xanthan and tragacanth gum, as well as polyacrylate thickeners. Gel ink contains pigment suspended in a viscous medium, so it has a thicker deposit of pigment making a bolder line. The pigments are opaque and come in a variety of bold colors. The pigments are typically iron oxides and copper phthalocyanine.

In addition to liquid and viscous inks, inks can also be solids. The Romans used lead rods for marking. When a large source of graphite was discovered in England in 1564, it was not understood that it was not a variety of lead, and the name remains today. *Pencil “lead”* consists of waxes, fillers (clay), powdered graphite, and water blended and extruded into rods which are dried and kiln fired. The result is porous and can be impregnated with wax to make the writing smoother. Colored pencils use colored pigments with clay and wax or fatty acid combined with water and an absorbent material like gum tragacanth. These are dried and the firing stage is omitted. Pencils come in several levels of hardness which come from varying the ratio of clay and graphite. In Europe, these range from 9H to H, F, HB, then 1B to 9B. H is a hard lead which deposits very little carbon on the paper making the mark very light, and B is a softer lead which writes very black. In North America, lead hardness is primarily indicated by numbers 1–4, with 1 corresponding to the European 1B, the most common hardness; 2 corresponding to HB; 3 corresponding to H; and 4 corresponding to 2H.

### Inks for Machine Printing

Printer ink is very different from pen ink. Ink characteristics are intertwined with machine printing technologies. This section focuses on the inks, with more details about the machine technologies identified in the section “[Machine Printing](#).<sup>19</sup>” Letterpress inks are viscous, almost like paint. Historically, it is sometimes called black “treacle” as it was made from linseed oil boiled down until it attained a glue-like consistency after it was freed from the fats it contained when raw. The coloring came from lamp black particles that were ground and reground until they were very fine and would not clog the counters of the smallest letters. Modern inks are made from a combination of solvents and plastics. These inks dry through absorption into the paper. Offset printing is a commonly used printing technology that transfers (or “offsets”) an inked image from plate to rubber blanket and then to paper. It also uses an ink that is a highly viscous pasty ink. It is made of hard resin, sometimes alkyd resin; vegetable oil (linseed, soy, wood); or mineral oil and pigment. Gravure printing ink has a lower viscosity making it a liquid ink so it can fill engraved cells. Common solvents are toluene, xylene or petroleum spirits, ethanol, ethyl acetate, or water (sometimes mixed with alcohol).

The ink used for typewriters is contained on a ribbon. The ribbons are sometimes textile ribbons, and the weave of the ribbon is often visible in the character image, Fig. 2.4b. Later development led to the production of a tape with a removable film of black ink that would transfer to the paper when pressure was applied. This tape was less susceptible to drying than the inked ribbons. Since it transferred a more

uniform coating of ink to the paper, it produced dark areas more uniformly than ribbons. It also kept the typeface from getting gummed up as the type only came in contact with the non-inked back of the tape (Fig. 2.5).

The toner used in *xerography*, as in laser printers or copy machines, is not restricted to a liquid ink and is most often a carbon-based powder mixture. The particles usually include magnetic carrier particles, often iron oxide, and a polymer that melts to affix the toner to the paper. The carrier is recycled and can be 80  $\mu\text{m}$  while the toner is 4–8  $\mu\text{m}$ . Toner without a carrier has particles 12–20  $\mu\text{m}$  in diameter. Liquid toner for xerography will contain particles of 1–2  $\mu\text{m}$  and allow colors to be directly mixed by mixing toner during the printing process.

Inkjet printers require a low viscosity ink that must be filtered so pigment agglomerates don't block channels in the nozzle of the print heads. Inkjet inks are usually water based. They tend to bleed or penetrate the substrate surface and can cause the substrate to warp or wave. Thus, specially coated paper is recommended for use with this printing method. Some inkjet papers melt a wax or plastic ink that remains on the surface of the paper.

---

## Writing and Printing Processes

The ink can be transferred to paper via a handheld device or a larger machine. This section describes the technologies in both categories. The inks used in these writing and printing methods were described in the section “[Inks](#).”

## HandHeld Writing Instruments

Before the advent of machine printing, all writing was done with a handheld writing instrument. Handwritten and hand-printed documents are all created by sliding a writing instrument across the writing substrate. There are many types of handheld writing instruments such as brushes, nib pens, ball-point pens, felt-tip pens, and pencils. The appearance of the stroke that results is determined by the shape of the writing tip, including how it deforms when in contact with the writing substrate, and the characteristics of the ink such as fluid type and opacity.

The Greeks used metal styli to mark on wax tablets. The Sumerians used reeds to imprint on clay tablets. In northern India people used a reed pen which led to the development of angular script forms, whereas in southern India, people used a metal stylus, and a more rounded script form evolved to not tear the paper. In Egypt, the stylus used for writing was a reed, whose end was chewed to make a kind of brush, so writing resembled painting, but the core of the reed held ink. In 1000 BC, the Chinese used a camel hair or rat hair brush. The medieval European scribes used a small brush for fine work called a *pencillus* (“little tail”) which led to the word pencil. Brushes are likely to have a variable stroke width and may have streaking in the stroke.



**Fig. 2.2** Samples of metal nibs and writing

*Quill pens* were introduced to Europe in the sixth century. The word pen comes from the Latin word penna which means feather. A nib, or tip, was cut into a feather from a large bird, usually a goose. This was then dipped in ink, usually water based, to form a reservoir of ink in the hollow shaft. Pressure between the nib and the paper caused ink to be transferred via capillary action to the paper. Through use, the point on a quill pen would wear down requiring the feather to be cut again. This could be done by the writer, or pen cutters often “stationed” themselves on streets offering their services and lending the word stationary to office supplies. In the 1800s, metal inserts (Fig. 2.2) were developed to remove the constant need to recut the nib. Early *metal nibs* had problems with lack of flexibility and corrosion, especially with the use of iron gall ink. When writing with a metal nib, the points of the tip often separate under pressure on the downstroke creating furrows in the paper that fill with extra ink called “*nib tracks*.” The shape of the nib influenced the writing styles and vice versa. The broad nib has a flat edge, and the thickness of the stroke depends on the angle of the stroke relative to the pen. Pointed nibs vary the stroke width by exerting variable levels of pressure to separate the tines different amounts. *Fountain pens* are nib pens that have an internal reservoir of ink. The first successful fountain pen was developed in 1884. This removed the gradual fading seen in writing as the reservoir of dip pens emptied.

*Ball-point pens* are the most common of all writing instruments today (Fig. 2.3a). The first patent for a ball-point pen was issued to an American named John Loud in 1888. He designed them to be able to write on rough surfaces. Improvements in ball grinding and measuring techniques enabled the pens to be constructed well.



**Fig. 2.3** Examples of writing tips and writing samples for (a) ball point, (b) gel (c) felt tip and (d) pencil

The Biro pen was introduced in England in 1944 and was widely adopted in the 1950s. Precision ground ball tips are 0.7–1.0 mm in diameter and fit in a socket with narrow ducts to channel ink from the reservoir. The ink is thick and gelatinous and is deposited by capillary action to the ball, thus requiring more pressure than when writing with a nib pen. Ball-point pens write best on rougher surfaces where friction can turn the ball and draw more ink. As the tip is used for propulsion, more ink will be on the edges of the stroke than in the center. *Rollerball pens* are similar to ball-point pens except the ink is more fluid. This ink soaks into the paper more than ball-point ink and produces a different stroke texture. Rollerball pens also require less pressure to write. *Gel pens* are ball-point or rollerball pens with a gelatinous ink (Fig. 2.3b).

*Felt-tip pens* or markers were introduced in Japan in 1964 (Fig. 2.3c). The tips are either hard felt or fiber bundles bound by resin. Felt-tip pens use a more liquid ink and thus are likely to have a less opaque stroke. In addition to black and dark blue, they contained brightly colored fluid inks. This type of pen is also used for highlighters which use an ink that intentionally produces a nonopaque mark. Permanent or indelible markers use a dye that is not easy to remove from a surface.

*Pencils* write by using the roughness of the writing substrate to grind off a thin layer of pencil “lead” or graphite which remains on the substrate as a mark (Fig. 2.3d). Variations in appearance result from the hardness of the lead and the pressure applied. Traditional wood pencils encase the lead in wood. In the first quarter of the nineteenth century, mechanical pencils which feed the lead through

a metal or plastic case became common. The common lead diameters are 0.5, 0.7, and 1.0 mm. Mechanical pencils have finer lead than wooden pencils. Wood pencils rely on a sharpening device to remove the surrounding wood and shape the tip to a finer point, usually smaller than the diameter of the fill lead. The degree of sharpness affects the writing appearance. This changes through use over time, and thus, a more consistent response is found with mechanical pencils.

The handheld writing instruments described so far transfer ink to a substrate like paper. The path the instrument takes is thus recorded and provides the image which is either directly viewed or ultimately digitized. *ePens* have been developed that allow this same pen trace to be recorded on a computer, either instead of marking the paper, or while marking the paper. They are often connected to displays where the trace is shown. To identify their position, some ePens are designed to work with special paper. The Anoto® paper has a dot pattern that varies across the paper. The pen, usually a ball-point pen, writes like a traditional pen but has an optical sensor built into it to see the dots and determine where the pen tip is hitting the paper. This information is delivered to a computer which decodes it to record the pen trace path. Other ePens have accelerometers to determine the motion directly which, when integrated, become positions. Alternatively, they rely on an electronic pad that detects the position of the pen. The ePens are the input device for online handwriting recognition and are discussed further in ►Chap. 26 (Online Handwriting Recognition) and ►Chap. 23 (Analysis of Documents Born Digital).

## Machine Printing

While ultimately the human hand participates in all printing processes, those where the hand is not directly involved are considered machine printing. Machine printing can be divided into two major subcategories, *impact* and *nonimpact*. Impact printing is the older of the technologies, while nonimpact printing has been enabled by the introduction of electronics. As the variety of new printing methods has expanded, all printing methods that do not require a printing plate master for the image or character have been categorized as nonimpact.

### Impact Printing

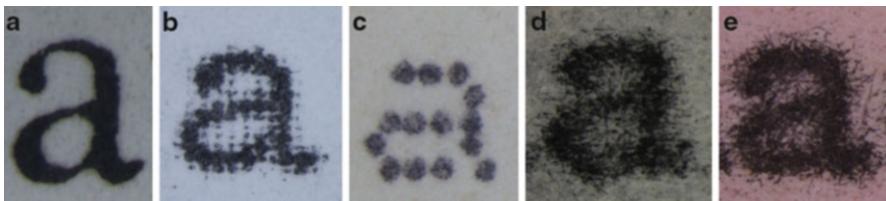
Impact printing uses four major methods to get ink onto paper. The documents produced by each of these methods have different image characteristics. Each has a preferred paper and ink. The combination of the ease of use and historical context of each has influenced how prevalent each type has been and how frequently they are the subject of document image analysis. Table 2.2 summarizes the methods and types of impact printing.

### Relief Printing

One of the oldest forms of machine printing came from the text or the images being carved into wood blocks, which were then inked and pressed onto the paper in a process called *xylography*. The first samples of this type of printing date to the eighth

**Table 2.2** Impact printing methods and types. Black represents the printing device and gray is the ink

#	Method	Type(s)
1	Pressure against a relief (raised) surface	Letterpress, typewriter
2	Chemical action from a planographic (flat) surface	Lithography, photo lithography, offset printing, ditto machine, hectograph
3	Lifting ink from an intaglio (engraved or depressed) area	Gravure, rotogravure
4	Seepage through a stencil or screen	Silkscreen, mimeograph



**Fig. 2.4** Relief print samples: (a) letterpress, (b) typewriter, (c) dot matrix, (d) carbon copy, and (e) carbonless copy

century AD in China. The first printed book was the “Diamond Sutra” printed in China in 868 AD, first in scroll form, then as books. A collection of 130 volumes of classics were printed in 953 AD. This printing method became common in Europe in the fifteenth century. A full page of print was carved into each block. The production of the wood block was labor intensive, so the quantity of material printed with this method was small, but the blocks could be used for an extended time to produce many copies of each printed page.

The time needed to produce a page of type can be reduced by producing smaller blocks for each character or letter and then rearranging them for each page. This is called *letterpress* (Fig. 2.4a). *Movable type* was first invented in China by Pi-Sheng around the year 1040 AD. It used a ceramic font type. This was found to be brittle, and the font needed for the Chinese script was too large. Due to the value placed on Chinese calligraphy, it was not a viable invention. In the year 1234, a movable-type printer using a bronze font was invented in Korea and was used in parallel with xylography. The metal font allowed finer type detail than xylography. However, the bronze type produced sharp edges, and the preferred paper in Korea was soft. Koreans alleviated the problems by placing the paper on the type and rubbing it to contact the type and transfer the ink. From 1436, they also experimented with type made from softer lead. While the script in use in Korea at that time was also Chinese, the political will to disseminate large quantities of printed material made movable type a viable instrument such that a second font set was created in 1403 and a third in 1420. Ultimately, movable-type printing systems did not retain long-term viability due to changes in political leadership.

Although not its first inventor, the invention of movable type is usually credited to Johannes Gutenberg. In Mainz, Germany, in 1440, he created a movable-type printing press. As the Latin character set is relatively small, it is better suited than Chinese to this printing method and was adopted by others in the west. Printing reached Cologne, Basel, Rome, Venice, Paris, Nuremberg, and Utrecht by 1470 and England, Spain, and Portugal by 1480.

Text was laid out by a typesetter or compositor who would assemble individual pieces of type into a form held together by a frame or chase. Multiple instances of each letter are needed for each page. Care was needed to assure the typefaces would be planar and each piece would have equal pressure against the paper across the whole surface. The filled form was placed on the bed of the printing press and inked. A damp piece of paper was placed on top of it. The metal plate (platen) was lowered to bring the paper type in contact with the paper. To resist the scratching of quill pens, European paper was tougher than Korean paper. As a result, instead of rubbing the paper on the type, the printer used the pressure of olive or wine presses to complete the ink transfer. A tin and lead alloy was used for the type. This makes a firm enough material to press onto the paper to make the print, but not so hard as to cut the paper nor too soft where the type would deform after a limited amount of use. Still type was prone to being dented, and small variations in characters can often be seen in letterpress documents.

Creation of the type started by cutting a metal punch to produce a raised, reversed image of the letter carved onto the end of a hard metal bar. This was driven into a softer metal to form an indented, reversed version of the letter called the matrix. The matrix was placed into a mold over which molten metal was poured to create individual pieces of type, again raised and reversed. Many pieces could be made from one matrix. With the use of molds, the shapes of the characters in a given document became repeatable. This is beneficial to today's optical character recognition technology in that the pattern to be matched can be better predicted. Handwritten writing styles existed before the printing press, such as Uncial and Carolingian. Initially, the characters mimicked these writing styles, but over time, a greater variety of fonts were developed (see section in ►Chap. 9 (Language, Script and Font Recognition)).

While movable type reduced the time spent carving blocks for each page, a significant amount of time was still needed to select individual-type pieces, place them in the frame, and then return them to the case after use. This process was partially automated with the introduction of the *Linotype* composing machine in 1886. The magazine did not hold type but instead matrices or molds to cast typefaces. Based on operator input, a whole line of type would be cast at a time which would then be set into the frame or chase. In 1897, a patent was issued for *Monotype* which also cast type upon demand, but cast individual-type pieces. For both methods, after printing the type would go back into the melting pot for reuse. These advances allowed for easier production, which increased production volume. Linotype was used primarily in newspaper and magazine production and Monotype mostly in book production. While these systems changed the composing method, they didn't change the printing method; however, every letter used in printing was new and sharp, thus increasing print quality.

- a The quick brown fox slyly jumped over the lazy dog.
- b The quick brown fox slyly jumped over the lazy dog.
- c The quick brown fox slyly jumped over the lazy dog.

**Fig. 2.5** Typewriter samples: (a) manual typewriter, (b) electric typewriter, and (c) electric typewriter with ink tape instead of ribbon

The printing process was accelerated by the introduction of the *rotary press* which used cylindrical plates. With cylindrical plates, the step of lifting the plate was removed from the process. Rotary presses could roll the cylindrical plate across a flat surface, or two cylinders, one with type and one blank, could oppose each other. Individual sheets could be fed, or paper could be fed from a roll or web, into the rotary press, further accelerating the process. The cylindrical plates could not be made by placing discrete pieces of movable type into a form. Instead, methods of *stereotype* and *electrotype* were used to form the plate. With stereotyping, a duplicate was made from a flat plate of type by pressing a papier-mâché or fiberboard matte to the plate to make a mold from which molten lead could be cast. That mold could be bent allowing the production of cylindrical in addition to flat plate duplicates. For electrotype, an impression of a plate was made in a wax-coated foil or sheet of thermal plastic material. The mold was polished with graphite or sprayed with metallic silver to make it conduct electricity. It was then immersed in a solution of copper sulfate and sulfuric acid, and a layer of copper was deposited by electrolysis to a thickness of 0.015 in. or 0.38 mm. The copper is separated from the mold and backed with a lead-based alloy and mounted. Stereotype was used most often in the production of newspapers. Electrotype was used mostly for magazines.

*Typewriters* were initially developed in 1714 to enable the blind to read by embossing paper with letters. In their established form, typewriters produce printing for sighted people, Figs. 2.4b and 2.5. They use a premise similar to letterpress of inking a raised piece of type and pressing it to the paper, but the characters are pressed to the paper individually instead of a whole page at once. The units containing the typeface are located on the ends of rods. When a key is pressed, the rods rise and strike a ribbon impregnated with ink against the paper. The first American typewriter was patented in 1868 and contained 44 keys for the 26 uppercase letters, the numerals 0–9, and some punctuation. Typewriters became common in commercial offices in the 1870s. The carriage that holds the paper is moved horizontally after each character is struck. Initially, the carriage progressed a uniform amount after each keystroke, resulting in a uniform spaced type. After the user presses a carriage return key, the carriage will rotate vertically and return to the left margin to ready the machine for the next line.

A shift bar was added to allow two character symbols, usually an uppercase and a lowercase form of a letter, to occupy each type unit. This increased the character set to 88 symbols. In 1961, a model was introduced that replaced the type on long rods with a ball of type that would rotate to select the character symbol. Another variation used a disk of type. These balls and disks were easy to remove, enabling users to change fonts. Typewriters are used primarily with the Latin character set, but are also found for the Greek and Cyrillic characters. A typewriter that was capable of printing Japanese kanji-kana text was developed in 1915. Due to the larger symbol set, it was a much larger and more complicated machine and was not widely adopted.

The typeface is a flat surface which is pressed on a cylindrical surface. The cylinder is soft, so within a small area, the full shape will reach the paper. However, maintenance of the units varied, and when the calibration is extremely poor, or the machine is operating at a high speed, the type may not fully hit the paper, resulting in partial characters and a nonuniform baseline of the text (Fig. 2.5a). Vertical misalignment of the characters could result from poor calibration of the keys or from the key being pressed when the type tray was still in transit from one position to the other. Early typewriters were fully manually powered so different amounts of pressure would be imparted on each stroke. The introduction of electric typewriters resulted in a constant amount of pressure on each character giving greater uniformity to the resulting image (Fig. 2.5b, c). Also, the loops in the characters were prone to filling either because the typeface was gummed up with ink from contact with the ink ribbon or because the type pressed down the ribbon far enough that not only the ribbon in front of the type came in contact with the paper, but from being stretched the ribbon within a loop and near the border also made contact (Fig. 2.4b).

Corrections could be made by painting the paper with an opaque white paste correction fluid and retying on top of the corrected area. A surface that was not smooth would result. Alternatively, a slip of tape or paper with a film of white “ink” could be inserted between the ribbon, and the paper and the character could be retyped covering the ink in the character “zone” with white, then a different character could be typed on top of this corrected area. Impressions of both characters would result on the paper, even though only one would be optically visible on the image surface.

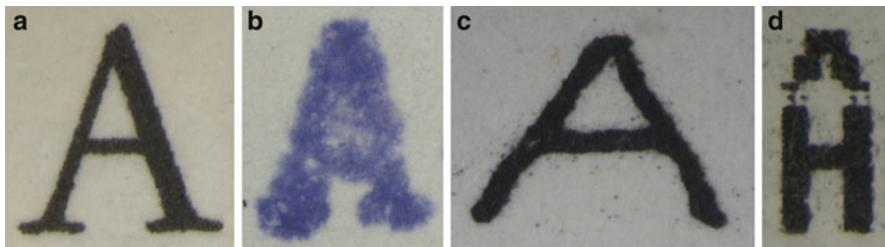
Later developments allowed the input to the typewriter to be remotely located, making typewriters early output devices for computers. *Dot matrix printers*, invented in 1964, were created to be output devices for computers. In functionality, they are a variation of a typewriter. The information is transferred to the paper via a rectangular grid of pin-shaped type that impacts an ink ribbon. A digital control signal from a computer determines which of the pins are activated. Because of the addressability, the shape of the character can be changed through software, and a greater variety of character shapes are possible than with a traditional typewriter. Initially, the printers had a grid of  $5 \times 7$  pins (Fig. 2.4c). This grew to 24 rows of pins. The dot density ranged from 60 to 360 dpi (dots per inch).

*Carbon paper* was a common way of producing two to four copies of a document. Pieces of paper coated on one side with a thin film of carbon black, initially bound to the paper by a thin film of oil, would be placed between pieces of blank paper. When the paper was pressed with high pressure, such as from a typewriter or pen, the carbon would transfer to the blank paper beneath it (Fig. 2.4d). Carbon paper was made at least as early as 1823 but was only made common with the introduction of typewriters because it did not work well with nib pens. At one time, the technology was so prevalent that the labels “cc” and “bcc” (for carbon copy and blind carbon copy) still appear in many email applications, where there is no carbon in sight. Because the carbon was prone to transferring to surfaces not intended, a similar process called *carbonless copies* could be made by using autocopy or NCR (no carbon required) paper, a set of papers containing encapsulated dye or ink on one piece and a reactive clay on the next on the facing side (Fig. 2.4e). When the two came in contact with enough pressure, such as that from a pen or typewriter, the microcapsules would break and spill their dye. Both these techniques produce lighter copies with broader strokes at layers further down the stack. The width of the stroke in the copies can be wider causing touching and filled characters.

### **Planographic Printing**

*Lithography* is a planar process where the printing surface is neither raised nor recessed. Lithographic printing started with the application of an oily substance on a stone printing plate, from which the process received its name (litho, stone; graphy, writing). The stone is ground to a smooth even finish. Grease is applied where color or ink is desired in the final image. An artisan can draw directly on the stone with a grease pencil. During printing, the stone passes beneath a dampening roller to apply water to the nongreasy parts, then beneath inking rollers where greasy ink is attracted to the grease of the template but repelled from the water. The stone is then pressed to a piece of paper where the ink is transferred to the paper. Zinc or aluminum plates can be substituted for the stone. These plates are lighter and take less space to store. They can also be bent into cylinders allowing use in a rotary press configuration, which accelerates the printing process.

In *photolithography*, a photographic negative of the image to be printed is made. The negative can be made from any source or collection of sources of images, including handwritten sources, or printed samples from any printing process. These can be composed as a single unit or by placing several different pieces from different sources together to form a single page of print. This process is useful to reprint books that are out of print, since the publisher would not need to repeat the time or expense of typesetting. The plate is sensitized with a bichromated emulsion which is hardened when exposed to light usually from passing light through a film negative. The design appears only where light passed through the film. The plate is rolled with a greasy ink and then washed to remove the unexposed emulsion and its ink layer, also filling in the non-inked areas with water. Printing mirrors the approach used in lithography, dampening and inking the plate at each revolution. The very thin film of ink that is used requires a very smooth paper. It can also lead to low depth of tone in screened images. An alternative deep-etched photolithography method exists that



**Fig. 2.6** Planographic printing methods (a) Offset lithography, (b) ditto machine, (c) hectograph, and (d) thermal printing

produces etching deeper than with regular lithography. Because this is a chemical rather than mechanical process, it is referred to as “cold type,” because no molten lead is involved.

In direct lithography, because the plate comes in direct contact with the paper, the image on the plate must be reversed from the desired final document image. *Offset printing* or indirect lithographic printing works on a similar premise as in direct lithography printing, but the ink is first transferred from the metal plate to a rubber plate and then transferred from the rubber plate to the paper (Fig. 2.6a). The rubber surface squeezes the image into the paper rather than placing it on the surface of the paper. This offers the advantage of enabling the use of cheaper papers since the rubber can press more firmly to the paper, opening up possibilities beyond using paper with more costly glossy or polished finishes. The extra transfer step can, however, introduce minor distortion to the character shapes, especially on interior and exterior corners. Conversion from letterpress to offset in larger print houses occurred in the 1970s.

Another planographic printing technique, *spirit duplicators* or *ditto machines*, was used often for small batch printing. Invented in 1923, a two-ply master was used that included one layer that would be written or typed on and the second which was coated with a layer of colorant impregnated wax. When the top sheet was written on, the wax was transferred to the back of that sheet. In the printing process, a thin coating of solvent (typically ammonia or methylated spirits, an even mixture of isopropanol and methanol) was applied to the master sheet, and a thin layer of wax would then transfer to the blank paper when contact was made. The wax, and thus the resulting print, was usually a purple color (Fig. 2.6b). The machines were limited to 500 copies for a single master.

The *hectograph* machine, also known as the gelatin duplicator or jelly graph, was invented in the late 1870s. Ink containing an aniline dye was used to write on a sheet of paper. The document image was transferred to a shallow tub of gelatin mixed with glycerin by placing the inked side of the paper onto the bed of jelly. A clean piece of paper would be pressed onto this gelatin, and some of the ink would transfer to the paper (Fig. 2.6c). Twenty to eighty copies could be made from each master, with later copies being lighter.

*Thermal printers* use special paper that when heated, turns black. Invented in the early 1970s, the printers function similar to dot matrix printers, except the grid of dots are heated and placed in contact with the paper and are not forcefully pressed into the paper. The heat usually results from an applied electric current. The paper is impregnated with a mixture of a dye and a matrix, so no external ink is used. When the matrix is heated above its melting point, the dye reacts with the acid and changes to a colored form. Thermal printers are faster than dot matrix printers and are usually smaller and consume less power. Thermal printing was commonly used in fax machines and is still frequently used in retail receipts (Fig. 2.6d).

### **Gravure/Intaglio Printing**

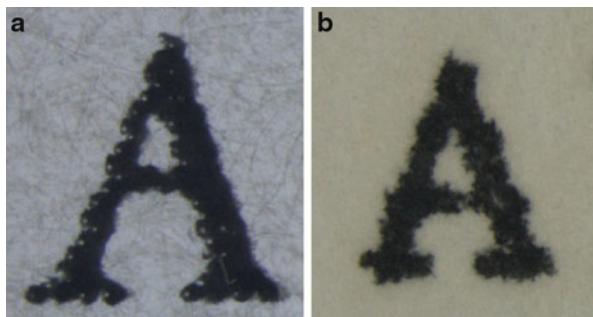
In *gravure*, also known as *intaglio* printing, the printing surface is recessed, and ink that fills the recessed areas is transferred to the paper. A plate is etched, or cut; coated with ink; and then wiped clean. When the paper is pressed against the plate, the ink in the wells is transferred to the paper. The amount of ink can be controlled by varying the depth of the engraving, the width of the engraving, or both. Similar to in offset printing, plates were etched with acid that eroded the metal in the desired printing locations to depths proportional to the hardness of the gelatin. *Photogravure* results when a photographic process makes the engraving. Photogravure has a higher initial cost for plates than photo-offset printing but produces higher quality documents over a longer run batch and was commonly used for high distribution magazines containing high-quality images. When the printing surface is on a cylindrical form, it is called *rotogravure*.

When a photograph or other continuous tone image will be included, the entire drum is pre-etched with a screen pattern (see section “[Multitone and Color Printing Dithering and Screens](#)”). Then, the desired text and images are etched on top of this. The variable size of the ink wells can produce a greater variety of tones achieving very high image quality. However, there is a tradeoff in quality in that a serrated or jagged edge typically appears on lines and text (Fig. 2.7a). Text also loses sharpness because of the watery consistency of the ink.

### **Silkscreen Printing**

*Silkscreen* is a printing method that fits into the fourth impact printing method of applying ink to substrate. Here, ink is passed through a stencil. The stencil has a stabilization mesh made of either silk or sometimes a metal screen to support the gutters. The screen is stretched across a frame producing a printing form. Material is applied to portions of the screen to block the pores. This can be done by cutting a pattern from a lacquer backed sheet to remove the backing and placing that on the screen. A screen can be coated with an ultraviolet cured varnish that is marked with a photomechanical stencil design that is washed away exposing the screen. Photographic development or lithographic crayons can also be used to create the stencil. A squeegee is passed over the screen to force the paint or ink through

**Fig. 2.7** Printing methods  
(a) Gravure and (b)  
mimeograph



the porous portions of the screen. The paint or ink has the consistency of syrup. Silkscreening is used primarily for posters, displays, and fine art reproductions, as well as on diverse surfaces such as bottles, wall paper, and fabric.

In offices, the *mimeograph* or *stencil duplicating machine* was used to print documents that needed more copies than a typewriter with carbon paper could produce, but not enough that turning to other techniques was reasonable (Fig. 2.7b). The typical capacity for one stencil was 200 copies. This machine, invented in 1890, was based on Thomas Edison's patent for autographic printing that is used to make the stencil. The stencil was usually made on a typewriter with the ink ribbon removed, so the type would directly impact the stencil and displace the wax. Ink is pressed through the stencil master and then onto the paper. Over the course of the run, the interiors of closed loops would often fall out, assuring that the loops of those characters would be filled with ink in subsequent copies. This technique produces documents with a higher quality than the ditto machine that was common at the same time for small batch printing.

### Nonimpact Printing

In the twentieth century, methods of printing that did not involve physically pressing a template onto a substrate were developed. These were made possible by the development of computers and electronics able to move the ink relative to the paper. The development of these technologies also increased the flexibility of the printing system, removing the need to create masters for everything that would be printed. The term nonimpact came when computer-controlled printing methods of dot matrix and electrophotographic (EP) printing were compared. Dot matrix printing impacted a pin to the paper and EP printing did not, even though in EP printing contact is made between a roller and the paper. Nonimpact printing came to refer to all printing methods that were masterless and does not always refer to printing processes that do not involve any contact with the paper. An advantage of nonimpact technologies is their ability to customize documents between rotations. Since these technologies don't have a fixed master, they can produce a greater variability within a print run even if a constant image is desired.

### Electrophotographic Printing

*Electrostatic* printing or *xerography* (from the Greek “dry writing”) was first publicly demonstrated in 1948 as an image copying process. The process starts by charging the photoconductor (PC) plate. Light discharges part of the PC. Oppositely charged toner particles are attracted to the remaining charged areas. Paper is charged, and as it comes in contact with the PC, the toner particles are transferred from the PC to the paper. Application of heat and pressure affixes the toner to the paper. Excess toner particles are removed from the PC, and the PC is neutralized for the next cycle.

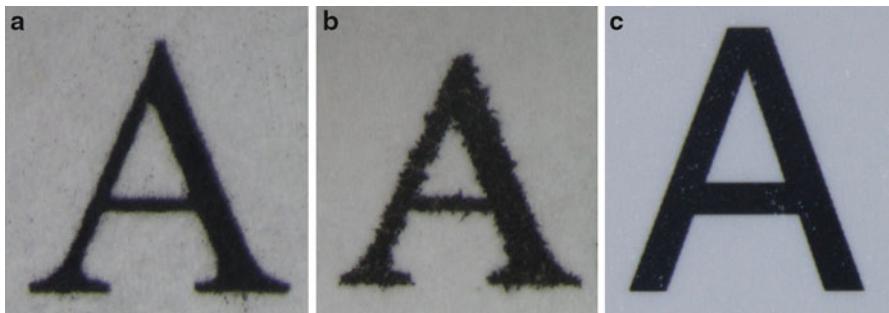
Computer-generated images are transferred to the PC by a row of LEDs (light-emitting diodes) or by a rotating mirror reflecting a laser beam onto the PC. In this case, the light source “paints” the PC with charge that attracts the toner particles. The use of a laser gave rise to the name *laser printer*, even though a laser is not always the light source involved. The print quality depends on the resolution or addressability of the imaging system. The basic procedure has the PC discharging one row at a time. The row is divided into addressable units which, in theory, are either on or off. This has the tendency to make diagonal lines “jagged.” When a laser is used, varying the laser intensity in finer gradations than the base addressable resolution allows smooth appearing lines to be produced at just about any angle (Fig. 2.8a).

For a *photocopier*, it is easiest to use the light reflected off the white part of the paper to discharge the PC. The same principle applies too when xerography is used to print microfilm or microfiche images. When using xerography as the initial printing mechanism, as with a laser printer, the PC is discharged where the ink is to accumulate. These two processes use the same basic principles, but opposite charges. Each cycle will deteriorate the quality of the resulting image, Fig. 2.9a–d, because degradations from both the printing process and the scanning process contribute to the resulting image; see the section “[Acquisition Methods](#).”

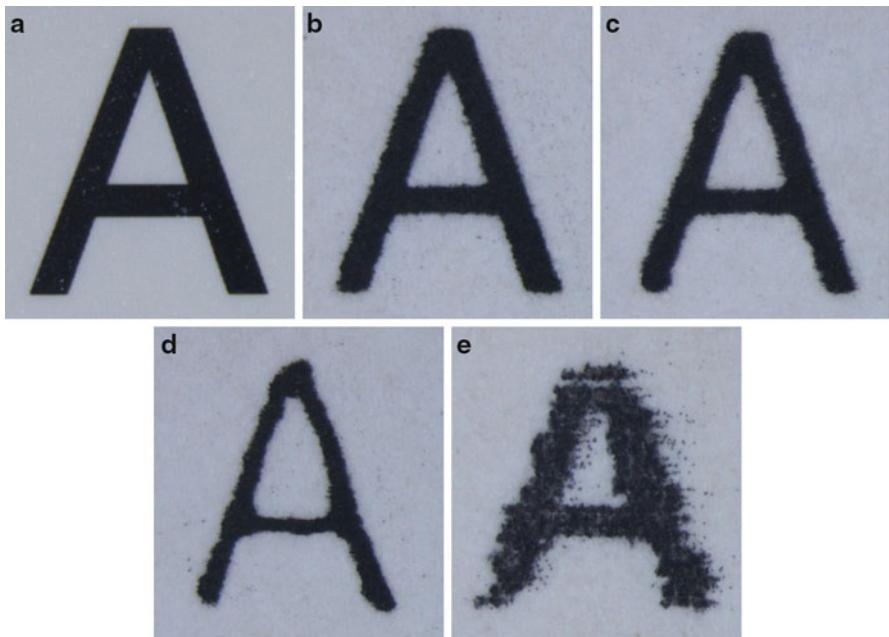
In a *facsimile* or *fax machine* images are spatially sampled in a location remote to the photoconductor. The samples are sent by telephone line to control a light source in a printer at a remote location. Because of the low bandwidth of the telephone line, facsimile machines usually sample the image at a lower resolution to transfer fewer samples. The printed image will thus have a lower quality than printing of an original on a laser printer, even though they use the same printing technology, Fig. 2.9e.

### Inkjet Printing

*Inkjet* printing is another nonimpact printing technology (Fig. 2.8b). Inkjet printing sprays a quantity of ink from a fine high-pressure nozzle onto the paper where it is absorbed. Inkjet printers use either a thermo- or piezoelectric method to control the ink flow. In the thermal method, the ink is heated to the boiling point, where built-up pressure in the nozzle tube ejects the ink. Alternatively, a piezoelectric-controlled pump-like device manually ejects the ink. In some instantiations, the ink will form a mist and produce a spattered appearance. Modern inkjet printers have developed techniques that reduce this effect. Inkjet printers may have a single nozzle for each



**Fig. 2.8** Examples of nonimpact printing methods. (a) Xerographic, (b) inkjet, and (c) image setter



**Fig. 2.9** (a) Original image. (b)–(d) Effect of repeated photocopying, copies 1, 2, and 7. (e) Output of fax machine

color that moves back and forth across the sheet of paper depositing ink or an array of nozzles that span the paper width depositing ink in parallel. This array-based process increases the inkjet printing speed to the point that is practical to customize the product from piece to piece in mass production runs by changing the document content.

### Other NIP Methods

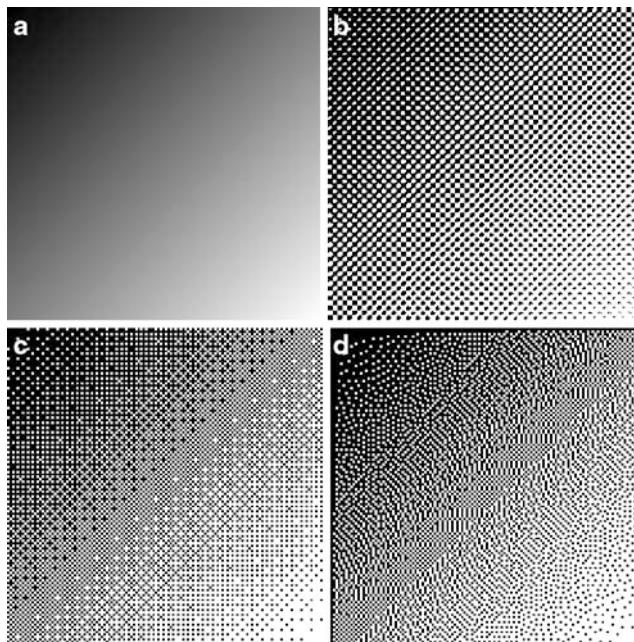
An *imagesetter* produces ultrahigh resolution images of documents by combining computer output with photographic technology (Fig. 2.8c). Imagesetters are often used to get high-quality proofs before another high-resolution printing mechanism is used that requires more time and materials in the setup process. Operating similarly to laser printers, imagesetters use a laser light source to expose film instead of (dis)charging a PC. The film is a high-sensitivity continuous tone film or a light room contact film. The film has a protective layer on top of a light-sensitive silver halide layer. Below this is a carrier base and then antihalation backing. After development, a high-quality black and white document image is visible. With a spot size of 7–45  $\mu\text{m}$ , resolutions up to 8,000 dpi are possible.

*Microfilm* and *microfiche* are two types of microforms. The images of documents are reduced to about 4 % of the original document size through a photographic process onto a photographic film substrate. Microfilm stores the images on reels of film. Microfiche stores the images on flat sheets of film. This storage mechanism was used for documents that were not often referenced but that had a large number of pages. This could significantly reduce the physical storage space required for these documents. The American Library Association endorsed this method of information storage in 1936. It is relatively inexpensive to make multiple copies of the films, and the films allow access to some documents that might be too rare or fragile for general access. This is similar to the reasoning behind many document digitization projects today. Pages from a microform could be viewed on readers, and selected pages could then be printed at original size. Microfilm itself gets scratched very easily. This deteriorates the image quality. Most microfilms are made with the silver halide process. These films if properly stored have a life expectancy of hundreds of years. However, in reality they have a much shorter shelf life than paper documents. After a few decades under certain environmental conditions, the old acetate films begin to slowly chemically decompose and produce a vinegar scent. Charts exist for acetate film that explain the relationship between temperature, relative humidity (RH), and the “vinegar syndrome.” For example, a fresh acetate film that is kept in a room maintained year round at 70 °F (21 °C) with a RH of 50 % will last for 40 years before the film exhibits the “vinegar syndrome.” At this point, the film is still useful, but deterioration begins to accelerate, requiring imminent action to avoid the loss of content.

### Multitone and Color Printing, Dithering and Screens

When images in more than the bi-level black and white are desired, the multitone or color effect is achieved by using a *dithering* or *screening* process. Here, the appearance of multiple levels of lightness or darkness is achieved by varying the percent of a viewing zone containing ink. Visually, the eye averages the image to produce the appearance of a range of tones related to the fill percentage.

Historically, this was achieved in monochrome by an engraving process where the image was converted to a texture with varying thickness lines. This only allowed coarse approximations, and creation of these drawings was very time consuming.



**Fig. 2.10** Examples of dithering methods. (a) Original image, (b) AM halftoning, (c) FM halftoning, and (d) error diffusion halftoning

In 1881, Georg Meisenbach introduced a process where a fine mesh or screen of diagonal lines crossed at right angles on a sheet of film or glass was placed in front of a photosensitive film. As the light from the image is passed through the screen, the lens breaks the image into thousands of small points of light which are stronger where the image is lighter and weaker where the image is darker. This is known as *AM halftoning* as the dots are uniformly spaced but have variable size and thus a variable brightness amplitude (Fig. 2.10b). Screens range from 65 lpi (lines per inch) for newspapers to 600 lpi for fine art and photography reprints. Most illustrated books are printed with 133 or 150 lpi line screens. At 150 lpi (60 l/cm), the eye can no longer detect the individual dots from a normal viewing distance of 12 in. (30 cm).

The percentage filled, and thus apparent darkness, can also be controlled by changing the number of dots per unit area. This technique is known as *FM halftoning*. The dots are of constant size and can be arranged in an orderly, uniformly distributed pattern or stochastically arranged to avoid the appearance of certain patterns that are noticeable to the eye and detract from the uniform field (Fig. 2.10c).

More advanced methods have been developed to produce more pleasing results. However, these come with a higher computational cost. One common method is *error diffusion*, which is based on sigma-delta modulation from one-dimensional

signal processing. The image is scanned and the pixel value is predicted based on linear combinations of the neighboring pixels. The sign of the error in the intensity value estimate determines whether or not to print a black dot at each location (Fig. 2.10d). Other methods make extra passes to remove certain dot combinations that are known to stand out to the human visual system. A process called direct binary search checks the dot arrangement and makes small changes so no specific texture artifacts are visible.

The halftoning algorithms assume that the dots will have a certain size and color when the algorithm is implemented to arrange them. The printing processes can cause the dot size to vary from the addressable resolution or the dot shape to differ from that modeled. This can make the total area appear brighter or darker in an artifact known as *dot gain*.

More than two gray values (on and off) are possible in many print technologies by varying the ink film thickness in a process referred to as *density modulation*. Combining density modulation with halftoning produces a greater number of apparent gray levels than halftoning with just two primary colors.

## Color

The visible spectrum includes light with wavelengths in the range of 350–750 nm. Light of all wavelengths is incident on a surface, and some wavelengths are absorbed while others are reflected. Those that are reflected then stimulate the cones in the human eye producing the appearance of light. There are three types of cones in the eye, each of which has a different sensitivity to, or “weighting” of, each light wavelength. How each cone is excited determines the color we perceive.

Similar to being able to create a range of gray tones from two base colors, black ink, and white paper, a range of colors can be created by combining a small set of color primaries with the white paper. To replicate the perception of a gamut of color, color documents are printed with three reflective color primaries, cyan (C), magenta (M), and yellow (Y). By mixing these three colors in small areas, the perceived color is the color that is *not* absorbed by the inks on the paper in a given region. Because the primaries are imperfect, when mixing all primaries not all the colored light will be absorbed, and usually a dark grayish brown results instead of the desired black. Thus printing systems usually also print black (K) as a fourth primary producing the CMYK color system. The space of colors spanned by a set of primaries is called a *color gamut*, and due to different ink primary properties, each printing and display system will have its own color gamut.

To break a color image into its color primaries for analog printing processes, the original image is photographed through a series of color filters that absorb colors of which it is composed and permits the remaining colors to be recorded. Color digital images come from scanning the image with a grid of sensors sensitive to a single color primary through smaller scale color filters.

When color images are printed, each color layer is screened separately. Use of AM halftoning is most common. The four colors are printed each at a different angle to avoid introducing interference or *moiré* artifacts. The most apparent color, black, is printed at an angle of 45° to be less visibly distracting. The least apparent color,

yellow, is printed at an angle of 10°. The remaining colors, cyan and magenta, are printed at angles of 75° and 15°, respectively.

### Digital Printing, Electronic Books and Displays

Computers and electronic technology have allowed document creation to take on many new forms. Some of these have increased the automation of the document creation process, so increased speed and precision are possible. Some have evolved the physical printing methods or allowed the document to be printed from a remotely created source. Another development has made the image itself digital from the start. ePens, discussed in the section “[Inks](#),” allow direct input of the pen trace into the computer. Displays allow the image of the document to be seen either directly from the word processing equipment (see [► Chap. 23](#) (Analysis of Documents Born Digital)) or after being digitized from hardcopy.

Electronic books have existed almost as long as computers, but the ready access to the material, and its adoption by society, is recent. Books and other documents have been available at libraries or through distribution on CD ROMs for viewing on computer displays. The jump to portability has come through the introduction and successful marketing of *eReaders* such as the Amazon Kindle, Barnes & Noble Nook, Sony Reader, and Apple iPad. eReaders are small electronic devices with a display about the dimensions of a medium paperback book. eReaders have enabled users to read traditional book content as eBooks. Some people appreciate being able to carry the content of several books in a physically small device. The increased speed with which new content can be accessed or updated, whether locally or remotely created is another advantage. They are also able to reduce waste from printing documents, like newspaper, that are not intended for long duration use. For these document types, the content analysis becomes the research focus. Further discussion of this type of document is covered in [► Chap. 23](#) (Analysis of Documents Born Digital).

### Displays

Images of documents before printing and after acquisition from their printed form are made visible through *electronic displays*. Displays can be emissive, transmissive, or reflective. The *emissive display* technology that dominated the market in the 1900s is the *Cathode Ray Tube (CRT)*. CRTs rely on cathodoluminescence. An electron beam is scanned across a phosphor screen, which then luminesces. The eye can retain memory of the light for a period of time, so the screen does not need to be imaged over the whole surface at once. The vertical information is refreshed 60 times per second in North America, and 50 times per second elsewhere. The signal is usually interlaced by scanning odd and even numbered lines in separate passes to increase the vertical speed, while avoiding flicker. For color images three electron beams are used which hit phosphor dots that luminesce either red, green, or blue when struck by the beam. The dots are arranged in triads and at angles such that only the correct electron beam will strike it. The control electrode strengthens or weakens the electron beam to vary the brightness level. CRT displays are characterized by

large bulky vacuum tubes that contain the electron beam, with a depth close to the size of the screen width. Flat panel displays have a much smaller depth profile and often use less energy. *Plasma displays* operate like fluorescent lights with a series of RGB pixels in long horizontal strips. The alternating current generates a plasma. The gas in the discharge radiates ultraviolet light when excited. This UV light then excites color phosphors via photoluminescence. *Light-emitting diodes* (LEDs) are another emissive display technology. LEDs of red, green, and blue colors are arranged in pixel clusters and produce the desired image. LED displays are most predominant as score boards and large advertising boards. Organic LEDs (OLEDs) are developing with smaller sizes and often flexible displays. These are used in some cell phones and PDAs (personal digital assistants) and are occasionally found in displays up to 19 in. (43 cm).

The most common *transmissive displays* are *liquid crystal displays* (LCDs). The liquid crystal is polarized, and based on an applied electric field, the direction of polarization can be changed. This is placed on top of another polarized layer. When the liquid crystal is polarized in the same direction as the base layer, light is permitted to pass. When the liquid crystal is polarized perpendicular to the static layer, light is blocked. Red, green, and blue color filters are applied in sets of three small vertical rectangles for each pixel, so the transmitted light is colored. LCDs require a back light source. This is most commonly cold cathode fluorescent light operating at high frequency, although sometimes the light source is a panel of white LEDs. The occasional use of LEDs as the light source sometimes leads those displays to be incorrectly called LED displays.

*Reflective displays* are used on many eReaders such as the Amazon Kindle and the Barnes & Noble Nook. They rely on an external light source, either the sun or a lamp, to provide the light. This light bounces off the display which in some areas absorbs the light and in others reflects it. This is the same principle used with paper documents, so these displays are sometimes referred to as *ePaper* or *eInk* depending on whether the white or the black part of the display provided the name. Reflective displays generally take less energy because they do not need to produce the light and because they do not need to be constantly refreshed to maintain the image but can be readdressed to change the image content. The reflective displays often produce less eye strain because the light is less harsh, and they are low glare and so can be used in a variety of lighting conditions. Another characteristic of these displays is that they can be much thinner and lighter than other displays and sometimes flexible like paper. Two main approaches to ePaper have been developed. One uses technology developed by Xerox PARC in the 1970s, which was spun off to Gyronix Media Inc. and 3M. This consists of spheres of diameter approximately 100  $\mu\text{m}$  that are painted half white and half black. This gives an effective resolution of about 250 dpi. The spheres change their orientation based on the application of an electrical field. The other technology, developed by Massachusetts Institute of Technology (MIT) and eInk Corp., fills microcapsules with a dark liquid and many charged white pigment particles. Depending on the electrical field the charged particles will move either to the top of the capsule near the surface of the paper to produce white or to the bottom so the dark fill is visible.

## Acquisition Methods

People use digital computers to perform document image analysis on digital images. There are many methods possible for converting a physical spatially continuous document image into a digital representation. These methods have developed to accommodate the broad range of document types that are of interest.

### Flatbed Scanner and Fax Machine Acquisition

Documents are traditionally printed on flat paper which led to the development of the *flatbed scanner* as the primary method of acquisition. For acquisition on flatbed scanners, a page is placed face down on the glass platen. The scanner has a scanning bar, which extends across the width of the page and moves down the length of the page taking a series of pictures, one for each row of the total image. In some scanner configurations, the light and sensor array are stationary, and the platen or paper moves. To take a picture of one row in the page image, a light source shines on the image. A series of mirrors redirects the light reflected from the original image to a focusing lens. This lengthens the effective focal length of the lens. The light is then directed to the sensor array, which collects charge as the bar moves continuously along the page. The optical resolution in the scanning bar direction depends on the distance the scanning bar moves during the exposure time of the sensors. The time spent traversing the distance corresponding to 1 pixel length is set such that the appropriate amount of charge can be accumulated. Resolution in the transverse direction is determined by the number of sensor elements in the linear array and the magnification of the optical system. The readings from each row are accumulated to form the two-dimensional (2D) bitmap for the image, which is then passed from the processor in the scanner to the computer. Some scanners have a 2-D array of sensors, so data for the whole image is acquired in one step, usually without moving parts.

The sensors can be either *charged couple devices (CCDs)* or *complementary metal-oxide silicon (CMOS)*-based devices. Both sensor types have a charged silicon layer that when exposed to light will dislodge electrons from the silicon lattice that when “counted” by the electronics is converted to a voltage that has a monotonic relationship with the amount of light reaching the sensor area. This voltage is then converted into a digital signal. CCD sensors pass the charge reading out along a row, cell by cell. CMOS sensors can be addressed in two dimensions so all values can be read essentially at the same time since they acquire an image in a 2-D array of sensor elements, removing the need to move a 1-D array of CMOS sensors relative to the page to acquire a 2-D image. CCDs are found in older equipment and higher end equipment. Until recently, the quality of the CCD-based sensors was notably higher than the quality of the CMOS sensors. CMOS sensors were preferred for low-cost applications because the manufacturing process for CMOS sensors could be done at the same time as processing for the accompanying circuitry.

The quality of the CMOS sensors has increased to the point where they are being used across a broad range of products and are becoming common even in high end devices.

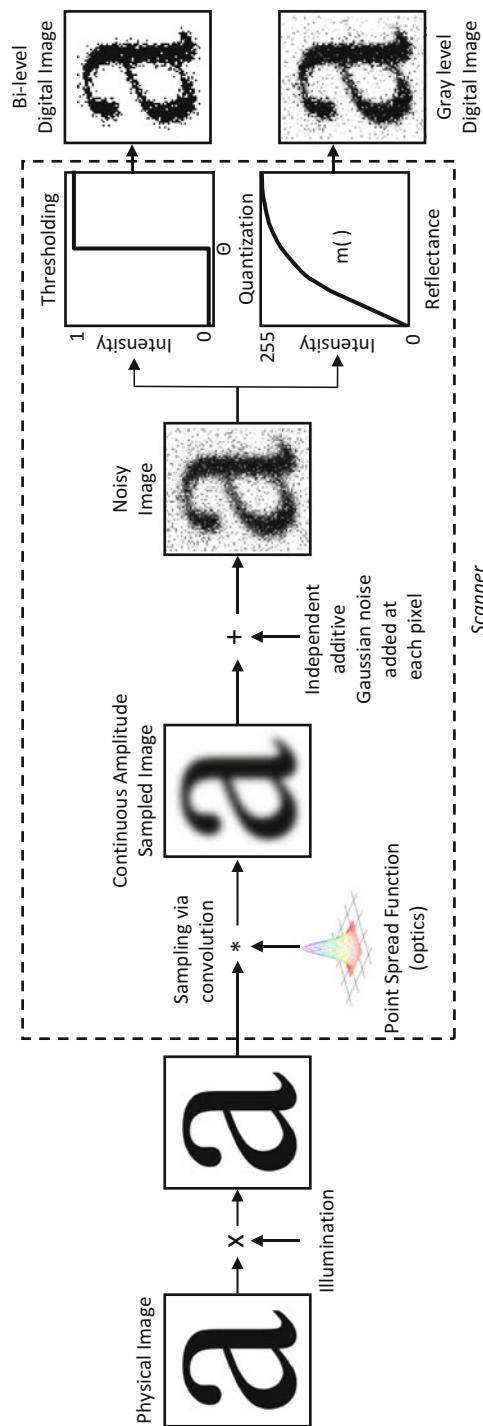
The physical 2-D document image is converted into a two-dimensional array of numbers (pixels) during the scanning process (Fig. 2.11). The actual darkness of the writing substrate (paper) and the ink together with the brightness of the illumination produce the recorded pixel values image. When the document is acquired on a flatbed scanner, the illumination is carefully controlled to produce a near uniform value to get a better representation of the document. Misfeeding of a page through a scanner can cause image distortion, such as skew.

During acquisition, light reflected off the document passes through lenses and is focused on an array of sensors. Each sensor returns a scalar value which is the average of the light over a region in the document. This can be mathematically modeled as 2-D convolution with impulse sampling. The light received at each sensor will come from a limited region on the paper and will also usually come from that area nonuniformly. The *point spread function* (PSF) describes how the light received at the sensor is spatially weighted. This can be measured using a “knife edge” image, which is a 2-D step input, and differentiating the response. This procedure has been standardized in the International Standard Organization’s ISO 12233 [1] procedure.

Noise is often unintentionally added to the sampled image. The noise can come from the variations in the paper and ink intensities as well as sensor noise. This is usually modeled as being additive i.i.d. Gaussian noise. Sometimes the noise that results has more of the characteristics of shot or impulse noise.

The acquired signal has continuous valued intensities. These are quantized to convert it to a digital signal. Most scanners initially store 16–48 bits of gray levels but then convert it to 2 or 256 levels per color channel based on the user’s choice of acquisition settings. An efficient quantization scheme is needed to retain the greatest amount of information. To determine the level reduction, either the scanner or user selects *brightness* and *contrast* values. When the scanner performs this function, it uses an automatic algorithm based on data sampled from the image.

A high (low)-brightness setting will map the input reflectances to a higher (lower) numerical value in the gray-level digital image. Contrast is the magnitude of the difference in output gray level between the brightest white and the darkest black. Therefore, a high-contrast setting will allow a larger range of gray values in the image than a low-contrast setting. This is used to accentuate highlights and shadows (Fig. 2.12). To avoid saturation, the scanner brightness and contrast settings should be adjusted such that the range of reflectances from black ink to white paper does not exceed the 0–255 range. The brightness and contrast can also be changed in post processing (see ►Chap. 4 (Imaging Techniques in Document Analysis Processes)), but if the scanner settings cause over- or undersaturation because the reflectances map to a gray level outside the 0–255 range, this cannot be compensated for later. The gray-level mapping function for a given scanner can be determined by scanning a test chart with calibrated gray-level reflectance patches and noting their gray-level value in the resulting image.



**Fig. 2.11** Model of the scanner acquisition process

**a**

ured  $\tau_1$  value to the  $\tau_1$  value  
placement curve will yield

(4.22)

(4.23)

to determine  $\Theta$ :

(4.24)

nd  $\Theta$ , for which the theor-  
n  $f_{r, \text{noisy}}(\tau)$ , data. Best fit is

(4.25)

4.2.1. Then the parametric  
gression. As described in  
a form for which a unique  
ure 4.9. Within the shaded  
escribe  $f(r)$ . The equations  
d star, there is a single scan-  
l be constant. In the test tar-  
4.9, this corresponds to a

pend on the value of  $\Theta$  and  
unctional forms depends on  
h horizontal slice, the MSE



Image Processing and Analysis

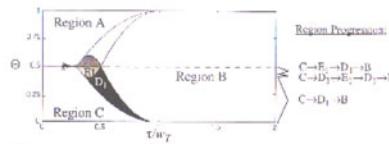


Figure 4.9: Fraction regions for triangular PSF.

the functional form of region  $E_1$ , then to the functional form of region  $D_1$  and finally to the functional form of region  $B$ . For  $\Theta$  values between these two ranges, the progression would be  $C \rightarrow D_1 \rightarrow E_1 \rightarrow D_1 \rightarrow B$ . A similar set of progressions occur for  $0.5 < \Theta < 1.0$ .

To fit the parametric fraction to the data, assumptions about which of these six categories of progressions is valid must be made. Then the location of the boundaries between regions with respect to the data must be hypothesized. Each data region will be tested separately by regression to find the best parameters. The partition location whose estimate of  $(w_T, \Theta)$  gives the minimum MSE is considered correct and the corresponding  $(w_T, \Theta)$  estimates are designated to be the correct parameter values.

The regions F, G, etc. (regions where  $0 < r(\tau) < 1$  and  $\tau/w_T < 1/4$ ) have been eliminated in the implementation because they are very small and the effect of the star being in two dimensions is likely to obscure the data in these regions any ways. If necessary, they could be added to the implementation.

#### 4.2.4 Implementation of Estimation from Wedges

The final estimation method uses isolated wedges. The location of the two straight edges and the apex of the blurred wedge must be found from the image and together they determine the PSF width and threshold value.

The pixels denoting the top and bottom edges of the wedge are identified by applying a MATLAB contour function to the image and these coordinates are fit to straight

**b**

ured  $\tau_1$  value to the  $\tau_1$  value  
placement curve will yield

(4.22)

(4.23)

to determine  $\Theta$ :

(4.24)

nd  $\Theta$ , for which the theor-  
n  $f_{r, \text{noisy}}(\tau)$ , data. Best fit is

(4.25)

4.2.1. Then the parametric  
gression. As described in  
a form for which a unique  
ure 4.9. Within the shaded  
escribe  $f(r)$ . The equations  
d star, there is a single scan-  
l be constant. In the test tar-  
4.9, this corresponds to a

pend on the value of  $\Theta$  and  
unctional forms depends on  
h horizontal slice, the MSE



Image Processing and Analysis

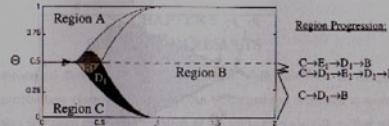


Figure 4.9: Fraction regions for triangular PSF.

the functional form of region  $E_1$ , then to the functional form of region  $D_1$  and finally to the functional form of region  $B$ . For  $\Theta$  values between these two ranges, the progression would be  $C \rightarrow D_1 \rightarrow E_1 \rightarrow D_1 \rightarrow B$ . A similar set of progressions occur for  $0.5 < \Theta < 1.0$ .

To fit the parametric fraction to the data, assumptions about which of these six categories of progressions is valid must be made. Then the location of the boundaries between regions with respect to the data must be hypothesized. Each data region will be tested separately by regression to find the best parameters. The partition location whose estimate of  $(w_T, \Theta)$  gives the minimum MSE is considered correct and the corresponding  $(w_T, \Theta)$  estimates are designated to be the correct parameter values.

The regions F, G, etc. (regions where  $0 < r(\tau) < 1$  and  $\tau/w_T < 1/4$ ) have been eliminated in the implementation because they are very small and the effect of the star being in two dimensions is likely to obscure the data in these regions any ways. If necessary, they could be added to the implementation.

#### 4.2.4 Implementation of Estimation from Wedges

The final estimation method uses isolated wedges. The location of the two straight edges and the apex of the blurred wedge must be found from the image and together they determine the PSF width and threshold value.

The pixels denoting the top and bottom edges of the wedge are identified by applying a MATLAB contour function to the image and these coordinates are fit to straight

**Fig. 2.12 (continued)**

**c**

ured  $\tau_1$  value to the  $\tau_1$  value  
placement curve will yield

(4.22)

(4.23)

to determine  $\Theta$ :

(4.24)

nd  $\Theta$ , for which the theor-  
n  $f_{r, \text{new}}(\tau)$ , data. Best fit is

(4.25)

4.2.1. Then the parametric  
gression. As described in  
a form for which a unique  
ure 4.9. Within the shaded  
escribe  $f(r)$ . The equations  
d star, there is a single scan  
be constant. In the test ta-  
4.9, this corresponds to a

pend on the value of  $\Theta$  and  
unctional forms depends on  
h horizontal slice, the MSE

Image Processing and Analysis

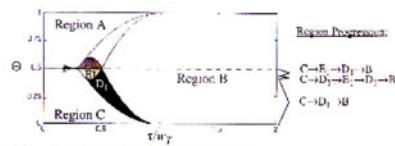


Figure 4.9: Fraction regions for triangular PSE

the functional form of region  $E_1$ , then to the functional form of region  $D_1$  and finally to the functional form of region  $B$ . For  $\Theta$  values between these two ranges, the progression would be  $C \rightarrow D_1 \rightarrow E_1 \rightarrow D_1 \rightarrow B$ . A similar set of progressions occur for  $0.5 < \Theta < 1.0$ .

To fit the parametric fraction to the data, assumptions about which of these six categories of progressions is valid must be made. Then the location of the boundaries between regions with respect to the data must be hypothesized. Each data region will be tested separately by regression to find the best parameters. The partition location whose estimate of  $(w_T, \Theta)$  gives the minimum MSE is considered correct and the corresponding  $(w_T, \Theta)$  estimates are designated to be the correct parameter values.

The regions F, G, etc. (regions where  $0 < f(\tau) < 1$  and  $\tau/w_T < 1/4$ ) have been eliminated in the implementation because they are very small and the effect of the star being in two dimensions is likely to obscure the data in these regions any ways. If necessary, they could be added to the implementation.

#### 4.2.4 Implementation of Estimation from Wedges

The final estimation method uses isolated wedges. The location of the two straight edges and the apex of the blurred wedge must be found from the image and together they determine the PSF width and threshold value.

The pixels denoting the top and bottom edges of the wedge are identified by applying a MATLAB contour function to the image and these coordinates are fit to straight

**d**

ured  $\tau_1$  value to the  $\tau_1$  value  
placement curve will yield

(4.22)

(4.23)

to determine  $\Theta$ :

(4.24)

nd  $\Theta$ , for which the theor-  
n  $f_{r, \text{new}}(\tau)$ , data. Best fit is

(4.25)

4.2.1. Then the parametric  
gression. As described in  
a form for which a unique  
ure 4.9. Within the shaded  
escribe  $f(r)$ . The equations  
d star, there is a single scan  
be constant. In the test ta-  
4.9, this corresponds to a

pend on the value of  $\Theta$  and  
unctional forms depends on  
h horizontal slice, the MSE

Image Processing and Analysis

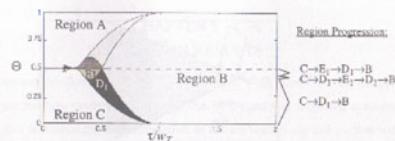


Figure 4.9: Fraction regions for triangular PSE

the functional form of region  $E_1$ , then to the functional form of region  $D_1$  and finally to the functional form of region  $B$ . For  $\Theta$  values between these two ranges, the progression would be  $C \rightarrow D_1 \rightarrow E_1 \rightarrow D_1 \rightarrow B$ . A similar set of progressions occur for  $0.5 < \Theta < 1.0$ .

To fit the parametric fraction to the data, assumptions about which of these six categories of progressions is valid must be made. Then the location of the boundaries between regions with respect to the data must be hypothesized. Each data region will be tested separately by regression to find the best parameters. The partition location whose estimate of  $(w_T, \Theta)$  gives the minimum MSE is considered correct and the corresponding  $(w_T, \Theta)$  estimates are designated to be the correct parameter values.

The regions F, G, etc. (regions where  $0 < f(\tau) < 1$  and  $\tau/w_T < 1/4$ ) have been eliminated in the implementation because they are very small and the effect of the star being in two dimensions is likely to obscure the data in these regions any ways. If necessary, they could be added to the implementation.

#### 4.2.4 Implementation of Estimation from Wedges

The final estimation method uses isolated wedges. The location of the two straight edges and the apex of the blurred wedge must be found from the image and together they determine the PSF width and threshold value.

The pixels denoting the top and bottom edges of the wedge are identified by applying a MATLAB contour function to the image and these coordinates are fit to straight

**Fig. 2.12** Page scans showing (a) high brightness (shows saturation), (b) low brightness, (c) high contrast, and (d) low contrast

If a bi-level image is to be produced, only the brightness setting is variable. This controls the *threshold level*,  $\Theta$ , used to convert the analog reflectance to a bi-level value

$$\text{bilevel}[i, j] = \begin{cases} 1 & \text{analog}[i, j] > \Theta \\ 0 & \text{analog}[i, j] < \Theta. \end{cases}$$

Values at every pixel will be converted by the same, global, threshold. Images acquired in gray scale usually are converted to bi-level through binarization before recognition algorithms are applied. These can be global thresholding algorithms or one of a variety of adaptive thresholding algorithms that target cases where the contrast between text and background is not uniform across the page or has significant noise. These are described further in ([►Chap. 4](#) (Imaging Techniques in Document Analysis Processes)).

There is a tradeoff between spatial sampling rate and gray depth. Gray scale acquisition can be completed at a lower resolution and still produce the same optical character recognition (OCR) accuracy as bi-level acquisition.

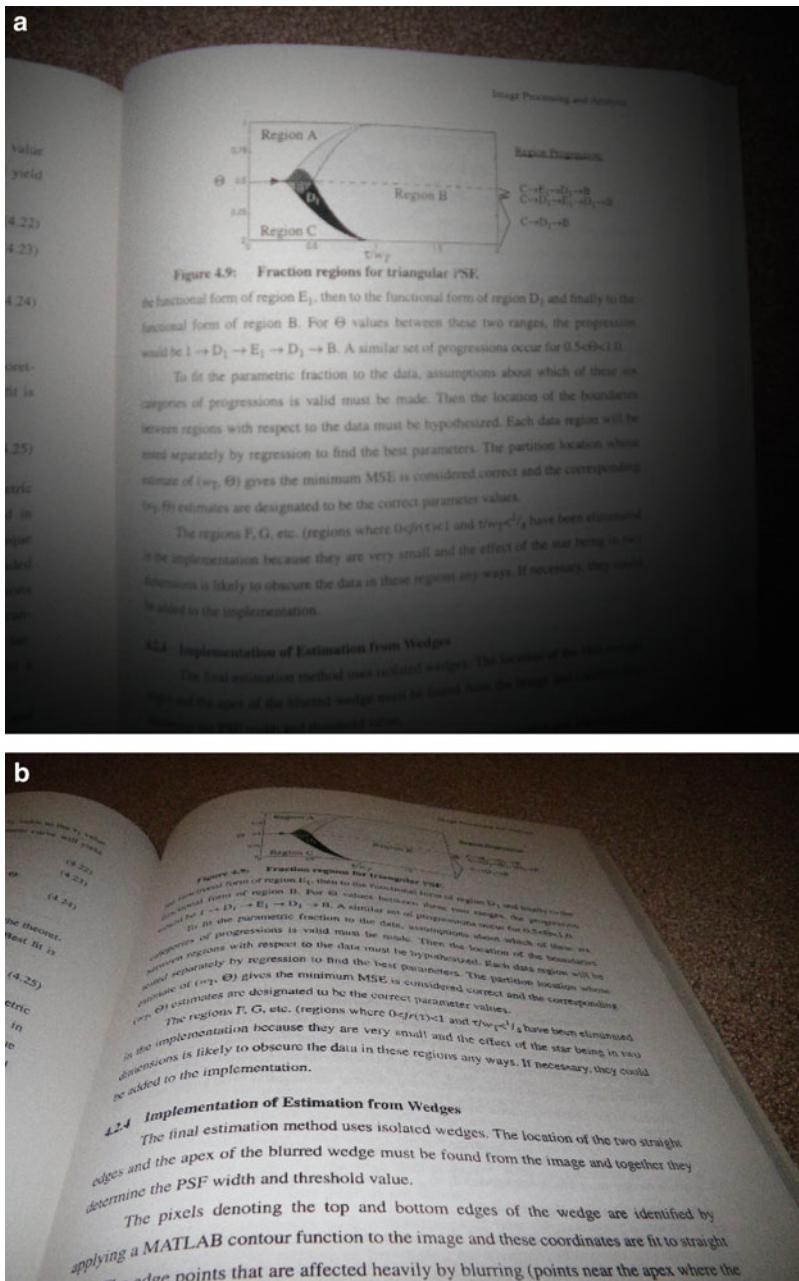
*Facsimile*, or fax, machines incorporate a type of flatbed scanner, but scan at much lower resolutions. The resolution is most commonly  $100 \times 100$  or  $100 \times 200$  dpi, Fig. 2.9e. The low resolution was chosen from earlier transmission bandwidth constraints. The fax machine sends the acquired image to a remote location for printing, rather than saving it for local processing.

Color scanners employ the same acquisition process, but divide the reflected light signal into three channels. Each channel is passed through a color filter. A process equivalent to gray-level acquisition is used on each color channel.

## Cameras and Mobile Devices

With camera-based acquisition, traditional paper documents are no longer the only content that can be acquired. Billboards, business signs, posters, and meeting room white boards are among the new material that is acquired and analyzed (see [►Chap. 25](#) (Text Localization and Recognition in Images and Video)). Camera-based acquisition also removes the restriction that the object being captured must be a planar object. Text on bottles or round sign pillars can also be acquired. Cameras usually acquire the image in color as they are not designed specifically for bi-level document image acquisition.

Cameras and mobile devices operate on principles similar to those of flatbed scanners, but the position of the sensor relative to the object is now variable and so is the illumination. This increases the range of image distortions. In camera-based acquisition, the light is often a near field point source so the illumination pattern can be nonuniform. It is common for it to have a maximum intensity near the center decreasing with distance from the center, usually with radial symmetry. This often results in the corners of the image being noticeably darker than the center, sometimes making paper at the corners appear darker than the text at the center (Fig. 2.13a). Glare from flash, or from ambient light, can oversaturate parts of the image, obscuring parts of the image.



**Fig. 2.13** Camera-based acquisition. **(a)** Nonuniform lighting and glare **(b)** perspective distortion

Cameras and mobile devices often have lower effective resolution than flatbed scanners. Both the camera optics (lenses) and sensor sensitivity contribute to this. Newer model cameras are able to acquire document images (A4 and US letter) at close to 300 dpi resolution. Some specialized cameras and document platforms have been developed to acquire book images. The Xerox Book Scanner was one of the early ones. The Google Million Book project utilizes camera-based imaging.

If the camera is not placed directly parallel to the image plane, perspective image distortions will result (Fig. 2.13b). These can be corrected for in post processing if the angle can be correctly determined (see ►Chap. 4 (Imaging Techniques in Document Analysis Processes)). Variable positioning of the camera can result in part of the image being out of focus, requiring more correction than just perspective or the introduction of motion blur. These defects can be corrected to some degree by methods described in ►Chap. 4 (Imaging Techniques in Document Analysis Processes).

## Video

Video in its most basic form is a temporal sequence of camera-based images. The object being captured can now exhibit motion relative to the sensor. This can cause motion blurring degradations and the need for object tracking. Text designed to provide information to the viewer that is computer generated and not acquired through the video device can be superimposed on the image such as with closed captioning.

For digital processing, analog video must be converted to a digital sequence. This involves spatial resampling and intensity quantization. In an analog recording, the vertical resolution is quantized, but along the horizontal axis, the signal is traced and an analog recording of the observed reflectance is acquired. Video is usually a color signal, and analog video is stored to be compatible with analog televisions. The color is therefore stored as an intensity channel and two color channels, instead of RGB as the image is acquired or as digital color images are often stored. The color channels are usually stored at a lower quality level to take advantage of compression possible based on the human visual system. As much of the world is converting to digital television, analog to digital video conversion will be less of an issue in the future, although digital video compression will still convert the colorspace and introduce artifacts.

Other image processing steps are usually needed when recognizing text in a video sequence. Even if the document is stationary, in video the camera may be moving so there is a possibility that the image is blurred. The document will appear in multiple image frames, so tracking of the document from frame to frame is needed. How these and other artifacts are handled is discussed in more detail in ►Chap. 25 (Text Localization and Recognition in Images and Video).

## Other Specialty Modes

While the majority of acquisition is done in the visible light band, there are applications that need other acquisition modes, such as historical document OCR and forensic OCR. For historical documents, the ink may have faded, or the paper may have yellowed or been damaged by watermarks or mold. These all reduce the contrast between the text and the background, sometimes to the point where they cannot always be distinguished in the visible spectrum. In the special historical document category of palimpsests, where the ink was deliberately removed to reuse the parchment or vellum to create a new book, trace particles from the original layer of ink may still be present and of interest to researchers, but not in quantities that will allow adequate detection in the visible band. Likewise, when trying to determine origins of questioned documents, details about the document beyond those observable in the visible band may be necessary. Differentiation between similar but different inks is often sought to identify forgeries. Identification of the type of ink or paper can provide a date of origin and determine authenticity of the document as a whole or certain parts of the text. The inks may appear similar in the visible band, but due to different chemical composition or decay rates, other imaging techniques may show a distinction.

Filters can be applied to cameras to accentuate color or spectral differences. Likewise, light at different wavelengths can be used to illuminate the document. Infrared or ultraviolet light sources are often used to show different inks. The same acquisition procedures discussed for visible band scanner or camera acquisition apply to these methods as well, but the light will be in a different frequency band. In addition, if there is a significant enough difference between the reflected light for text and substrate regions at those wavelengths, the text can become visible or show characteristically different properties, allowing different ink samples to be distinguished. Microfilm and microfiche systems need a projection system to project visible light through the film and receive it at a sensor.

Sometimes, the light changes frequency when interacting with the ink or paper, and this change in frequency is measured at the sensor, such as in Raman spectroscopy. Another method uses the incident light to cause the materials in the document to fluoresce. Ultraviolet illumination can cause organic material, such as in parchment, to fluoresce, emitting longer wavelength light that may be in the visible band. The ink will block some of this fluorescence, making it visible, although the document is not reflecting visible light. Several acquisition modes can be used in conjunction if the images are properly registered after imaging, and their combination can highlight document features.

Documents are usually thought of as being 2-D planar or nearly planar surfaces, and 2-D imaging techniques are most often used. Occasionally, the document is rolled or bound and has become fragile so transforming the document to a physically planar object is not possible. In these cases, 3-D imaging techniques, such as MRI and PET as used in medical imaging, can be called upon to provide images of

the document. The 2-D paper surface will be a planar curve in the interior of the acquired volume “cube.” The 2-D surface can be extracted and “flattened” digitally.

---

## Document Quality

The factors affecting quality are many, and ways to specify or quantify them exist. Some defects have been modeled. Models help understanding and can improve algorithms that work with images containing defects. When the image quality is low, it can affect analysis and recognition results.

It is always desirable to have high document quality for both reading and analysis. However, humans and computers do not always define image quality the same way. For humans, the ability to comfortably read the content is the deciding goal, but more than the individual characters affect this ability. Three levels of document quality exist:

- *Decipherability* – the ability to identify and read correctly a certain letter form
- *Legibility* – the ability to identify and read correctly a certain word form
- *Readability* – the ability to identify and read a certain meaningful chain of words correctly

Image quality can also be determined by the computer’s ability to correctly recognize characters or words, as indicated in character and word error rates. Physiological comfort and familiarity are only human factor issues, not a machine readability issue, but both computers and humans look for contrast, uniform font, and variations between character classes.

Unfortunately, quality can decrease based on several factors: the quality of the materials used in document production, the method used to produce the document, the content, and the acquisition method. Knowledge of the document content and lexicographic clues influence these items. This is also influenced by typography: the choice of typeface, size, boldness, length of line, and margins (size and justified/even or uneven) or paper gloss or glare. For a computer, these clues can be included in its recognition method.

## Factors Affecting Document Quality

There are many sources of document image degradations. They can be categorized into three areas:

1. defects in the paper (yellowing, wrinkles, coffee stains, speckle)
2. defects introduced during printing (toner dropout, bleeding and scatter, baseline variations, show-through)
3. defects introduced during digitization through scanning (skew, mis-thresholding, resolution reduction, blur, sensor sensitivity noise, compression artifacts)

To provide an objective and quantifiable measure of the quality of newly printed documents, an international standard ISO 13660 [2] was developed. The standard provides a set of device-independent image quality attributes, along with

**Table 2.3** Summary of attributes used in print quality standard ISO 13660

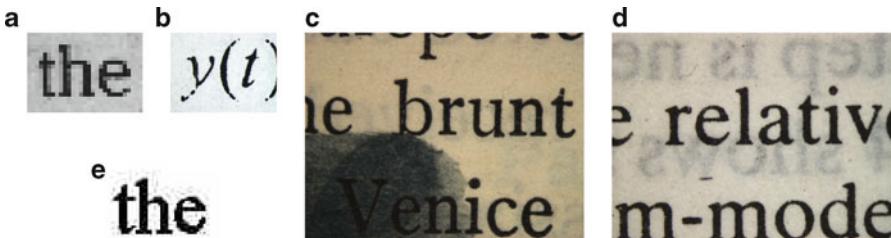
Large area density attributes	
Darkness	How dark the large inked areas are in terms of optical density
Background haze	Whether colorant/ink is visible in the background field but not as specific individual marks
Graininess	Standard deviation of aperiodic fluctuations of optical density at spatial frequencies greater than 0.4 cycles/mm
Mottle	Aperiodic fluctuations of optical density in printed zones at spatial frequencies less than 0.4 cycles/mm
Extraneous marks	(Count of visible unintended regions of colorant in the background area and voids (holes or gaps) within a solid image area)
Character and line attributes	
Blurriness	Rate of transition from black to white
Raggedness	The amount the path of an edge boundary varies locally
Line width	Average stroke width
Darkness	Optical density
Contrast	The relationship between the darkness (reflectance) of a line segment or character image and the background
Fill	Appearance of homogeneity of darkness within the boundary of a line segment or character image
Extraneous marks	Unintended regions of colorant in the background area near a character or line segment
Background haze	Colorant near a character that is visible but not resolvable as distinct marks at standard viewing distance to the unaided eye

measurement methods, and analytical procedures to describe the quality of high-contrast documents across a range of printers including impact printers, nonimpact printers, and photocopies. The standard includes attributes to measure features over large areas such as the background or large solid-filled areas and to measure text and line quality attributes, Table 2.3. All of these factors intended for the printer development community also affect document image analysis.

The attributes in ISO 13660 target newly printed documents. In document image analysis, document quality is also affected by photocopying, digitization, binarization, compression artifacts, aging, fading, yellowing, bleedthrough, and show-through.

When digitizing an image, spatial resolution is lost, and this may obscure some image features (Fig. 2.14a). During printing or copying, ink may drop out (Fig. 2.14b). Photocopying in addition to digitization involves reprinting, a process that may cause additional loss of features, Fig. 2.9. If the image is high contrast and on a uniform background, binarization is not an overly difficult problem, but if the document has aged, or was written in a lighter ink, there may be fading, yellowing, and staining which complicate the binarization problem (Fig. 2.14c).

Thin or backlit paper, or documents written with a very penetrable ink, can face problems where the text on the verso side becomes visible on the recto side as bleedthrough or show-through, again decreasing the document quality (Fig. 2.14d).



**Fig. 2.14** Examples of degradations affecting document image quality. (a) Low-resolution scanning, (b) ink dropout, (c) marks, (d) show-through, and (e) JPEG image compression artifacts

Images can require a significant amount of digital memory to store. To compensate for this, various techniques have been developed that can reduce the memory requirements by taking advantage of redundancy in image content. If the compression technique allows reconstruction of the original image with no change in any of the pixel values, then it is called a lossless compression scheme, such as used in PNG, GIF, and TIF. However, some compression schemes, like JPEG, make small changes to the image that are designed to be minimally observable by humans. Some changes alter individual pixel values, and when color images are saved in JPEG format, the hue and chrominance in  $2 \times 2$  squares of pixels are averaged and one value instead of four is stored for each. These minimal changes are more noticeable in high-contrast images than in natural scene images and can cause greater effects on automated processing (Fig. 2.14e).

## Effects of Document Quality on Analysis and Recognition Results

The goal in optical character recognition is to accurately recognize all the characters in the document image. Character recognition and document analysis algorithms will perform best if the image is cleanly printed on quality paper with dark, non-spreading ink. Documents with these attributes make up only a small portion of the documents being processed. Research has been done to automatically determine if the image quality is “good.” One of the most common metrics is OCR error rate. Early definitions of document image quality determined whether a page was likely to have few enough OCR errors to make it cheaper to run OCR applications and correct, versus to hand type a document in its entirety. If the image has low quality, then the OCR accuracy is likely to be lower. Low OCR accuracy leads to the need to apply more algorithms to improve the image or post processing algorithms to fix or compensate for the errors, and if the quality is too low, then manually entering the document content may be more efficient.

Some common OCR errors are the letters *r* and *n* being merged and interpreted as an *m*, the letter *m* being broken and interpreted as an *r* followed by an *n*, or an *e* losing its horizontal bar or having its loop filled in so it is recognized as a *c* (Fig. 2.15). Rice et al. [3] looked at large-scale studies of common OCR errors and summarized the sources of these errors, providing both samples of



**Fig. 2.15** (a) Original characters and (b) characters after degradation. Through touching and broken artifacts, recognition algorithms can easily misclassify the characters

**Table 2.4** Quality measures used to quantify document image degradations

Quality measure	Description	Method	Sources
Small speckle factor (SSF)	The amount of black background speckle in the image	Number of the black connected components in an image that contain a range of pixel counts	Inkjet, laser print, sensor noise, JPEG artifacts
White speckle factor (WSF)	The degree to which fattened character strokes have shrunken existing holes in or gaps between characters causing several small white islands to form	The ratio of the number of white connected components less than $3 \times 3$ pixels in size relative to the number of white connected components in the total image	Ink dropout
Touching character factor (TCF)	The degree to which neighboring characters touch	Number of long and low connected components	Ink bleeding
Broken character factor (BCF)	If characters are broken, there will be many connected components that are thinner than if the characters were not broken	The number of thin connected components	Ink dropout

the degraded text and the common OCR results. One category of errors was due to imaging defect, which included heavy print, light print, stray marks, and curved baselines. Some image features have been identified that are correlated with OCR error rates. Documents with low recognition rates often have touching characters from stroke thickening and broken characters from stroke thinning. These degradations lead to certain characteristics that can be arbitrarily measured from a document page. Quality measures (Table 2.4) look at quantifiable features common in characters [4] and [5].

Document images are subjected to image processing algorithms (►Chap. 4 (Imaging Techniques in Document Analysis Processes)) to improve the quality of the image before being passed to the recognition stage. The effectiveness of these algorithms is often evaluated based on the relative OCR accuracy achievable with different techniques. While it is the image processing algorithm being adjusted, and thus affecting the recognition results, compatibility of the image processing and the recognition algorithm also determines the net results.

## Models of Document Degradations

Document creation was described in the section “Writing and Printing Processes” of this chapter. Documents can be created by handwriting, hand printing, or by

**Table 2.5** Summary of degradation models

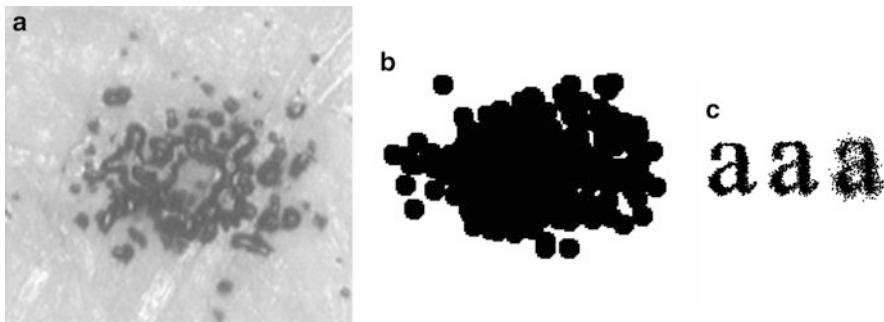
Model	Type of Process Modeled
<b>Franke and Rose</b>	Handwriting ink deposition
<b>Norris and Barney Smith</b>	Electrophotographic printing
<b>Baird</b>	Machine print printing and acquisition
<b>Obafemi-Ajayi and Agam</b>	Typewriter printing
<b>Kanungo</b>	General machine print degradation
<b>Show-through</b>	Appearance of ink from recto on verso
<b>Book binding</b>	Curvature of page

machine printing. As has been described, document images can contain various degradations, and they almost always decrease the image processing or recognition algorithm performance. Many of these degradations have been modeled mathematically (Table 2.5). The models aid development and analysis of image processing and recognition algorithms. At times they also allow researchers to generate a greater supply of samples for algorithm evaluation. Some are physics based, some aim to mimic a specific degradation, and others aim to produce degradations in general, not specifically related to any physical cause such as a problem with a particular paper or ink. Some models describe a specific document creation process, like pen handwriting, laser printing, or typewriters. Others describe the image acquisition process.

Physical movements of the pen relative to the paper affect the ink deposited on the paper in hand-printed or handwritten documents determining a relationship between the pen angle and pen force and the resulting ink intensity. The type of pen, its corresponding ink, and the type of paper influence the resulting ink intensity for a given pen force. The ink deposition model by Franke and Rose [6] relates the intensity to the applied force for solid, viscous, and liquid (or fluid) inks. This is a useful model for both handwriting recognition (►Chap. 11 (Handprinted Character and Word Recognition)) and document forensic analysis.

Nonimpact electrophotographic printing (i.e., laser printing) has been modeled by Norris and Barney Smith [7]. This provides ways to generate and analyze printed samples including the degradations that occur from toner spatter (Fig. 2.16). When the photoconductor drum is discharged by a laser with a Gaussian intensity profile, the charge change on the photoconductor will be proportional to the light intensity. Toner will stick to the photoconductor proportional to the amount of charge. That toner is then transferred to the paper and fused. The model depends on the radius of the toner pieces and the toner spatial density. The amount of paper actually covered by toner can also be modeled. Higher toner density will appear darker to the eye, scanner, or camera. The toner density and the coverage can be spatially varying functions and can provide a description for a whole page.

Typewritten characters have characteristic degradations of uneven ink intensity due to uneven typewriter key pressure, faded ink, or filled-in characters due to gummed type (Fig. 2.5). Degradations seen in characters produced through impact printing from a typewriter were modeled by Obafemi-Ajayi and Agam [8], Fig. 2.17.



**Fig. 2.16** Examples of (a) real and (b) simulated print samples. (c) Sample characters made with the Norris printer model



**Fig. 2.17** Examples of characters produced by the Obafemi-Ajaiy model. Filled and broken samples degraded at multiple levels

Broken characters are created by randomly selecting windows of varying size and flipping the foreground pixels from black to white. Filled or thickened characters can be created from repeated morphological closing of the original character with a large kernel until the image is entirely closed. The closed image is subtracted from the original image to get regions defined as holes. The holes are eroded iteratively with a small kernel until it is completely eroded. The filled characters are created from turning background pixels at the boundary of the holes and the character from white to black. The level of broken degradation is measured by the percentage of foreground pixels in the degraded broken image relative to the number of foreground pixels in the original image. The filled degradation level is the percentage of foreground pixels added relative to the total number of pixels that can be filled.

Baird [9] proposed a parameterized model that looks at the whole document creation process, including both typesetting and scanning. Some degradations are global for the page, and some are local to the character or even the pixel. This model has a comprehensive set of deformation variables. Parameters that relate to the document formatting or typesetting include size, skew, xscale, yscale, xoffset, and yoffset (Table 2.6). To consider the defects introduced in acquired images that are related to the scanning process, as discussed in the section “[Acquisition Methods](#)” (Fig. 2.11), the parameters resolution, blur, threshold, sensitivity, and

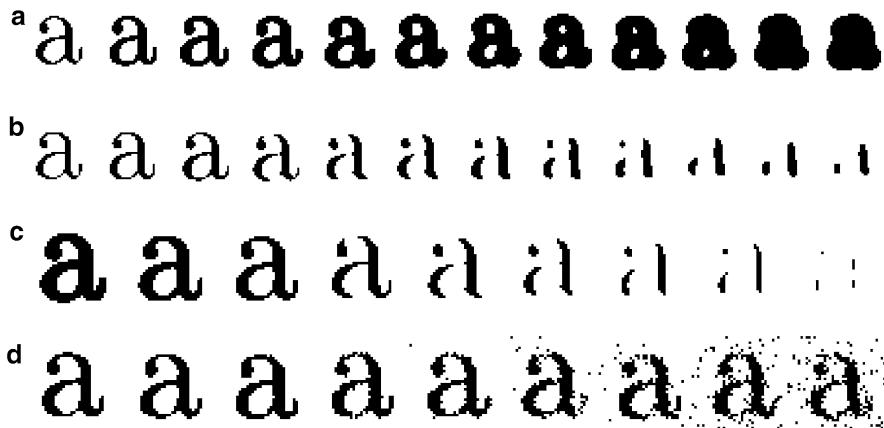
**Table 2.6** Summary of Baird degradation model parameters

	Parameter	Description
<b>Creation</b>	Size	Text size in points
	Skew	The amount of rotation from horizontal of each character
	xscale and yscale	Multiplicative scaling factors to adjust the size of the character from its nominal value
	xoffset and yoffset	Translations of the individual characters with respect to the baseline and the sampling grid. These could also be considered sampling phases in acquisition
<b>Acquisition</b>	Resolution	A combination of the character size in points and the scanning resolution
	Blur	The size of the PSF, which Baird models as the standard deviation of a Gaussian filter
	Threshold	The intensity level which determines whether the pixel will be black or white
	Sensitivity	Noise amount added to the level at each sensor before thresholding
	Jitter	A random offset of the pixel centers from a square grid

jitter are included. Baird included a likely range and distribution for each parameter associated with each degradation in the model which he used to generate characters for OCR experiments. Blur, threshold, and sensitivity control the appearance of the individual character (Fig. 2.18). These parameters were determined by Ho and Baird [10] to be the most related to the accuracy of an OCR system. One approach to getting the “best quality” image acquisition for best OCR accuracy could be to configure the acquisition system to a PSF width and binarization threshold that yield lower OCR error rates.

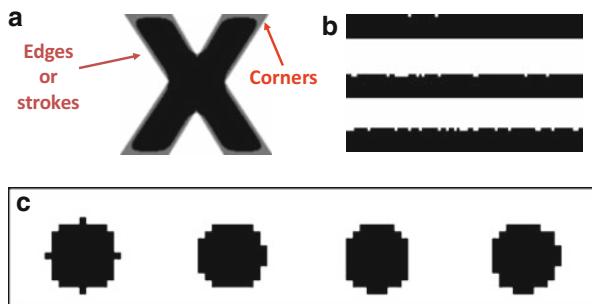
Based on the physical degradation model of blurring with a PSF of width  $w$ , and thresholding at a global threshold of  $\Theta$ , plus additive Gaussian noise with a standard deviation  $\sigma_{\text{noise}}$ , models to describe and quantify three resulting degradations have been developed by Barney Smith et al. [11, 12]. Edges between black and white such as character borders will relocate on the page as a function of the width of the PSF and the binarization threshold (Fig. 2.19a). If the edge spread is positive, strokes will be thicker, and when it is negative strokes will become thinner. If it is too negative, strokes will vanish, and if it is too positive, holes can fill and neighboring characters can touch, Fig. 2.15. Corners will be rounded by the scanning and binarization process. Because of the corner erosion and the possibilities of strokes or spaces totally vanishing, inverting the degradation to return the characters to their original states is not possible.

The noise that is present in gray-level images can be characterized by the standard deviation of the additive noise. If the image is thresholded after noise is added, the standard deviation of the noise does not do a good job of characterizing its effect on the binarized image (Fig. 2.19b). The metric *noise spread* describes the noise present in a quantitative measure that also qualitatively describes the noise effect. Noise spread takes into account the blur and threshold.



**Fig. 2.18** Examples of characters produced by the Baird degradation model. (a) Varying PSF width with low threshold (b) medium threshold, (c) varying threshold, and (d) varying additive noise (sensitivity)

**Fig. 2.19** (a) Common degradations from scanning without noise include edge spread and corner erosion; (b) edge noise is also common. It can be measured better through noise spread (*NS*) which is different for these samples than through noise variance  $\sigma_{noise}$  which is the same for these samples. (c) One image sampled at four different phases



Photocopying and facsimile involve steps of image digitization and image printing, both steps introduce additional distortions, Fig. 2.9. Some of the printing noise can be accounted for in the additive noise in the scanner model, so photocopying can be modeled by repeated application of the scanner model.

Since the Nyquist criterion does not hold for document images, sampling during acquisition plays a role in determining the resulting digital image and what information it will contain. The location of the character image relative to the sampling grid can cause significant differences in the bi-level outcome of the digitization process (Fig. 2.19c). As a result, the edge pixel values do not vary independently of each other. Sarkar [13] developed a model of the random-phase sampling aspect of the physical scanning process. He showed the effect that the random-phase sampling has on the character images produced after digitization.



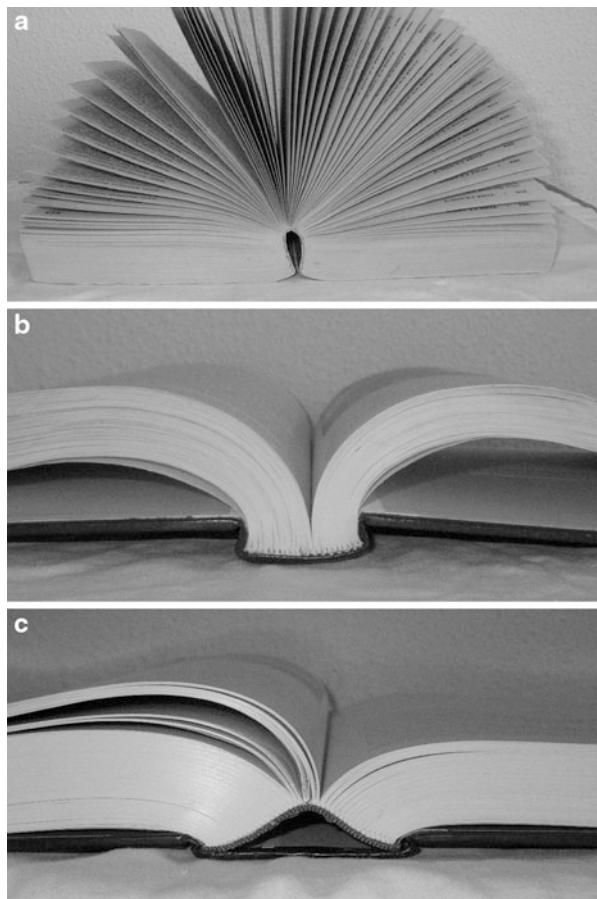
**Fig. 2.20** Examples of characters produced by the Kanungo degradation model. Effects of changing (a) foreground transition probability, (b) background transition probability, (c) base transition probability, and (d) diameter of morphological closing operator. Default parameters are  $[\eta, \alpha_0, \alpha, \beta_0, \beta, k] = [0, 1, 1.5, 1, 1.5, 5]$

The degradations seen in digitized characters can also be mimicked without the model having a connection to the physical creation or acquisition process. Kanungo et al. [14] developed a morphological model for local distortion. The degradation model has six parameters:  $\eta$ ,  $\alpha_0$ ,  $\alpha$ ,  $\beta_0$ ,  $\beta$ , and  $k$ . These parameters are used to degrade an ideal binary image. In this noise model, the distance of a pixel from the boundary of the template character determines its probability of changing from black to white or from white to black. The background pixels are flipped with probabilities determined by the parameters  $\beta$  and  $\beta_0$ , with probability decreasing as the pixels get further from the edge of the character. Likewise, the foreground pixels are flipped with probabilities determined by the parameters  $\alpha$  and  $\alpha_0$ . After bit flipping, the character is processed by a morphological closing operation with a disk structuring element of diameter  $k$  to fill in some of the holes in the character and to account for the correlation present in the scanning process. This model can be used to produce degraded characters that have a wide range of parameterized degradations. Samples of characters generated through this model with a range of parameters are shown in Fig. 2.20.

### Show-Through

If the paper used for printing is thin, or some of the light source comes from the back side of the paper, the printing on the verso side can become visible on the recto side (Fig. 2.14d). Show-through models are used with images of both the observed verso and recto to estimate the corresponding original images. Nominally, a scaled version of the ideal verso side is added to the intensity of the image on the recto side [15, 16]. Variations on this basic idea have been developed to consider the optical

**Fig. 2.21** Examples of (a) flexible spine, (b) fast spine, and (c) hollow spine bindings



blurring or the optical diffusion the paper caused to the verso image before it is visible on the recto. Some models operate on the reflectivity of the verso image and some on the optical density of the ideal recto and verso sides. A model that utilizes more detailed physics of the process was developed by Khan and Hasan [17]. This includes the effect of the light of the scanner penetrating the paper and bouncing off the usually white reflecting surface of the scanner top.

### Book Binding

Finally, there are models addressing how book binding affects document image analysis during acquisition. There are three main types of bindings for books: *flexible spine*, *fast spine*, and *hollow spine* (Fig. 2.21). When imaged on a flatbed scanner, the paper near the spine will not sit as close to the glass where the focal point of the imaging system lies and will not receive as much light. This causes shadows and out of focus problems as well as varying curvature of the text lines (Fig. 2.22). Kanungo et al. [14] modeled this defect based on the underlying perspective geometry of the optical system. He assumed the curved part was circular

**a**

ured  $\tau_1$  value to the  $\tau_1$  value  
placement curve will yield

$$(4.22)$$

$$(4.23)$$

to determine  $\Theta$ :

$$(4.24)$$

and  $\Theta$ , for which the theoretical  $f_{r_{meas}}(\tau)$ , data. Best fit is

$$1)^2 . \quad (4.25)$$

4.2.1. Then the parametric progression. As described in a form for which a unique figure 4.9. Within the shaded region, there is a single scan-line constant. In the test target 4.9, this corresponds to a

pend on the value of  $\Theta$  and functional forms depends on the horizontal slice, the MSE

Image Processing and Analysis

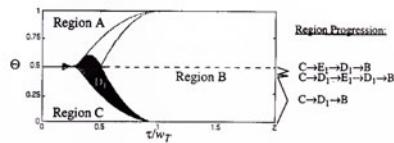


Figure 4.9: Fraction regions for triangular PSF.

the functional form of region  $E_1$ , then to the functional form of region  $D_1$  and finally to the functional form of region  $B$ . For  $\Theta$  values between these two ranges, the progression would be  $C \rightarrow D_1 \rightarrow E_1 \rightarrow D_1 \rightarrow B$ . A similar set of progressions occur for  $0.5 < \Theta < 1.0$ .

To fit the parametric fraction to the data, assumptions about which of these six categories of progressions is valid must be made. Then the location of the boundaries between regions with respect to the data must be hypothesized. Each data region will be tested separately by regression to find the best parameters. The partition location whose estimate of  $(w_T, \Theta)$  gives the minimum MSE is considered correct and the corresponding  $(w_T, \Theta)$  estimates are designated to be the correct parameter values.

The regions F, G, etc. (regions where  $0 < r(\tau) < 1$  and  $t/w_T < 1/4$ ) have been eliminated in the implementation because they are very small and the effect of the star being in two dimensions is likely to obscure the data in these regions any ways. If necessary, they could be added to the implementation.

#### 4.2.4 Implementation of Estimation from Wedges

The final estimation method uses isolated wedges. The location of the two straight edges and the apex of the blurred wedge must be found from the image and together they determine the PSF width and threshold value.

The pixels denoting the top and bottom edges of the wedge are identified by applying a MATLAB contour function to the image and these coordinates are fit to straight

**b**

ured  $\tau_1$  value to the  $\tau_1$  value  
placement curve will yield

$$(4.22)$$

$$(4.23)$$

to determine  $\Theta$ :

$$(4.24)$$

and  $\Theta$ , for which the theoretical  $f_{r_{meas}}(\tau)$ , data. Best fit is

$$1)^2 . \quad (4.25)$$

4.2.1. Then the parametric progression. As described in a form for which a unique figure 4.9. Within the shaded region, there is a single scan-line constant. In the test target 4.9, this corresponds to a

pend on the value of  $\Theta$  and functional forms depends on the horizontal slice, the MSE

Image Processing and Analysis

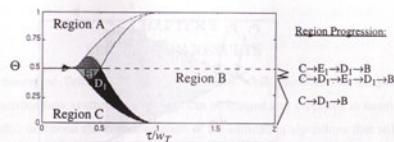


Figure 4.9: Fraction regions for triangular PSF.

the functional form of region  $E_1$ , then to the functional form of region  $D_1$  and finally to the functional form of region  $B$ . For  $\Theta$  values between these two ranges, the progression would be  $C \rightarrow D_1 \rightarrow E_1 \rightarrow D_1 \rightarrow B$ . A similar set of progressions occur for  $0.5 < \Theta < 1.0$ .

To fit the parametric fraction to the data, assumptions about which of these six categories of progressions is valid must be made. Then the location of the boundaries between regions with respect to the data must be hypothesized. Each data region will be tested separately by regression to find the best parameters. The partition location whose estimate of  $(w_T, \Theta)$  gives the minimum MSE is considered correct and the corresponding  $(w_T, \Theta)$  estimates are designated to be the correct parameter values.

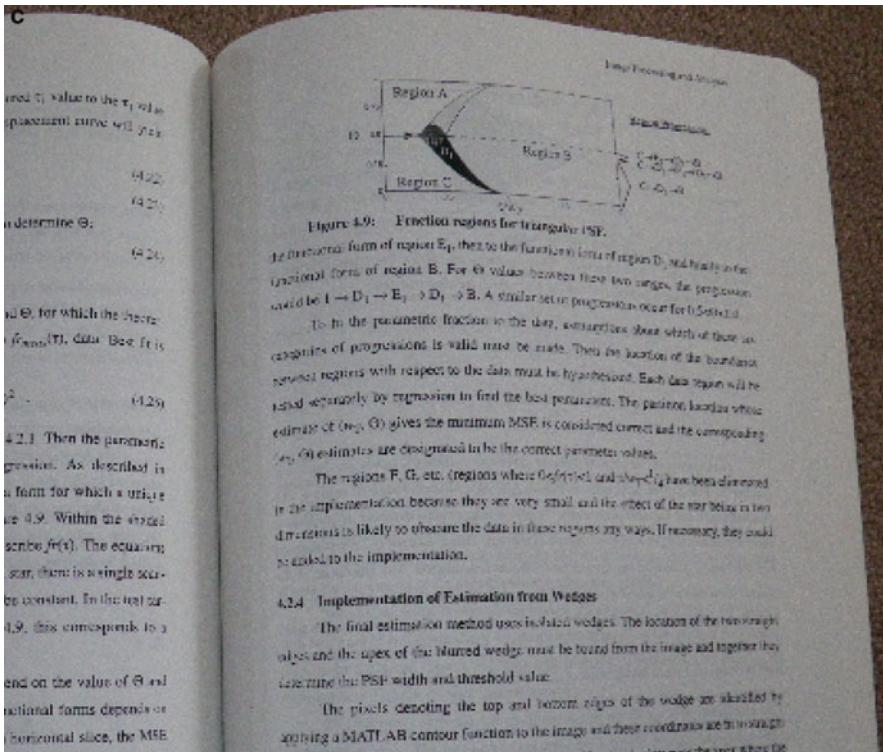
The regions F, G, etc. (regions where  $0 < r(\tau) < 1$  and  $t/w_T < 1/4$ ) have been eliminated in the implementation because they are very small and the effect of the star being in two dimensions is likely to obscure the data in these regions any ways. If necessary, they could be added to the implementation.

#### 4.2.4 Implementation of Estimation from Wedges

The final estimation method uses isolated wedges. The location of the two straight edges and the apex of the blurred wedge must be found from the image and together they determine the PSF width and threshold value.

The pixels denoting the top and bottom edges of the wedge are identified by applying a MATLAB contour function to the image and these coordinates are fit to straight

**Fig. 2.22** (continued)



**Fig. 2.22** Examples images of bound books taken with (a) a bi-level flatbed scanner, (b) a gray-level flatbed scanner, and (c) a camera

with a radius  $\rho$  and calculated the amount of bending of all parts of the image, which was followed by a perspective distortion, nonlinear illumination, and nonlinear blurring to account for out of focus. This can be used to correct the illumination and straighten out the lines of text.

## Conclusion

Over time, many methods and materials have been developed and used to create documents. Each offered an advantage based on current technology and available materials, and each produces a resulting document image that will have different characteristics. If the best combination of materials is used, good quality documents can be created. The recognition of the content of a document depends heavily on the processes of how a document was created. To begin document analysis, digital images of the physical document will be acquired. The acquisition process produces an image which can contain many image features from the document but can also introduce undesired artifacts. When those image characteristics and acquisition artifacts are undesired, they can decrease document analysis performance. Recognition

methods are developed based on what is expected from the document generation and acquisition stages. Models have been created to try and understand the degradations and to provide tools to increase performance of future document analysis algorithms.

---

## References

1. ISO/IEC 12233:2000, Photography – Electronic still-picture cameras – Resolution measurements (2000)
2. ISO/IEC 13660:2001 Information Technology – Office equipment – Measurement of image quality attributes for hardcopy output – Binary monochrome text and graphic images (2001)
3. Rice SV, Nagy G, Nartker T (1999) Optical character recognition: an illustrated guide to the frontier. Kluwer Academic, Boston
4. Cannon M, Hochberg J, Kelly P (1999) Quality assessment and restoration of typewritten document images. *Int J Doc Anal Recognit* 2:80–89
5. Souza A, Cheriet M, Naoi S, Suen CY (2003) Automatic filter selection using image quality assessment. In: Proceedings international conference on document analysis and recognition, Edinburgh, pp 508–511.
6. Franke K, Rose S (2004) Ink-deposition model: the relation of writing and ink deposition processes. In: Proceedings of the 9th international workshop on frontiers in handwriting recognition (IWFHR-9 2004), Kokubunji
7. Norris M, Barney Smith EH (2004) Printer modeling for document imaging. In: Proceedings of the international conference on imaging science, systems, and technology (CISST'04), Las Vegas, 21–24 June 2004, pp 14–20
8. Obafemi-Ajayi T, Agam G (2012) Character-based automated human perception quality assessment in document images. In: IEEE Trans Syst Man Cybern A Syst Hum 42(3):584–595
9. Baird HS (1990) Document image defect models. In: Proceedings of the IAPR workshop on syntactic and structural pattern recognition, Murray Hill, 13–15 June 1990
10. Ho TK, Baird HS (1997) Large-scale simulation studies in image pattern recognition. IEEE Trans Pattern Anal Mach Intell 19(10):1067–1079
11. Barney Smith EH (2005) Document scanning. McGraw-Hill 2005 yearbook of science and technology. McGraw-Hill, New York/London
12. Barney Smith EH (2009) A new metric describes the edge noise in bi-level images. SPIE Newsroom. doi:10.1117/2.1200910.1829
13. Sarkar P, Nagy G, Zhou J, Lopresti D (1998) Spatial sampling of printed patterns. IEEE Trans Pattern Anal Mach Intell 20(3):344–351
14. Kanungo T, Haralick RM, Phillips I (1994) Nonlinear local and global document degradation models. *Int. J Imaging Syst Technol* 5(4):220–30
15. Sharma G (2001) Show-through cancellation in scans of duplex printed documents. IEEE Trans Image Process 10(5):736–754
16. Tonazzini A, Salerno E, Bedini L (2007) Fast correction of bleed-through distortion in grayscale documents by a blind source separation technique. *IJDAR* 10(1):17–25
17. Khan MR, Hasan MK (2010) A novel model for show-through in scan of duplex printed documents. *Signal, Image Video Process*, 6(4):625–645

## Further Reading

### Document Creation Materials: Paper, Pens, and Inks

- Nickell J (1990) Pen, ink & evidence: a study of writing and writing materials for the penman, collector and document detective. University Press of Kentucky
- Whalley JI (1975) Writing implements and accessories. Gale Research, Detroit

### Printing

- Kipphan H (2001) *Handbook of print media*. Springer, Heidelberg
- Williams EM (1993) *The physics and technology of xerographic processes*. Krieger
- Proudfoot WB (1972) *The origin of stencil duplicating*. Hutchinson, London
- Robert U (2000) A review of halftoning techniques. In: *Color imaging: device-independent color, color hardcopy, and graphics arts V*. Proceedings of SPIE, vol 3963, pp 378–391

### Imaging Methods

- Trussell HJ, Saber E, Vrheil M (2005) Color image processing. *IEEE Signal Process Mag*, Jan 2005, 14–22

### Displays

- Heikenfeld J (2010) Light, brite displays. *IEEE Spectrum*, Mar 2010, NA-28-33

### Image Quality

- ISO/IEC 12233:2000 (2000) Photography – electronic still-picture cameras – resolution measurements
- ISO/IEC 13660:2001 (2001) Information technology – office equipment – measurement of image quality attributes for hardcopy output – binary monochrome text and graphic images
- Ho TK, Baird HS (1997) Large-scale simulation studies in image pattern recognition. *IEEE Trans Pattern Anal Mach Intell* 19(10):1067–1079
- Rice SV, Nagy G, Nartker T (1999) Optical character recognition: an illustrated guide to the frontier. Kluwer Academic
- Cannon M, Hochberg J, Kelly P (1999) Quality assessment and restoration of typewritten document images. *Int J Doc Anal Recognit* 2:80–89
- Souza A, Cheriet M, Naou S, Suen CY (2003) Automatic filter selection using image quality assessment. In: *Proceedings of the international conference on document analysis and recognition*, pp 508–511

### Models

- Franke K, Rose S (2004) Ink-deposition model: the relation of writing and ink deposition processes. In: *Proceedings of the 9th international workshop on frontiers in handwriting recognition (IWFHR-9 2004)*
- Norris M, Barney Smith EH (2004) Printer modeling for document imaging. In: *Proceedings of the international conference on imaging science, systems, and technology: CISST'04*, Las Vegas, 21–24 June 2004, pp 14–20
- Obafemi-Ajayi T, Agam G (2012) Character-based automated human perception quality assessment in document images. *IEEE Trans Syst Man Cybern A-Syst Hum* 42(3):584–595
- Baird HS (1990) Document image defect models. In: *Proceedings of the IAPR workshop on syntactic and structural pattern recognition*, Murray Hill, 13–15 June 1990
- Kanungo T, Haralick RM, Phillips I (1994) Nonlinear local and global document degradation models. *Int J Imaging Syst Technol* 5(4):220–230
- Sarkar P, Nagy G, Zhou J, Lopresti D (1998) Spatial sampling of printed patterns. *IEEE Trans Pattern Anal Mach Intell* 20(3):344–351
- Barney Smith EH (2005) Document scanning. McGraw-Hill 2005 yearbook of science and technology. McGraw-Hill
- Barney Smith EH (2009) A new metric describes the edge noise in bi-level images. *SPIE Newsroom*, Nov 2009. doi:10.1117/2.1200910.1829
- Sharma G (2001) Show-through cancellation in scans of duplex printed documents. *IEEE Trans Image Process* 10(5):736–754
- Tonazzini A, Salerno E, Bedini L (2007) Fast correction of bleed-through distortion in grayscale documents by a blind source separation technique. *IJDAR* 10:17–25
- Khan MR, Hasan MK (2010) A novel model for show-through in scan of duplex printed documents. *Signal Image Video Process*, 1–21

---

# The Evolution of Document Image Analysis

3

Henry S. Baird and Karl Tombre

## Contents

Introduction.....	64
Isolated Character Recognition.....	64
Beyond Recognition of Isolated Characters: Exploitation of Context.....	66
From Words to Pages, from Pages to Structured Documents, and Onwards to Non-textual Documents.....	67
Stubborn Obstacles to Document Image Recognition.....	68
Conclusion.....	69
Cross-References.....	69
Notes.....	70
References.....	70
Further Reading.....	71

---

### Abstract

One of the first application domains for computer science was Optical Character Recognition. At that time, it was expected that a machine would quickly be able to read any document. History has proven that the task was more difficult than that. This chapter explores the history of the document analysis and recognition domain, from OCR to page analysis and on to the open problems which are still to be completely dealt with.

---

### Keywords

Document image analysis • History • OCR

---

H.S. Baird

Computer Science and Engineering Department, Lehigh University, Bethlehem, PA, USA  
e-mail: [baird@cse.lehigh.edu](mailto:baird@cse.lehigh.edu)

K. Tombre

Université de Lorraine, Nancy, France  
e-mail: [Karl.Tombre@loria.fr](mailto:Karl.Tombre@loria.fr); [Karl.Tembre@univ-lorraine.fr](mailto:Karl.Tombre@univ-lorraine.fr)

## Introduction

The first computers and computer science as a field emerged from World War II. Once this field extended beyond scientific computing and defense applications, one of its first uses was Optical Character Recognition (OCR). At that time, it was expected that a machine would soon be able to read any document. But it quickly became evident that progress would be slower than expected and that the vast diversity of applications and needs would make it impossible to just rely on improved scores for single-character recognition. Every error in a single digit for a ZIP code can send a letter to the wrong destination, five character errors on a page of text are often enough to add a significant cost of post-OCR editing, and when the problem comes to extracting the major information relayed by a complete document, without knowing the typewriting font or when it is handwritten, the challenge becomes much more one of analyzing the global document “scene” than to decipher single characters.

This vast diversity of document analysis problems, almost none of which could be automatically analyzed with the state of the art, led to extreme specialization of the systems and to a slow pace of development for new applications [1].

This chapter does not pretend to provide a complete history of how document analysis systems evolved, but points out some of the main landmarks in this evolution.

---

## Isolated Character Recognition

Schantz, in his history of OCR [1], mentions an 1809 US patent for assisting reading by the blind. By 1870, a mosaic of photocells was used by C. R. Carey to transmit images, and by the turn of the 20th century, P. Nipkow used “scanning disk” to analyze images line by line. In 1912 Emmanuel Goldberg patented a machine to convert typed messages to telegraphic messages, and in 1914 Edmund F. D’Albe invented a handheld scanner that converts images of text into sounds to assist blind readers. By 1931 Goldberg patented an optical scanner driving a template-matching character classifier. Accuracy of this and many similar technologies was limited, until the 1950s, by “difficulties in precisely aligning source patterns with template patterns.”

In 1951 David Shepard demonstrated an OCR machine able to read 23 letters typed by a “standard typewriter.” Around the same time Jacob Rabinow refined template matching to search among a range misregistrations etc. for a “best match” and extended the alphabet to both upper and lower case. In the early 1950s advances seemed to depend, principally, on better imaging sensors and controlling movement of sensors and paper. Recognition methods included “area correlation, feature analysis, [and] curve tracing”; refinements included “noise filtering [and] image preprocessing.” Applications spread rapidly in the late fifties; most were custom

designed: in 1959, a machine built for the US Air Force could read both upper and lower case alphanumerics – but only in a single typeface. Even typewriter fonts posed a challenge in 1954 an OCR manufacturer proposed widespread standardization of printing using a specially designed  $5 \times 7$  grid typeface, in order to assist OCR.

In the 1960s typewriting applications spread rapidly even though each was typically custom trained for the application and could handle only one typeface; still, it was generally believed that the technology could be trained to usefully high accuracy on “almost any consistently formed character patterns.” OCR machines could read only a single format of document in a batch: users were expected to standardize their input. Soon customers were asking for machines that could handle a variety of documents including a variety of typefaces. The first commercial “multifont” machine appeared in 1964; by the late 1960s such a machine, custom built for the US Army, “read 63 % [...] error-free” of a highly heterogeneous input stream of documents containing upper and lower case letters in over 30 pretrained fonts. Through the 1960s all OCR machines were large custom installations handling large batches, often at very high speeds.

The early 1960s saw the first promising experiments on hand-printed (not cursive) characters. In the late 1960s, user-trainable OCR machines appeared, which were then marketed as (potentially) “omnifont”; but these were seldom effective. The technical obstacles facing the technology are reflected clearly in this proposal by Rabinow in [2]:

The more control one puts into a document, the simpler and less costly is the reading machine. . . . [How can this be done?] Standardize the type of paper, [...] the size of paper, [...] the quality of printing, [...] the format, and [...] the font.

The industry echoed this cry, and the two standard OCR fonts (OCR-A and OCR-B) resulted.

By the late 1970s, character readers were complemented by other control-intensive technologies including bar code readers and mark sense readers. The early 1980s saw a significant change in the market as fax machines spread, and higher-resolution flat-bed document scanners became affordable, and these were connected to personal computers. Then, OCR companies raced to deliver “personal OCR” where pre-training and most kinds of “control” were missing. Through massive training on large collections (of tens of millions) of character images from scores of typefaces, OCR companies tried to fulfill their claim of truly “omnifont” recognition systems.

In 1992, Mori et al. presented a historical review of these early years in OCR Research and Development [3]. At the same time, George Nagy [4] prophetically criticized as “exhausted” the then state of the art which relied on accurate recognition of isolated character images, and pointed to the promise of exploiting larger contexts including style consistency within documents, and broader multicharacter contextual analysis including layout context. Mori, Nishida, and Yamada [5] later summarized the state of the art in 1999 of approaches to isolated character recognition.

## Beyond Recognition of Isolated Characters: Exploitation of Context

As pointed out by Nagy, it was necessary to go beyond progress in recognition rates for isolated characters if machines were expected to read like human beings do. As a matter of fact, we humans do not only learn the individual letters, in first grade; we also learn to read and understand complete texts, to extract meaningful information from forms, and to communicate with each other through sophisticated documents such as accountancy reports, news articles, poetry, or even maps and engineering drawings. Even handwritten documents where any single character may be extremely hard to decipher become meaningful because we take the context of the document into account.

This starts by looking at a typewritten, printed, or handwritten document like a message using a *language* known to both emitter and receiver. Thus, the analysis of a written object can take into account linguistic aspects.

Sampson, in his seminal study *Writing Systems* [6] noted that

although the tide is beginning to turn now [1985], for most of the twentieth century linguistics has almost wholly ignored writing.

Thus, serious academic attention to the linguistics of writing is very recent, contemporary with the appearance of OCR machines designed for “general-purpose” use by nonexperts. Within the academic linguistics community, computational approaches have also been very much in the minority through the 1970s; even today, it can be difficult for an OCR researcher to locate counterparts in the linguistics community who are willing and able to share their insights in readily usable forms of data and software. We believe that this fact, in turn, has significantly slowed the exploitation of knowledge discovered by linguists with document image recognition technology.

One of the earliest steps in automating natural language text is to provide a means for checking the legality of words. The simplest such means is surely a more or less exhaustive list, or flat lexicon. The earliest exploitation of dictionary context in OCR systems relied on such lists and continues to do so up to the present. The collection of computer-legible lexica accelerated rapidly in the 1970s and is now nearing saturation for languages supported by a robust information technology industry; but, as always, many remote languages are underserved, and thus the extension of modern OCR systems, designed to be cheaply retargeted to new languages by providing a lexicon, may hit a significant barrier.

Many languages however are highly *inflected* so that a large number of variations of words occur: the characteristics common to them all is sometimes called the stem word, and the variations are often provided by suffixes, prefixes, and more complex rewritings. Latin, Spanish, and Russian are extreme cases. For most of these languages, it is possible to capture all or most of the *inflectional morphology* rule within a computational linguistics algorithm, which offers several benefits:

1. **Smaller lexicons**, since many variations collapse into the same rule.
2. **Ease of entering new words**, since with the addition of only the stem of a new word, all its inflections are covered.

3. **Recognition of neologisms**, so that words never seen before can be correctly identified (by derivational morphology).
4. **Faster lookup** is a possibility, in spite of computational overhead, in cases where an equivalent lexicon is unmanageably huge.

All of these benefits are potentially enjoyed by an OCR system.

Ritchie et al. [7] presented a nearly exhaustive analysis of such morphological structure of English words, which requires a “two-level” system of regular grammar rewriting rules. They mention that this approach works on languages including Finnish, French, German, Japanese, Rumanian, Old Church Slavonic, and Swedish. Unfortunately, Semitic languages such as Hebrew and Arabic possess “non-concatenative” morphologies which require more advanced models. An implication for OCR systems is that linguistic contexts as elementary as “dictionary checks” may not be feasible even today for underserved languages, and making progress may require professional linguistic effort or even research in linguistics.

---

## From Words to Pages, from Pages to Structured Documents, and Onwards to Non-textual Documents

One thing is to work on recognizing characters, words, or sentences; another is to get access to all the information present in a document such as a letter to be handled by the postal service, a bank check, a completed form, or a business letter. Beyond character and word recognition, this includes numerous tasks, especially related to the spatial analysis of the document page, which is actually a scene analysis problem, and the mapping between the layout structure and the semantics conveyed by this layout.

Early work in that area addressed the most common layouts. The layout in rectangular shapes, which can be found in books, newspapers, journals, etc., was extracted through various methods designed by research teams in the 1980s. Methods such as the run length smoothing algorithm designed at IBM [8] and used in systems analyzing newspaper archives [9], or the X-Y tree which decomposed a journal article into homogeneous parts [10], are still widely used nowadays, as explained in ▶Chap. 5 (Page Segmentation Techniques in Document Analysis).

Specific classes of documents, where the layout and/or syntactical constraints are strong and well known, and the need for reliability on large volumes of documents is high, were also given special attention very early. Thus, systems were designed for postal automation [11] or bank check recognition (see ▶Chap. 21 (Document Analysis in Postal Applications and Check Processing) for a history of this field), tables and forms (see ▶Chap. 19 (Recognition of Tables and Forms)), or business letters.

It also became necessary to look beyond text, as documents in the most general meaning are formalized ways for humans to communicate with each other, using a commonly understood language, which can also include graphical parts, images, etc. This led to work on systems for the analysis of maps [12, 13], of electrical diagrams [14], or of engineering drawings [15]. If these early systems were often limited, fine-tuned for a narrow set of documents, and difficult to maintain and to expand,

they still helped in developing basic methods for graphics recognition which are still in use, as detailed in ►Chaps. 15 (Graphics Recognition Techniques), ►16 (An Overview of Symbol Recognition), and ►17 (Analysis and Interpretation of Graphical Documents).

---

## Stubborn Obstacles to Document Image Recognition

In 1982, Schantz said that “the rate of correct character recognition is directly proportional to the quality of source data” [1]. In 1999, Rice, Nagy, and Nartker [16] published a profusely illustrated taxonomy of frequently occurring OCR errors and discussed the origins of these errors with unprecedented insight. One aspect of quality, which turns out to be amenable to formal modeling and systematic engineering attack, is the degradation of the image due to both printing and image capture [17].

Among the many obstacles which are still on the road of the evolution of document image analysis and recognition, let us mention those which seem to us to be the most difficult to deal with and which thus have to be given continuing attention in the coming years.<sup>1</sup>

- (a) Document images are not always captured in an optimal and controlled way, and their quality is often too low. In some cases, such as managing large collections of heritage documents, decisions can have been made on the resolution of the scanning process, and the documents themselves are sometimes degraded. Later processes have to use the images as they are, even when it becomes evident that the quality is far from being adapted to the analysis processes. Other cases where the image quality can lead to specific problems include text in video and documents captured by a camera or a phone (see ►Chap. 25 (Text Localization and Recognition in Images and Video)).
- (b) Many recognition processes rely on classification methods which need to be trained. But it is not always possible to work with large enough sets of training samples, covering the full diversity of the analysis problem. This is especially true for non-textual documents. Related to that, for the purpose of evaluating the performances of document analysis systems, it is often difficult to have sufficiently ground-truthed data. See ►Chaps. 29 (Datasets and Annotations for Document Analysis and Recognition) and ►30 (Tools and Metrics for Document Analysis Systems Evaluation) for further discussions of these problems.
- (c) We have seen that linguistic tools are an important asset in designing efficient document analysis systems. But in many languages, such tools lack or are not sufficiently developed.
- (d) No document analysis system can be made completely automatic, so that it could just work as a post-processing step on the output of the scanner. But it is difficult to construct effective user interfaces, to effectively integrate document image analysis into a larger workflow, even the more because it is not

- always easy to have the user accept the error-prone nature of document image processing and recognition.
- (e) There seems to be infinite ways in which people create documents, with complex layouts or with inconsistent or nonexistent typographical and semantic rules. It is impossible to train a system for all such variations. This has led researchers and companies to focus on small subsets of problems, and the solutions they design are very often not applicable to slightly different problems or document categories.
  - (f) Today, many companies are faced with the problem that their customers or suppliers send documents via multiple channels, in order to communicate messages having a legal or economic relevance. This includes filled forms via printed mail, faxes, scanned document images sent as email, or even electronic documents in PDF or TIFF format, complemented by metadata. Although most of these channels provide a certain amount of metadata (a fax provides a fax number, emails have information in their headers, and electronic documents have full sets of descriptors), they still necessitate addressing the whole range of document analysis problems, as this book largely demonstrates. Moreover, the messages conveyed by the document are an integral part of the workflow, i.e., they may request information or answer such a request. Helping to make this multichannel information directly available to feed the workflow is a challenge, and good solutions to this challenge would have a high economic value.

---

## Conclusion

Sellen and Harper [18] cogently argue that paper's role as a medium for communication is not likely to decline in scale within the foreseeable future, even as purely digital media continue to grow exponentially. Lesk's prophetic study of digital libraries [19] points out that, even as inevitably much modern data will be "born digital" and so never have to be converted from images of documents, the total volume of printed paper will grow alongside the accelerating scale of digital libraries. In Nunberg's vision of the future of the book [20], digital and document-based versions of information will coexist and, assisted by document image analysis technology, refer richly to one another.

---

## Cross-References

- ▶ [An Overview of Symbol Recognition](#)
- ▶ [Analysis and Interpretation of Graphical Documents](#)
- ▶ [Analysis and Recognition of Music Scores](#)
- ▶ [Analysis of the Logical Layout of Documents](#)
- ▶ [Document Analysis in Postal Applications and Check Processing](#)
- ▶ [Graphics Recognition Techniques](#)
- ▶ [Logo and Trademark Recognition](#)

- ▶ Machine-Printed Character Recognition
  - ▶ Page Segmentation Techniques in Document Analysis
  - ▶ Processing Mathematical Notation
  - ▶ Recognition of Tables and Forms
- 

## Notes

<sup>1</sup>Many thanks to the authors of several chapters of this handbook for their contribution to the list of stubborn obstacles.

---

## References

1. Schantz HF (1982) History of OCR, optical character recognition. Recognition Technologies Users Association, Manchester Center, Vt., USA
2. Rabinow J (1969) Whither OCR? And whence? Datamation 15(7):38–42
3. Mori S, Suen CY, Yamamoto K (1992) Historical review of OCR research and development. Proc IEEE 80(7):1029–1058
4. Nagy G (1992) At the frontiers of OCR. Proc IEEE 80(7):1093–1100
5. Mori S, Nishida H, Yamada H (1999) Optical character recognition. Wiley, New York
6. Sampson G (1985) Writing systems. Stanford University Press, Stanford
7. Ritchie G, Russell G, Black A, Pulman S (1992) Computational morphology. MIT, Cambridge
8. Wong KY, Casey RG, Wahl FM (1982) Document analysis system. IBM J Res Dev 26(6): 647–656
9. Wang D, Srihari SN (1989) Classification of newspaper image blocks using texture analysis. Comput Vis Graph Image Process 47:327–352
10. Nagy G, Seth S, Viswanathan M (1992) A prototype document image analysis system for technical journals. IEEE Comput Mag 25(7):10–22
11. Schürmann J (1978) A multifont word recognition system for postal address reading. IEEE Trans Comput 27(8):721–732
12. Kasturi R, Alemany J (1988) Information extraction from images of paper-based maps. IEEE Trans Softw Eng 14(5):671–675
13. Shimotsuji S, Hori O, Asano M, Suzuki K, Hoshino F, Ishii T (1992) A robust recognition system for a drawing superimposed on a map. IEEE Comput Mag 25(7):56–59
14. Groen F, Sanderson A, Schlag J (1985) Symbol recognition in electrical diagrams using probabilistic graph matching. Pattern Recognit Lett 3:343–350
15. Vaxivière P, Tombre K (1992) Celesstin: CAD conversion of mechanical drawings. IEEE Comput Mag 25(7):46–54
16. Rice S, Nagy G, Nartker T (1999) OCR: an illustrated guide to the frontier. Kluwer, Boston
17. Baird H (2007) The state of the art of document image degradation modeling. In: Chaudhuri B (ed) Digital document processing. Springer, London
18. Sellen A, Harper R (2003) The myth of the paperless office. MIT, Cambridge
19. Lesk M (1997) Practical digital libraries: books, bytes, & bucks. Morgan Kaufmann, San Francisco
20. Nunberg G (1996) The future of the book. University of California Press, Berkeley
21. Baird HS, Bunke H, Yamamoto K (eds) (1992) Structured document image analysis. Springer, Berlin/New York
22. Nagy G (2000) Twenty years of document image analysis in PAMI. IEEE Trans PAMI 22(1):38–62

## Further Reading

The beginning of the 1990s was a time when document analysis had already matured substantially, and the research community felt the need for a consolidation of the knowledge gained that far. Several special issues and books were published in that period and are a good start to understand the early years of document image processing and recognition.

- The Proceedings of the IEEE, vol. 80, no. 7, 1992, contained several interesting articles on the history and the state of the art, including Refs. [5] and [4].
- A special issue of IEEE COMPUTER Magazine (vol. 25, no. 7, July 1992) had a number of articles on state of the art document analysis systems, including (but not limited to) [13] and [15].

The early 1990s was also a time when the document image analysis community created its mainstream conference, the International Conference on Document Analysis and Recognition, whose first instance was held in Saint-Malo, France, in 1991. Prior to that, a workshop in New Jersey held in 1990 gave way to a book which also gave a good overview of the state of the art of the field at that time [21].

In 2000, George Nagy published a survey article revisiting 20 years of document image analysis seen through the window of the IEEE Transactions on Pattern Analysis and Machine Intelligence [22].

For review of later evolutions, the reader is referred to the “Further readings” of the different chapters of this book, as the field became very broad, with specializations on many different topics.

---

# Imaging Techniques in Document Analysis Processes

4

Basilis G. Gatos

## Contents

Introduction.....	74
Basic Image Processing Algorithms.....	74
Morphological Operations.....	75
Skeletonization.....	75
Connected Component Labeling.....	76
Run-Length Smoothing Algorithm (RLSA).....	76
Distance Transform.....	76
Hough Transform .....	77
Projection Profiles.....	77
Document Image Binarization.....	77
Global Thresholding Techniques.....	79
Local Thresholding Techniques.....	86
Hybrid Thresholding Techniques.....	90
Combining Different Binarization Techniques.....	91
Using Training Samples.....	92
Binarization of Color Documents.....	94
Document Image Enhancement.....	95
Low Contrast and Uneven Background Illumination.....	95
Bleed-Through, Shining, or Shadow-Through Effects.....	97
Damaged Characters or Noisy Background.....	101
Borders or Parts of Adjacent Page.....	102
Document Image Normalization.....	104
Page Orientation.....	104
Deskew and Deslant.....	112
Dewarping.....	123
Conclusion.....	127
Cross-References.....	127
References.....	128
Further Reading.....	131

---

B.G. Gatos

Institute of Informatics and Telecommunications, National Center for Scientific Research  
“Demokritos”, Agia Paraskevi, Athens, Greece  
e-mail: [bgat@iit.demokritos.gr](mailto:bgat@iit.demokritos.gr)

**Abstract**

Imaging techniques are widely used in document image analysis in order to process, enhance, analyze, and recognize document images. In this chapter, we present an overview of basic image processing algorithms used in document image analysis and focus on the techniques used for document image binarization, enhancement, and normalization.

**Keywords**

Binarization • Border removal • Color reduction • Dewarping • Image enhancement • Morphological operations • Page curl correction • Skew correction

---

## Introduction

Several image processing algorithms are used at different stages of document image analysis process. These include morphological operations, skeletonization, connected component labeling, run-length smoothing, distance calculation, Hough transform, and projection profiles. In this chapter, we present an overview of these algorithms as well as an analysis of the imaging techniques used for document image binarization, enhancement, and normalization.

Document image binarization is used to separate the text from the background regions. It is an important, critical, and, at the same time, difficult and challenging task due to possible image degradations, nonuniform background intensity, low contrast, shadows, smear, etc. Document image enhancement aims to improve the quality of document images by diminishing artifacts such as low contrast and uneven background illumination, bleed-through and shadow effects, damaged characters, and noisy black borders. Document image normalization refers to the task of restoring the document image horizontal alignment after correcting possible page skew, character slant, warping, and perspective distortions.

The organization of this chapter is as follows. Section “[Basic Image Processing Algorithms](#)” presents the basic image processing algorithms used in document analysis process. Section “[Document Image Binarization](#)” gives an overview of document image binarization methods while section “[Document Image Enhancement](#)” presents document image enhancement state-of-the-art techniques. In section “[Document Image Normalization](#),” document image normalization methods are presented. Finally, a summary of the key issues of this chapter is given.

---

## Basic Image Processing Algorithms

Digital image processing is a subcategory of digital signal processing and refers to the use of computer algorithms in order to process digital images. The development

of image processing algorithms started in 1960s and till now has grown considerably with several applications including medical image processing, remote sensing, robot vision, image transmission, and coding. A detailed description of the available image processing algorithms can be found in several survey papers and books [55]. In this section, basic image processing algorithms used in document analysis process will be introduced. These include morphological operations, skeletonization, connected component labeling, run-length smoothing, distance calculation, Hough transform, and projection profiles.

## Morphological Operations

Mathematical morphology was introduced in 1960s for describing the structure of materials by image analysis and is a very popular nonlinear theory for image processing, based on set theory. Originally developed for binary images, it was later generalized for grayscale images. The basic idea in binary morphology is to probe an image with a simple, predefined shape, drawing conclusions on how this shape fits or misses the shapes in the image. The two most basic operations in mathematical morphology are erosion and dilation. These operations work by translating a structuring element to various points in the input image and examining the intersection between the translated kernel coordinates and the input image coordinates. Morphological operators have been used extensively for various machine vision and recognition tasks. In document image analysis, morphological operations have been proved useful for several tasks like image cleaning and noise removal, layout analysis, skew correction, text line finding, and feature extraction.

## Skeletonization

Skeletonization or thinning refers to the core-line detection of images. The purpose of skeletonization is to reduce the image components to their essential information so that further analysis and recognition are facilitated. In document image analysis, skeletonization is commonly used at the preprocessing, segmentation, and feature extraction stages.

Several methods for image skeletonization have been proposed in the literature and can be categorized in the following three broad categories. One category of skeletonization methods is based on distance transforms. Those methods detect all points that correspond to the centers of the maximal discs contained in the given image. A second category of methods produces median or center lines of the digital object in a non-iterative way. Usually, some critical image points are first calculated and then a specific path is defined by connecting these points. A third category of skeletonization methods is characterized by iterative thinning. In every

iteration, every pixel is examined to be removed based on several pixel connectivity rules. The skeleton can be calculated using the basic operations of mathematical morphology which makes the skeleton a morphological representation.

## **Connected Component Labeling**

Connected component labeling is used to assign each image region a unique label, enabling the distinct objects to be distinguished. In binary images, a label is assigned to each image pixel so that connected pixels have the same label. There are two common ways of defining connected pixels for a 2-D image: 4-connected pixels (connected only horizontally or vertically) and 8-connected pixels (connected horizontally, vertically, or diagonally). Connected component labeling is an important step in many image processing applications. In document image processing, it is used mainly at the preprocessing and the segmentation stages.

Connected component labeling algorithms are divided into three broad categories: multi-pass algorithms, two-pass algorithms, and one-pass algorithms.

## **Run-Length Smoothing Algorithm (RLSA)**

In binary images, white runs correspond to successive horizontal or vertical background pixels. RLSA is one of the most popular document imaging techniques and is based on examining the white runs existing in the horizontal and vertical directions. For each direction, white runs whose lengths are smaller than a threshold smoothing value are eliminated [73]. RLSA is usually used for image enhancement and segmentation as well as for object recognition. In document image analysis process, RLSA has been proved very useful mainly for preprocessing, segmentation, and layout analysis.

## **Distance Transform**

According to the distance transform, each pixel is labeled by the shortest distance from the boundary of the region within which it is contained. Distance transform can be used to obtain the thinned image by keeping only points of maximum local distance measures. This thinned image combined with the distance values can be used as a concise and descriptive representation of the original image from which the original image can be reconstructed. Distance transformed is used for a wide range of document image processing tasks including preprocessing, segmentation, and feature extraction.

There are two general approaches to calculate the distance transform. The first is the iterative approach according to which on each iteration boundaries are peeled

from regions and set to the distance from the original boundary. According to the second approach, a fixed number of passes through the image is needed. Usually, two passes following different path directions are sufficient.

## Hough Transform

The Hough transform has emerged in recent decades as a powerful method for many image processing and pattern recognition applications. It involves a transformation from the image coordinate plane to parameter space, and it is useful when the objective is to find lines or curves that fit groups of individual points on the image plane. The purpose of Hough transform is to find imperfect instances of objects within a certain class of shapes by a voting procedure. This voting procedure is carried out in the parameter space, from which object candidates are obtained as local maxima in a so-called accumulator space that is explicitly constructed by the algorithm for computing the Hough transform. In document image analysis, it is mainly used for document skew estimation, text line detection, and feature extraction.

A major drawback of its implementation in large images is its relatively low speed. For the acceleration of the Hough transform, one can select only several characteristic image points or connected component centers in order to calculate the Hough space.

## Projection Profiles

Projection profiles are based on image profiling (projecting the image) in various directions. By calculating the local minima of horizontal and vertical projections, we can define several segments into which the image can be divided. Projection profiles are also used for document skew detection and feature extraction.

---

## Document Image Binarization

Image binarization refers to the conversion of a grayscale or color image into a binary image. In document image processing, binarization is used to separate the text from the background regions by using a threshold selection technique in order to categorize all pixels as text or non-text. It is an important and critical stage in the document image analysis and recognition pipeline since it permits less image storage space, enhances the readability of text areas, and allows efficient and quick further processing for page segmentation and recognition. In the literature, several other terms are also used for document image binarization such as thresholding, text/background separation or segmentation, and background elimination. Binarization has been a subject of intense research in the field of document image processing during the last years. It is a challenging task due to

several difficulties such as image degradations, nonuniform background intensity, low contrast, shadows, and smear.

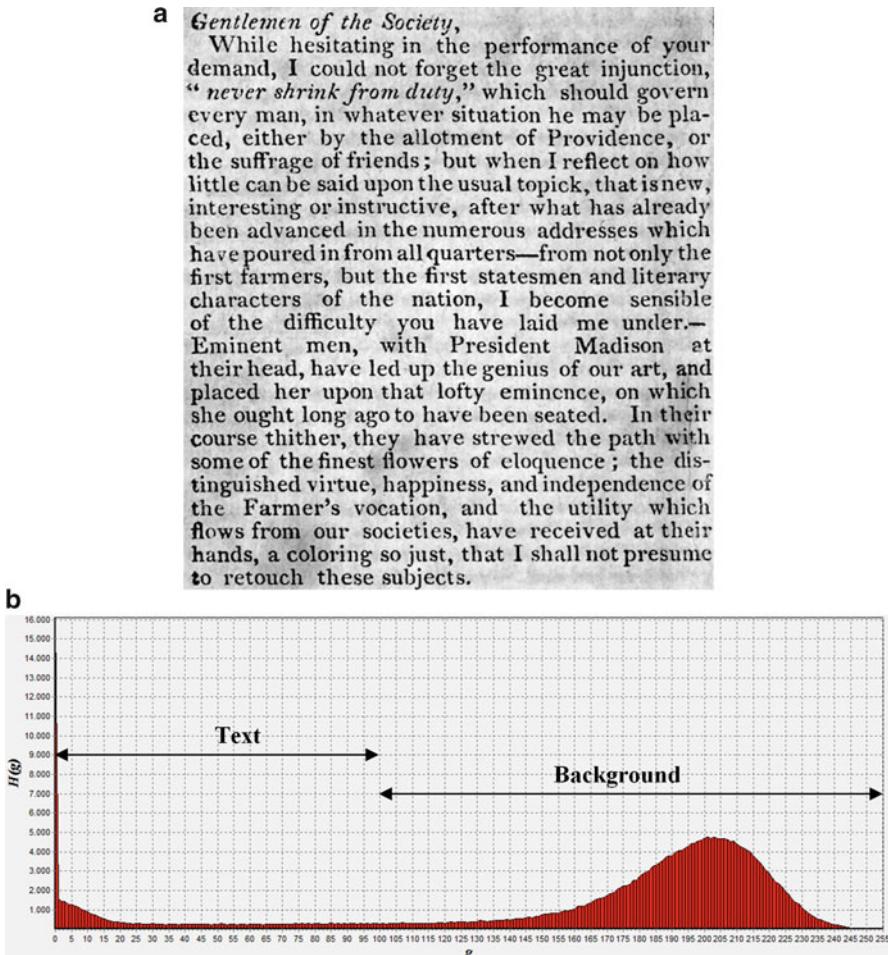
In general, document image binarization approaches are either global or local. In a global approach, threshold selection leads to a single threshold value  $T$  for the entire image. If  $I(x, y)$  is the original grayscale image, then the resulting binary image  $B(x, y)$  is defined as follows:

$$B(x, y) = \begin{cases} 1, & \text{if } I(x, y) \leq T \\ 0, & \text{if } I(x, y) > T \end{cases} \quad (4.1)$$

The grayscale histogram  $H(g)$  of a document image that represents the distribution of the pixels in the image over the gray-level scale is presented in Fig. 4.1 (0 value corresponds to black while 255 to white). It can be observed that in this example, the histogram has a bimodal distribution: the left hill corresponds to text areas while the right hill to the background. In **global thresholding** techniques [16, 54, 61], the task is to calculate the optimal threshold  $T$  in order to separate these two hills. The selection of threshold  $T$  directly affects the quality of the produced binary image. As it is demonstrated in Fig. 4.2, a smaller  $T$  value may lead to broken or faint characters while a larger  $T$  value to noisy or merged characters in the resulting binary image.

Global thresholding has a good performance when there is a good separation between the foreground and the background areas (such as in Fig. 4.1b). If there is an overlap between these two areas, then global thresholding techniques will fail. An example of such a grayscale image is given in Fig. 4.3. In this example, if we apply a small global threshold  $T = 130$  (see Fig. 4.4a), although background is almost completely removed, some text areas are also removed. A larger threshold ( $T = 170$ ) leads to a binary result without missing any text information but with background areas that create a noisy image (see Fig. 4.4c). As we can also see in Fig. 4.4b, a threshold value between the above values ( $T = 150$ ) leads to both missing text information and adding background noisy. In order to solve such problems, **local (adaptive) thresholding** techniques [27, 32, 50, 56, 77] have been introduced. According to these techniques, local area information guides the threshold value for each pixel of the image. These techniques have been widely used in document image analysis because they have a better performance in extracting the character strokes from an image that contains spatially uneven gray levels due to degradations.

There are also some **hybrid methods** [35, 68, 72] proposed for document image binarization that use both global and local information in order to decide if a pixel belongs to text or background category. Recently, it has been proposed [6, 28, 64] to **combine** and take into account the results of a set of binarization techniques in order to use the complementarity in success of each technique. Finally, a special category of document image binarization techniques is based on using **a training set and a machine learning** framework [14, 18, 30] while special techniques are used for the binarization of **color documents** [7, 67, 74] and are based on processing color information. In this section, the most representative works for



**Fig. 4.1** Grayscale histogram demonstration: (a) original grayscale image and (b) the corresponding grayscale histogram  $H(g)$

the above mentioned document image binarization categories will be presented. Table 4.1 presents an overview of key document image binarization techniques. Comprehensive overviews of binarization techniques can be found in several survey papers [57, 66].

## Global Thresholding Techniques

An efficient and frequently used global thresholding technique has been proposed by Otsu [54] and is based on histogram analysis. The threshold operation is regarded as the partitioning of the pixels of an image into two classes  $C_0$  and  $C_1$  where  $C_0$

**a** *Gentlemen of the Society,*

While hesitating in the performance of your demand, I could not forget the great injunction, "*never shrink from duty*," which should govern every man, in whatever situation he may be placed, either by the allotment of Providence, or the suffrage of friends; but when I reflect on how little can be said upon the usual topick, that is new, interesting or instructive, after what has already been advanced in the numerous addresses which have poured in from all quarters—from not only the first farmers, but the first statesmen and literary characters of the nation, I become sensible of the difficulty you have laid me under.—Eminent men, with President Madison at their head, have led up the genius of our art, and placed her upon that lofty eminence, on which she ought long ago to have been seated. In their course thither, they have strewed the path with some of the finest flowers of eloquence; the distinguished virtue, happiness, and independence of the Farmer's vocation, and the utility which flows from our societies, have received at their hands, a coloring so just, that I shall not presume to retouch these subjects.

**b** *Gentlemen of the Society,*

While hesitating in the performance of your demand, I could not forget the great injunction, "*never shrink from duty*," which should govern every man, in whatever situation he may be placed, either by the allotment of Providence, or the suffrage of friends; but when I reflect on how little can be said upon the usual topick, that is new, interesting or instructive, after what has already been advanced in the numerous addresses which have poured in from all quarters—from not only the first farmers, but the first statesmen and literary characters of the nation, I become sensible of the difficulty you have laid me under.—Eminent men, with President Madison at their head, have led up the genius of our art, and placed her upon that lofty eminence, on which she ought long ago to have been seated. In their course thither, they have strewed the path with some of the finest flowers of eloquence; the distinguished virtue, happiness, and independence of the Farmer's vocation, and the utility which flows from our societies, have received at their hands, a coloring so just, that I shall not presume to retouch these subjects.

**Fig. 4.2** (continued)

c *Gentlemen of the Society,*

While hesitating in the performance of your demand, I could not forget the great injunction, “*never shrink from duty*,” which should govern every man, in whatever situation he may be placed, either by the allotment of Providence, or the suffrage of friends; but when I reflect on how little can be said upon the usual topick, that is new, interesting or instructive, after what has already been advanced in the numerous addresses which have poured in from all quarters—from not only the first farmers, but the first statesmen and literary characters of the nation, I become sensible of the difficulty you have laid me under.—Eminent men, with President Madison at their head, have led up the genius of our art, and placed her upon that lofty eminence, on which she ought long ago to have been seated. In their course thither, they have strewed the path with some of the finest flowers of eloquence; the distinguished virtue, happiness, and independence of the Farmer’s vocation, and the utility which flows from our societies, have received at their hands, a coloring so just, that I shall not presume to retouch these subjects.

**Fig. 4.2** Binarization results of the image in Fig. 4.1a: (a) with global threshold  $T = 100$  (a value near the ideal threshold), (b) with  $T = 30$ , and (c) with  $T = 140$

is the foreground (text) and  $C_1$  is the background when global threshold  $t$  is used. In order to measure how good threshold  $t$  is, the discriminant criteria maximizing  $\eta$  are used where  $\eta$  is a separability measure defined as

$$\eta = \frac{\sigma_B^2}{\sigma_T^2} \quad (4.2)$$

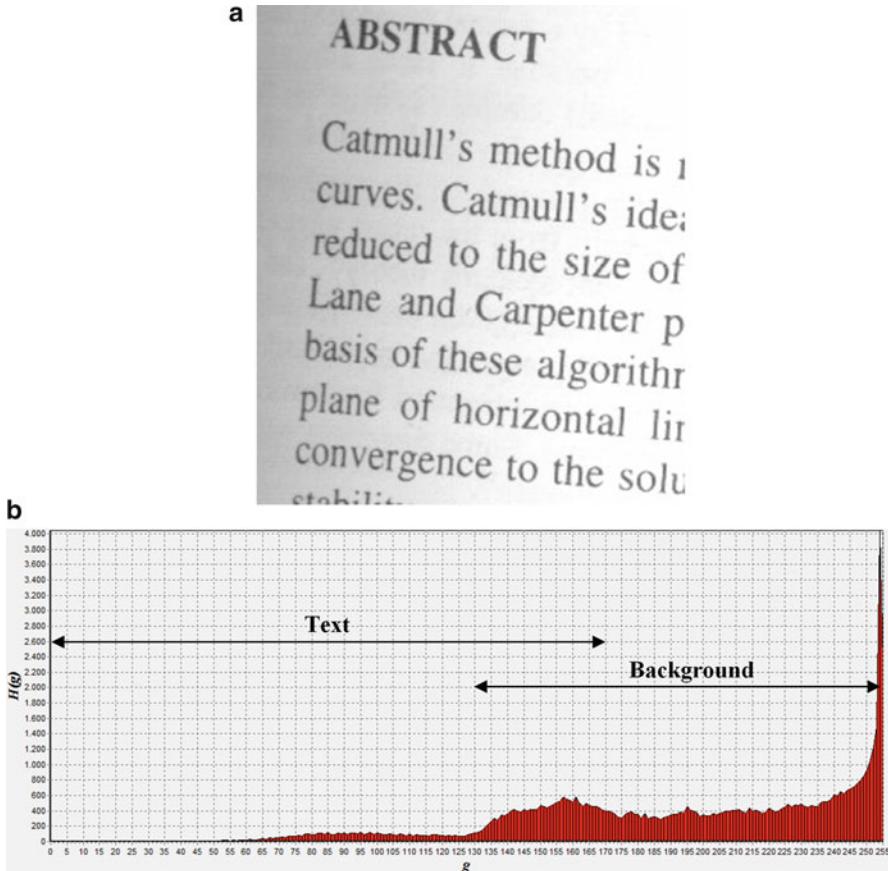
where  $\sigma_B^2$  and  $\sigma_T^2$  are the between-class variance and the total variance, respectively, and

$$\sigma_T^2 = \sum_{i=0}^{255} (i - \mu_T)^2 p_i, \quad \mu_T = \sum_{i=0}^{255} i p_i \quad (4.3)$$

$$\sigma_B^2 = \omega_0 \omega_1 (\mu_1 - \mu_0)^2 \quad (4.4)$$

$p_i$  is the probability of occurrence of gray-level  $i$  and defined as

$$p_i = \frac{H(i)}{N} \quad (4.5)$$



**Fig. 4.3** A grayscale histogram of an image with overlapping text and background areas: (a) original grayscale image and (b) the corresponding grayscale histogram  $H(g)$

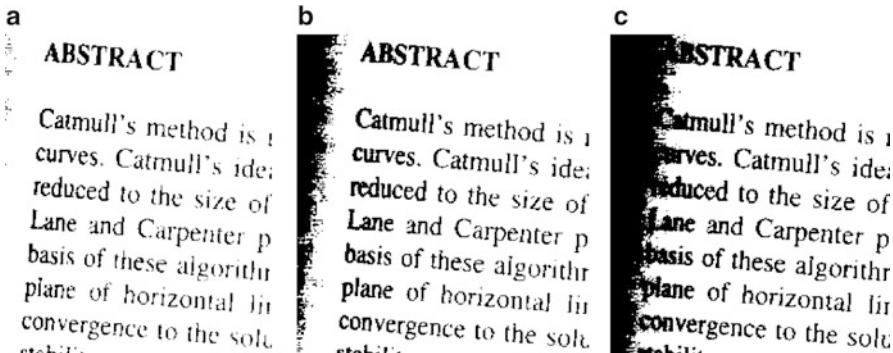
where  $H(i)$  is the grayscale histogram and  $N$  the total number of pixels.  $\omega_0$ ,  $\omega_1$ ,  $\mu_0$ , and  $\mu_1$  are defined as follows:

$$\omega_0 = \sum_{i=0}^t p_i, \quad \omega_1 = 1 - \omega_0, \quad \mu_0 = \frac{\sum_{i=0}^t i p_i}{\omega_0}, \quad \mu_1 = \frac{\sum_{i=t+1}^{255} i p(i)}{\omega_1} \quad (4.6)$$

The optimal global threshold  $t_{\text{opt}}$  is defined as

$$t_{\text{opt}} = \arg \max_{t \in [0..255]} \eta \quad (4.7)$$

where  $\eta$  is defined by Eq. (4.2).



**Fig. 4.4** Binarization results of the image in Fig. 4.3a: (a) with global threshold  $T = 130$ , (b) with  $T = 150$ , and (c) with  $T = 170$

Otsu's algorithm can be successfully applied only to document images with a histogram that has a bimodal distribution. In order to treat document images with a non-uniform background, a modification of Otsu's approach has been proposed by Cheriet et al. [16] that is based on a recursive application of the above procedure. At each recursion, it segments first the object with the lowest intensity from the given image. The recursive process continues until there is only one object (the darkest) left in the image. It is recommended that the recursion stops when

$$\eta \geq 0.95 \quad (4.8)$$

where  $\eta$  is the separability measure defined by Eq.(4.2). The efficiency of this methodology has been demonstrated for the binarization of real-life bank checks.

According to iterative thresholding methods (see survey paper [57]), the optimum threshold is chosen automatically as a result of an iterative process. Successive iterations provide increasingly cleaner extractions of the text region. At iteration  $n$ , a new threshold  $T_n$  is calculated using the average of the foreground and background class means  $m_f(T_n)$  and  $m_b(T_n)$ :

$$T_{n+1} = \frac{m_f(T_n) + m_b(T_n)}{2} \quad (4.9)$$

Iterations terminate when  $|T_n - T_{n+1}|$  becomes sufficiently small.

Several binarization approaches exploit the entropy of the distribution of the gray levels and use the maximization of the entropy of the thresholded image as indicative of maximum information transfer (see survey paper [57]). Knowing the a priori entropy  $H_T$  of the gray-level histogram, the optimal threshold is estimated by maximizing the upper bound of the a posteriori entropy. The a priori entropy  $H_T$  is

**Table 4.1** Overview of key document image binarization techniques

Reference	Category	Short description	Remarks
Otsu [54]	Global thresholding	The optimum threshold is calculated by separating the two classes of pixels (e.g., foreground and background) so that their between-class variance is maximized	Successful only to document images having bimodal distributed histogram
Cheriet et al. [16]	Global thresholding	A recursive application of [54] in order to treat document images with multiple background	Demonstrated for the binarization of bank checks
Solihin et al. [61]	Global thresholding	A two-stage thresholding approach requiring at a first stage each pixel to be assigned to one of three classes (foreground, background, fuzzy area)	Designed to binarize grayscale handwriting images
Niblack [50]	Local thresholding	A local binarization algorithm that calculates a pixelwise threshold by shifting a rectangular window across the image	Noise that is presented in the background may remain dominant in the final binary image
Sauvola and Pietikainen [56]	Local thresholding	A modification of [50] that adds a hypothesis on the gray values of text and background pixels	Noise is eliminated but text regions may be missed
Kamel and Zhao [32]	Local thresholding	The character stroke width and a global threshold are used in order to detect significant changes in gray-level values around the character body	It is very difficult to tune the parameters of the method
Yang and Yan [77]	Local thresholding	A modification of [32] in which character stroke width is automatically detected and the threshold is locally adapted	For poor quality grayscale document images without need of any manual parameter tuning
Gatos et al. [27]	Local thresholding	Adaptive method based on preprocessing, background surface calculation, suitable thresholding, and post-processing	For degraded documents with shadows, nonuniform illumination, low contrast, and noise
Kim et al. [35]	Hybrid thresholding	Global thresholding [54] on a difference image between the original and the water-filled image	Selection of two critical parameters on an experimental basis
Vonikakis et al. [72]	Hybrid thresholding	It combines the characteristics of the OFF center-surround cells of the Human Visual System with the classic Otsu binarization technique [54]	Experiments performed on a set of images with various computer-generated degradations
Tseng and Lee [68]	Hybrid thresholding	Based on analyzing the document layout and the intensity distribution. Pixels outside the detected blocks are binarized using Otsu technique [54]	Tested on a large set of images including newspapers, magazine articles, and business cards

(continued)

**Table 4.1** (continued)

Reference	Category	Short description	Remarks
Badekas and Papamarkos [6]	Combination of techniques	Combines the results of several global and local binarization algorithms using a Kohonen self-organizing map neural network	Tested on a variety of degraded document images
Gatos et al. [28]	Combination of techniques	Combines the binarization results of several global and adaptive methodologies using a majority voting strategy and incorporates edge map information	For historical and degraded document images
Su et al. [64]	Combination of techniques	Image pixels are labeled as foreground, background, or uncertain based on the results of several binarization techniques. Uncertain pixels are classified at a next step as foreground or background	It improves the performance of existing binarization methods, robust on degraded images
Chou et al. [18]	Using training samples	Divides the image into regions and decides how to binarize each region using a learning process and a support vector machine (SVM) classifier	Better visual quality and OCR performance than several global and local methods
Huang et al. [30]	Using training samples	Hidden Markov model (HMM)-based binarization using feature vectors calculated from several image directions	Better OCR performance than other conventional methods
Chen and Leedham [14]	Using training samples	Local feature vectors are used to analyze and find the best approach to threshold a local area after quad-tree decomposition	Trained using 300 historical images and tested on 300 degraded document images
Badekas et al. [7]	For color documents	A color reduction procedure is followed by a connected component analysis in order to classify components as text or non-text	Appropriate for noisy color or grayscale documents
Tsai and Lee [67]	For color documents	The color image is transformed into the luminance and saturation spaces. A decision tree is used to decide if luminance, saturation, or both features will be employed for the binarization process	Trained on 150 different color documents and tested on an additional 519 color documents
Wang et al. [74]	For color documents	Color text image binarization based on color clustering and binary texture feature analysis. Two kinds of texture features, run-length histogram and spatial-size distribution related, respectively, are extracted and explored	Experiments with more than 500 color text images collected from the Internet

calculated as follows:

$$H_T = - \sum_{i=0}^{255} p_i \ln p_i \quad (4.10)$$

where  $p_i$  the probability of occurrence of gray-level  $I$  (see Eq. (4.5)).

Two probability distributions are assumed, one for the object area and one for the background. The a priori entropies for the two classes of pixels are defined as

$$H_b = - \sum_{i=1}^t \frac{p_i}{P_t} \ln \left( \frac{p_i}{P_t} \right), \quad H_w = - \sum_{i=t+1}^{255} \frac{p_i}{1-P_t} \ln \left( \frac{p_i}{1-P_t} \right) \quad (4.11)$$

where  $P_t$  is the cumulative probability function and calculated as follows:

$$P_t = \sum_{i=0}^t p_i \quad (4.12)$$

When the sum of the two class entropies  $H_b$  and  $H_w$  reaches its maximum, the image is considered as optimally thresholded.

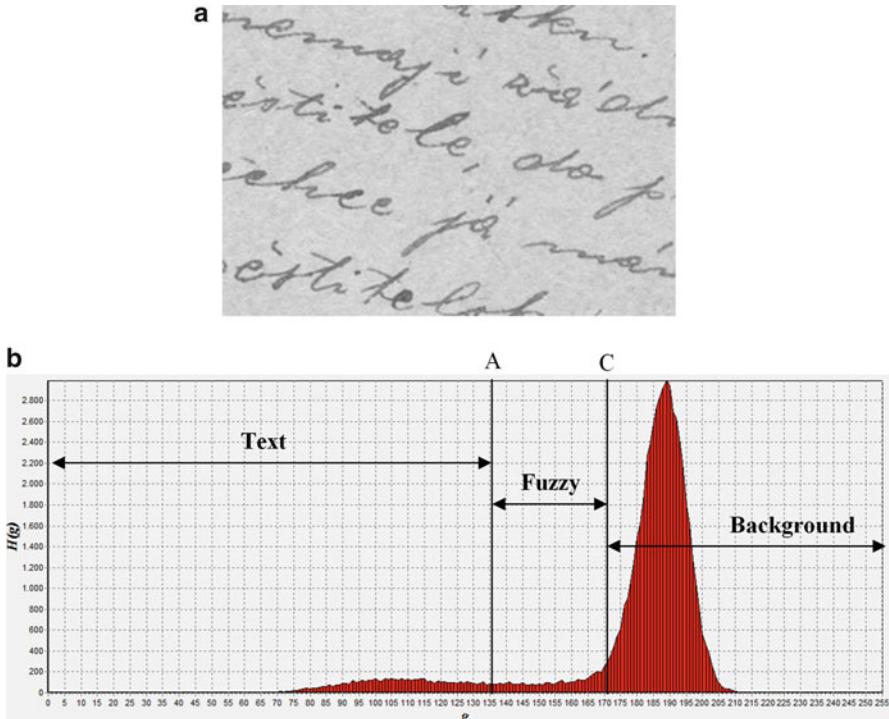
A class of global thresholding methods designed to binarize grayscale handwriting images has been proposed by Solihin et al. [61]. It is based on a two-stage thresholding approach requiring at a first stage each pixel of handwritten image to be assigned to one of three classes: foreground, background, and a fuzzy area between them where it is hard to determine whether a pixel belongs to the foreground or the background. Parameters  $A$  and  $C$  that are used to define these three classes (see Fig. 4.5) are calculated based on an integral ratio function. At a second stage, the optimum threshold  $T$  was empirically calculated based on the handwriting media used:

$$T = \begin{cases} C - \frac{1}{2(C-A)}, & \text{if the writing device is a felt tipped pen} \\ C - \frac{1}{10(C-A)}, & \text{if the writing device is a ballpoint pen} \\ C, & \text{if the writing device is a pencil} \\ C - \frac{1}{10(C-A)}, & \text{if the writing device is not specified} \end{cases} \quad (4.13)$$

## Local Thresholding Techniques

Niblack [50] introduced a local binarization algorithm that calculates a pixelwise threshold by shifting a rectangular window across the image. The threshold  $T$  for the center pixel of the window is computed using the mean  $m$  and the variance  $s$  of the gray values in the window:

$$T = m + ks \quad (4.14)$$



**Fig. 4.5** A grayscale histogram of a handwriting image: (a) original grayscale image and (b) the corresponding three class grayscale histogram  $H(g)$

where  $k$  is a constant set to  $-0.2$ . The value of  $k$  is used to determine how much of the total print object boundary is taken as a part of the given object. This method can distinguish the object from the background effectively in the areas close to the objects. The results are not very sensitive to the window size as long as the window covers at least one and two characters. However, noise that is present in the background remains dominant in the final binary image. Consequently, if the objects are sparse in an image, a lot of background noise will be left.

Sauvola and Pietikainen [56] propose a method that solves this problem by adding a hypothesis on the gray values of text and background pixels (text pixels have gray values near 0 and background pixels have gray values near 255), which results in the following formula for the threshold:

$$T = m + \left(1 - k \left(1 - \frac{s}{R}\right)\right) \quad (4.15)$$

where  $R$  is the dynamics of the standard deviation fixed to 128 and  $k$  takes on positive values (usually set to 0.5). Although this method gives better results for document images and noise is significantly eliminated, text regions may be also

missed. Example results of applying local thresholding methods [50, 56] as well as global thresholding method [54] are presented in Fig. 4.6.

Logical level thresholding technique uses not only the image gray-level values but also the stroke width  $SW$  of the characters to improve the binarization quality. According to the algorithm of Kamel and Zhao [32], computations are performed according to  $SW$  and a global threshold which are predetermined by the user. In particular, the gray level or the smoothed gray level of each processing point is compared with four local averages located in  $(2SW + 1) \times (2SW + 1)$  windows centered at two pairs of diametric points. This is expressed as follows:

$$b(x, y) = \begin{cases} 1, & V_{i=0}^3 [L(P_i) \wedge L(P'_i) \wedge L(P_{i+1}) \wedge L(P'_{i+1})] \text{ is true} \\ 0, & \text{otherwise} \end{cases} \quad (4.16)$$

where  $P'_i = P_{(i+4) \bmod 8}$ ,  $L(P) = \text{ave}(P) - g(x, y) > T$ , and

$$\text{ave}(P) = \frac{\sum_{-sw \leq i \leq sw} \sum_{-sw \leq j \leq sw} f(P_x - i, P_y - j)}{(2sw + 1)^2} \quad (4.17)$$

where  $P_x$  and  $P_y$  are the coordinates of the point  $P$  and  $g(x, y)$  is the gray level or its smoothed value. The original logical level technique of [32] is user-dependent since both the  $SW$  and the global threshold are predetermined by the user. Moreover, a global threshold is very difficult or even impossible to be tuned for document images with uneven illumination, big amount of noise, and other degradations. Yang and Yan [77] proposed the adaptive logical level technique (ALLT), in which the  $SW$  is automatically detected and the threshold is locally adapted. Concerning the  $SW$  detection in ALLT, the input image is divided in  $N \times N$  ( $N = 4, \dots, 8$ ) regions with the aim of finding some local areas with quasi-bimodal histogram. Specifically, the histogram analysis is performed within regions of the two diagonal directions if  $N$  is even and additionally in the vertical and horizontal directions if  $N$  is odd. The quasi-bimodal regions are used for the run-length histogram analysis and the  $SW$  is defined as the run-length with the highest frequency. The other improvement proposed in ALLT concerns the local adaptive threshold. For each processing point  $P$ , the minimum (min), maximum (max), and average (ave) gray value are calculated within a  $(2SW + 1) \times (2SW + 1)$  window centered at  $P$ . It is worth mentioning that if  $|\max - \text{ave}| = |\min - \text{ave}|$ , then the window expands to  $(2SW + 3) \times (2SW + 3)$  and calculations are performed one more time. The adaptive threshold  $T$  is produced as follows:

$$T = \begin{cases} a \left( \frac{2}{3} \min + \frac{1}{3} \text{ave} \right), & \text{if } |\max - \text{ave}| > |\min - \text{ave}| \\ a \left( \frac{2}{3} \min + \frac{2}{3} \text{ave} \right), & \text{if } |\max - \text{ave}| < |\min - \text{ave}| \\ a \text{ave}, & \text{if } |\max - \text{ave}| = |\min - \text{ave}| \end{cases} \quad (4.18)$$

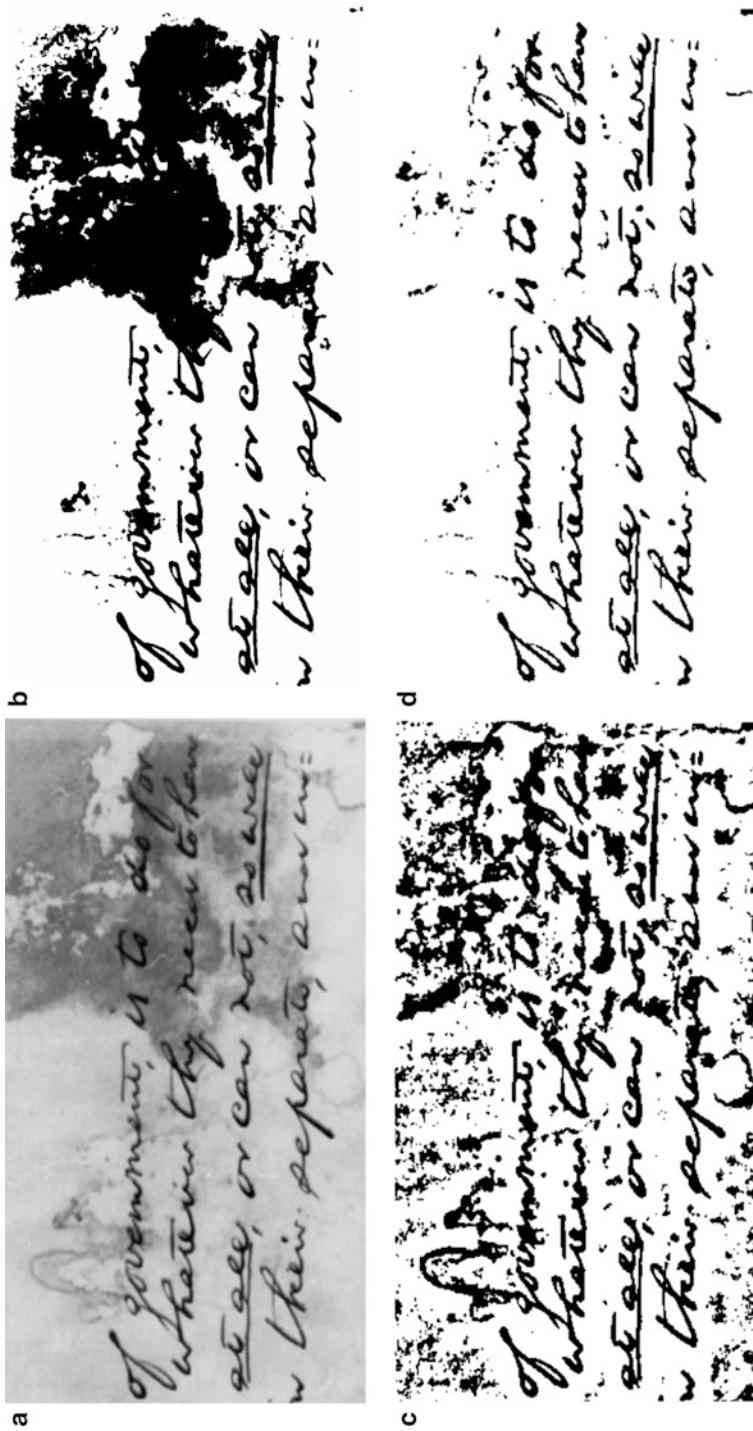
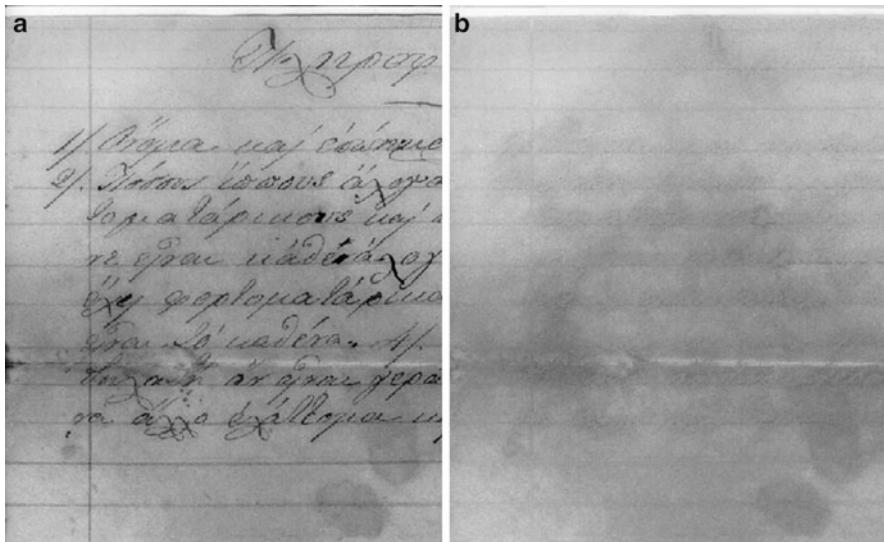


Fig. 4.6 Binarization results using global and local thresholding: (a) original grayscale image, (b) Otsu's method [54], (c) Niblack's method [50], and (d) Sauvola and Pietikainen method [56]



**Fig. 4.7** An example of the background surface calculated according to the local thresholding (Gatos et al. [27]): (a) original grayscale image and (b) the calculated background surface

where  $\alpha$  is a global predetermined parameter between 0.3 and 0.8, while  $2/3$  is recommended for most cases.

A local thresholding methodology has been proposed by Gatos et al. [27] for the binarization and enhancement of documents with degradations which occur due to shadows, nonuniform illumination, low contrast, large signal-dependent noise, smear, and strain. It follows several distinct steps: (a) a preprocessing procedure using a low-pass Wiener filter, (b) a rough estimation of foreground regions using Sauvola's approach [56] with a small  $k$  parameter (such as 0.1) in order to obtain a binary image that contains the foreground pixels plus some noise, (c) a background surface calculation by interpolating neighboring background intensities (see Fig. 4.7), (d) a thresholding by combining the calculated background surface with the original image using a threshold  $d$  that changes according to the grayscale value of the background surface  $B$  in order to preserve textual information even in very dark background areas, and, finally, (e) a post-processing step based on a consequent application of shrink and swell filtering in order to improve the quality of text regions and preserve stroke connectivity.

## Hybrid Thresholding Techniques

Hybrid thresholding techniques first enhance the difference between the background and foreground and then binarize the enhanced image by a simple global algorithm.

Kim et al. [35] propose a local adaptive thresholding method where an image is regarded as a 3-D terrain and its local property is characterized by a water flow model. The water flow model locally detects the valleys corresponding to regions that are lower than neighboring regions. The deep valleys are filled with dropped water whereas the smooth plain regions keep up dry. The final step in this method concerns the application of a global thresholding such as Otsu's method [54] on a difference image between the original terrain and the water-filled terrain. A shortcoming of this method is the selection of two critical parameters for the method, namely, the amount of rainfall and the parameter of mask size, which is done on an experimental basis.

Vonikakis et al. [72] proposed a document binarization algorithm that combines the characteristics of the OFF center-surround cells of the Human Visual System with the classic Otsu binarization technique [54]. Cells of two different scales are combined, increasing the efficiency of the algorithm and reducing the extracted noise in the final output. A new response function, which regulates the output of the cell according to the local contrast and the local lighting conditions, is also introduced. The Otsu technique [54] is used to binarize the outputs of the OFF center-surround cells. Quantitative experiments performed on a set of images with various computer-generated degradations, such as noise, shadow, and low contrast, demonstrate the superior performance of the proposed method against six other well-established binarization techniques.

Tseng and Lee [68] proposed a document image binarization approach based on analyzing the document layout and the intensity distribution. All blocks with individual background intensity in a document image were first extracted by applying a two-stage block extraction technique. Then, the intensity distribution of each extracted block was analyzed to determine the ranges of background intensity. Those pixels in a block were binarized according to whether their intensity values are within the union range of background intensity. Other pixels outside all extracted blocks were binarized using Otsu's global thresholding method [54]. The proposed technique was tested on a large set of images that included newspapers, magazine articles, and business cards.

## Combining Different Binarization Techniques

Recently, several binarization techniques combine the results of a set of binarization techniques in order to use the complementarity in success of each technique.

Badekas and Papamarkos [6] have proposed a document image binarization technique that takes advantage of the benefits of a set of selected global and local binarization algorithms by combining their results using a Kohonen self-organizing map neural network. Specifically, in the first stage the best parameter values for each independent binarization technique are estimated. In the second stage and in order to take advantage of the binarization information given by the independent techniques, the neural network is fed by the binarization results obtained by those techniques using their estimated best parameter values.

This procedure is adaptive because the estimation of the best parameter values depends on the content of images. The proposed technique is suitable to classify pixels that have high vagueness, that is, pixels which belong to edges and shadow areas and generally pixels that cannot be easily classified as foreground or background pixels. The proposed technique was tested on a variety of degraded document images. Motivated by the aforementioned technique, Gatos et al. [28] proposed a binarization methodology for historical and degraded document images that is based on (i) efficient preprocessing using the Wiener filter, (ii) combination of the binarization result of several global and adaptive state-of-the-art methodologies using a majority voting strategy, (iii) incorporation of the edge map in the grayscale image produced by the Canny edge detector, and (iv) application of efficient image post-processing based on mathematical morphology for the enhancement of the final result.

Su et al. [64] claim that by combining different binarization techniques, better performance can be achieved with careful analysis. Those pixels that are labeled the same by different methods are usually correctly classified, and those pixels which are classified as text by some methods and labeled as background by other methods have higher possibility to be misclassified than others. Based on such observation, we divide all the image pixels into three sets: foreground set, where those pixels are classified into foreground by all the examining binarization methods; background set, where those pixels are classified into background by all the examining binarization methods; and uncertain set, where the rest of pixels belong to, which is defined as follows:

$$P(x, y) = \begin{cases} \text{foreground}, & \sum_{i=1}^n B_i(x, y) = 0 \\ \text{background}, & \sum_{i=1}^n B_i(x, y) = n \\ \text{uncertain}, & \text{otherwise} \end{cases} \quad (4.19)$$

where  $P(x, y)$  denotes one image pixel, and  $B_i(x)$ , which is either 0 (foreground) or 1 (background), denotes the corresponding binarization result of pixels  $P(x, y)$  generated by the  $i$ th binarization methods. The pixels are then projected into a feature space. Those pixels in foreground and background sets can be viewed as correctly labeled samples and used to determine the label of those uncertain pixels. A classifier is then applied to iteratively classify those uncertain pixels into foreground and background.

## Using Training Samples

A training procedure for binarization is used by Chou et al. [18] for document images produced by cameras. This method divides an image into several regions and decides how to binarize each region. The decision rules are derived from a learning process that takes training images as input. Within each region  $r$ , one of

the following four operations is applied: set the whole of  $r$  to black, set the whole of  $r$  to white, use Otsu's method [54] to compute the threshold for  $r$ , or use the smallest Otsu threshold in the neighboring regions as the threshold for  $r$ . A learning process is used to establish the rules for deciding which of the above options should be adopted for each region. The rules are expressed as decision functions, which take a number of features extracted from  $r$  as input. For the learning process, a support vector machine (SVM) classifier is used having as input the following three features:  $T_{\text{Otsu}}(r) - T_{\min}(r)$ ,  $\mu(r)$ , and  $\sigma(r)$  where  $T_{\text{Otsu}}(r)$  is the Otsu threshold for  $r$ ,  $\mu(r)$  and  $\sigma(r)$  the mean and the standard deviation of the distribution of gray values in  $r$ , and  $T_{\min}(r)$  is defined as follows:

$$T_{\min}(r) = \min\{T_{\text{Otsu}}(r), \min_{s \in \Lambda(r)} T_{\text{Otsu}}(s)\} \quad (4.20)$$

where  $\Lambda(r)$  is the set of neighboring regions of  $r$ . Tests on images produced under normal and inadequate illumination conditions show that this method yields better visual quality and better OCR performance than three global binarization methods and four locally adaptive binarization methods.

In [30], a hidden Markov model (HMM)-based document image binarization algorithm is presented by Huang et al. In a first stage, a coarse global thresholding method is used to discriminate the bright part of the whole image from the foreground pixels which have lower values. The mean gray-level value is regarded as the preliminary threshold value of this process. In a second stage, the remaining pixels which are supposed to be a mixture of foreground and background are applied to the HMM pixel classifier to obtain the attribute of each pixel. A novel feature extraction method suitable for HMM-based image binarization is used. Inside a window of  $N$  by  $N$ , four feature vectors are calculated from the four following directions, horizontal, left up to right down, vertical, and right up to left down, around each of which contains  $N$  elements.

In [14], Chen and Leedham propose a machine learning framework for thresholding degraded historical document images. A new thresholding structure called the decompose algorithm is proposed and compared against some existing single-stage algorithms. The decompose algorithm uses local feature vectors to analyze and find the best approach to threshold a local area. Instead of employing a single thresholding algorithm, automatic selection of an appropriate algorithm for specific types of subregions of the document is performed. The original image is recursively broken down into subregions using quad-tree decomposition until a suitable thresholding method can be applied to each subregion. The algorithm has been trained using 300 historical images and evaluated on 300 “difficult” document images, in which considerable background noise or variation in contrast and illumination exists. Quantitative analysis of the results by measuring text recall and qualitative assessment of processed document image quality are reported. The decompose algorithm is demonstrated to be effective at resolving the problem in varying quality historical images.

## Binarization of Color Documents

A method for the binarization of color document images is presented in [7] by Badekas et al. Initially, the colors of the document image are reduced to a small number using a new color reduction technique. Specifically, this technique estimates the dominant colors and then assigns the original image colors to them in order that the background and text components become uniform. Each dominant color defines a color plane in which the connected components (CCs) are extracted. Next, in each color plane, a CC filtering procedure is applied which is followed by a grouping procedure. At the end of this stage, blocks of CCs are constructed which are next redefined by obtaining the direction of connection (DOC) property for each CC. Using the DOC property, the blocks of CCs are classified as text or non-text. The identified text blocks are binarized properly using suitable binarization techniques, considering the rest of the pixels as background. The final result is a binary image which contains always black characters in white background independently of the original colors of each text block. The proposed document binarization approach can also be used for binarization of noisy color (or grayscale) document images.

According to the methodology proposed in [67] by Tsai and Lee, the color document image is initially transformed into the luminance and saturation spaces. Then, the luminance histogram is computed and a Gaussian smoothing filter is employed to remove unreliable peaks and valleys upon the histogram. The threshold candidates are then selected from the histogram. Next, the entire image distribution is analyzed to extract statistical features to be applied in a decision tree. The tree decides if luminance, saturation, or both features will be employed to binarize color document images. First, if the document image colors are concentrated within a limited range, saturation is employed. Second, if the image foreground colors are significant, luminance is adopted. Third, if the image background colors are concentrated within a limited range, luminance is also applied. Fourth, if the total number of pixels with low luminance (less than 60) is limited, saturation is applied; else, both luminance and saturation are employed. Finally, the color features are used for image binarization. The method was trained on 150 different color documents and tested on an additional 519 color documents. The performance analysis indicated that the method is efficient and effective in color documents with foreground and background colors either close or mixed.

A color text image binarization based on binary texture analysis is proposed in [74] by Wang et al. It is designed to overcome the limitations of existing techniques for color text images. This algorithm efficiently integrates color clustering and binary texture feature analysis. Two kinds of features capable of effectively characterizing text-like binary textures, including run-length histogram and spatial-size distribution-related features, are extracted and explored. In addition, in order to handle varying colors of the text in the image, a combination strategy is implemented among the binary images obtained by color clustering. The effective cooperation of these techniques enables the algorithm to survive those complex background conditions, as existing techniques tend to fail.

## Document Image Enhancement

Document images usually suffer from several defects mainly due to natural ageing, environmental conditions, usage, poor storage conditions of the original, as well as human manipulations during the scanning process. Since document image analysis and recognition methods usually assume a smooth background and a good quality of printing or writing, it is imperative to have an efficient image enhancement process in order to restore the good quality of document images. Moreover, document image enhancement process enhances the readability of text areas and permits less image storage space. The main artifacts encountered in document images can be categorized into the following four broad categories:

**Low contrast and uneven background illumination.** At low contrast documents, text regions are not clear due to poor contrast between the foreground text and the background. Documents with uneven background illumination have a variable background intensity (see Fig. 4.8).

**Bleed-through, shining, or shadow-through effects.** Bleed-through is the result of the diffusion of ink from one side of a page through the other side. The letters from one page also appear on the other page when the printing ink was not dry or due to the long time of pressing two leaves together (see Fig. 4.9). A similar degradation phenomenon is the shining or shadow-through effect. In this case, if a paper is relatively thin, then because of the transparency of the paper the information on one side of document interferes with the information contained on the other side.

**Damaged characters or noisy background.** The characters may be of poor quality, fainted, and merged, with holes or noisy edges. Spots, smears, shadows, or noise may appear at the background (see Fig. 4.10).

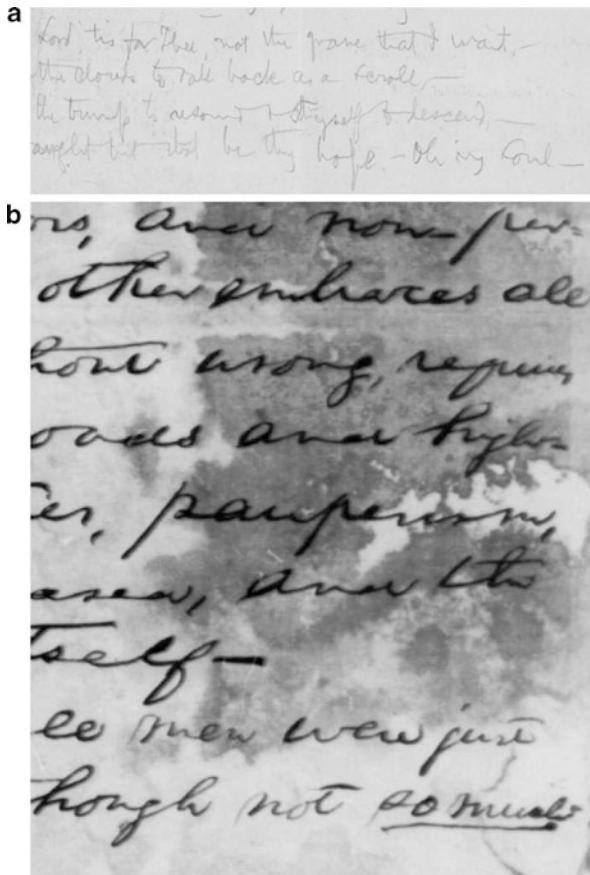
**Borders or parts of neighboring page.** When capturing a document, the resulting document image is often framed by a noisy black border or includes noisy text regions from neighboring pages (see Fig. 4.11).

In this section, the main document image enhancement approaches that have been proposed in the literature to face all the above artifacts will be presented. An overview of the key document image enhancement techniques is given in Table 4.2.

### Low Contrast and Uneven Background Illumination

A preprocessing stage of the document image is essential for contrast enhancement between background and text areas, smoothing of background texture, as well as the elimination of noisy areas. The use of a low-pass Wiener filter has been proved efficient for the above goals and has been used as preprocessing stage at a binarization process [27]. Contrast enhancement is a major issue in image processing, while histogram equalization is the most common, simple, and effective contrast

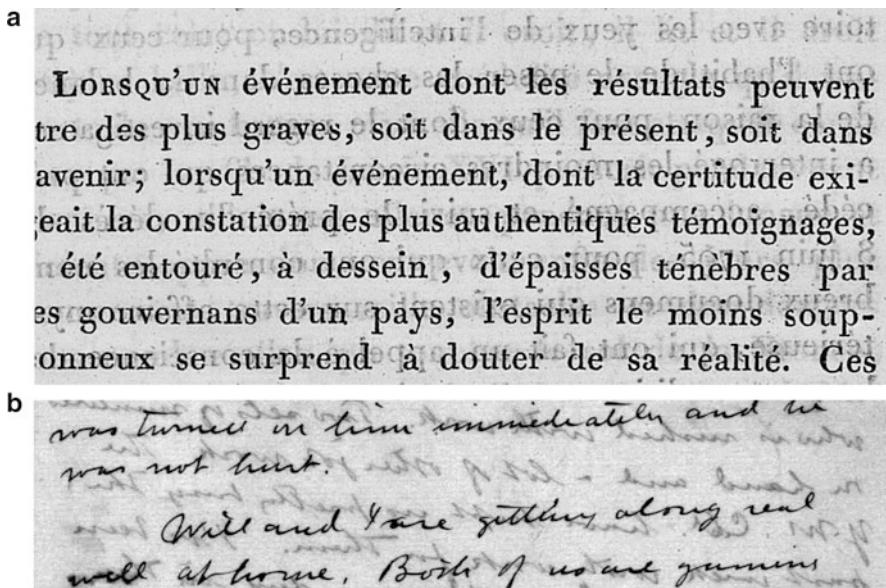
**Fig. 4.8** Document image examples having (a) low contrast and (b) uneven background illumination



enhancement technique. A generalized fuzzy operator preprocessed with histogram equalization and partially overlapped sub-block histogram equalization is used from Leung et al. [40] for reducing background noise and increasing the readability of text by contrast enhancement. This method achieves good performance in contrast enhancement of extremely low contrast and low illuminated document images.

Total variation regularization flattens background gray levels and produces an image where background noise is considerably reduced. Nonlocal means filtering can smooth character parts and improve character quality based on neighboring data information. The above two techniques are combined by Likforman et al. [43] in order to enhance the quality of historical printed document images.

Morphological operations can be also used in order to enhance the background of document images. In Nomura et al. [51], morphological operations are used to remove undesirable shapes called critical shadows on the background of document images before proceeding to binarization. These critical shadows may appear due to changes in color or size, very low contrast, low quality of focalization, and poor,



**Fig. 4.9** Parts of machine-printed and handwritten document images suffering (a) from the bleed-through and (b) from the shining-through effect

**Fig. 4.10** An example of a document image with damaged characters and noisy background

An ACCOUNT of their ORIGIN, LANGUAGE, CIVIL CUSTOMS, LAWS, FORM OF GOVERNMENT, WAR and DOMESTIC LIFE, their HABITS, DISEASES, FACTURES, and METHOD of CURE, sufficient to render it

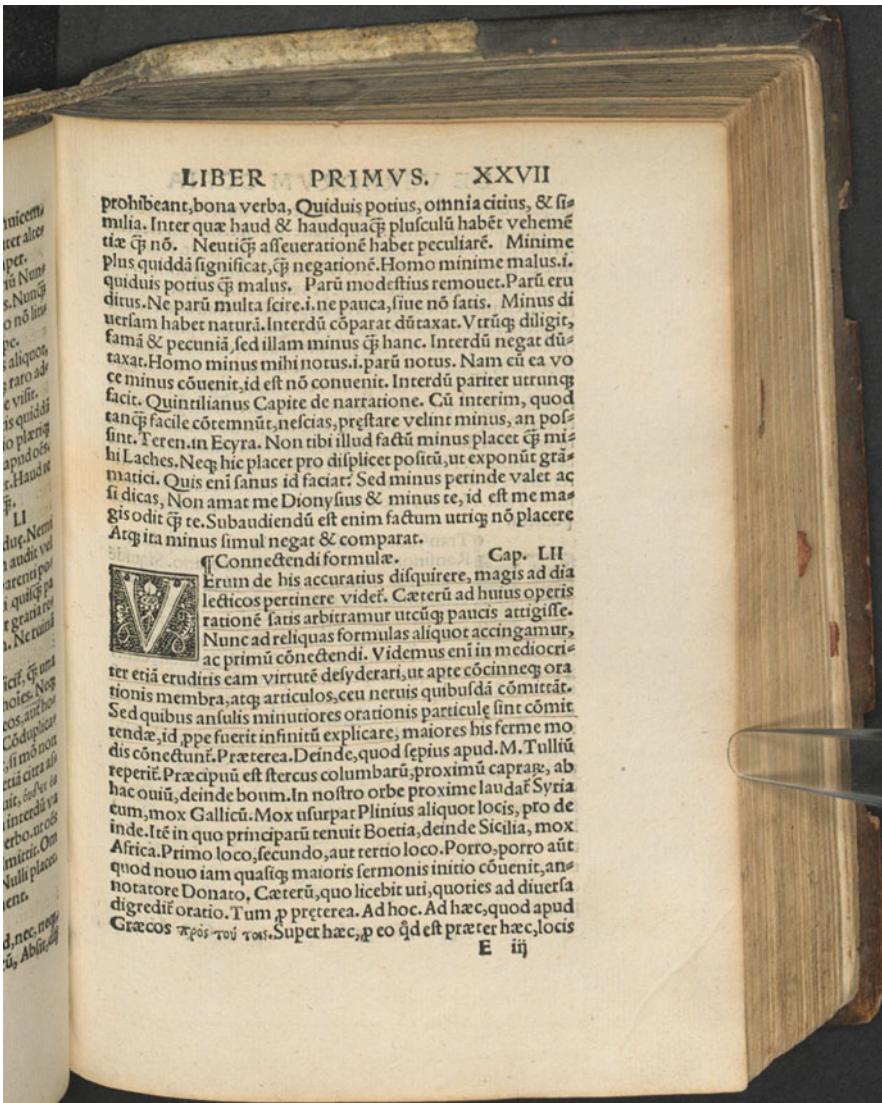
and nonuniform illumination. Method [51] adaptively and without manual fine-tuning of parameters locates these critical shadows on grayscale degraded document images using morphological operations and lightens them before applying eventual thresholding process.

## Bleed-Through, Shining, or Shadow-Through Effects

The methods proposed in the literature dealing with bleed-through, shining, or shadow-through effects can be categorized into either non-registration (or blind) or registration (or non-blind) methods, depending on whether the verso image of a page is available and precisely registered to the recto image.

### Non-registration (Blind)-Based Methods

In non-registration-based methods, the two sides of a page are treated independently and processed separately. Those methods attempt to clean the front side of a



**Fig. 4.11** An example of a document image with noisy black border as well as with text part from the neighboring page

document without referring to the reverse side. Most of these methods treat bleed-through interference as a kind of background artifacts or noise and remove it using threshold-like techniques [27, 35, 68].

A recursive unsupervised segmentation approach applied on the decorrelated data space by the principal component analysis has been proposed by

**Table 4.2** Overview of key document image enhancement techniques

Reference	Category	Short description	Remarks
Leung et al. [40]	Low contrast and uneven background illumination	Uses a generalized fuzzy operator preprocessed with histogram equalization and partially overlapped sub-block histogram equalization	Good contrast enhancement of extremely low contrast and low illuminated document images
Likforman et al. [43]	Low contrast and uneven background illumination	Combination of total variation regularization and nonlocal means filtering in order to enhance the quality of historical printed document images	It includes only one main parameter. Extensive tests based on OCR results
Nomura et al. [51]	Low contrast and uneven background illumination	Morphological operations are used to remove undesirable shapes called critical shadows on the background of document images	Does not need manual fine-tuning of parameters. Tested on degraded word images
Fadura et al. [23]	Bleed-through, shining, or shadow-through effects – non-registration	A recursive unsupervised segmentation approach applied on the decorrelated data space by the principal component analysis to remove bleed-through	Does not require any specific learning process or any input parameters
A. Tonazzini [65]	Bleed-through, shining, or shadow-through effects – non-registration	It is based on the projection of the RGB image into alternative color spaces that allow the enhancement of the main text areas	Does not require any intervention from the user side and no parameters need to be set
Dubois and Pathak [22]	Bleed-through, shining, or shadow-through effects – registration	An affine transformation is used to register the two sides; bleed-through is identified and replaced by the background color or intensity	Tested on documents generated under controlled conditions as well as on original manuscripts
Moghaddam and Cheriet [49]	Bleed-through, shining, or shadow-through effects – registration	It uses a new variational model with an extra term for reverse diffusion between the two sides of the document transferred to the wavelet domain	A blind (non-registration) version is also provided
Ajayi et al. [52]	Damaged characters or noisy background	An automated adaptive system to correct text degradations in typewritten documents based on lookup table classification algorithms	It is dependent on generating good quality ground truth images used for training
Drira et al. [21]	Damaged characters or noisy background	It suggests the application of the Partial Diffusion Equation (PDE) for enhancing text in degraded document images	This approach requires neither training nor segmentation steps
Shafait and Breuel [58]	Borders or parts of adjacent page – connected component analysis	It combines projection profile analysis with connected component removal to identify borders of noise regions	An efficient method simple to understand and implement

(continued)

**Table 4.2** (continued)

Reference	Category	Short description	Remarks
Le et al. [38]	Borders or parts of adjacent page – projection profiles	It is based on classification of blank, textual, and non-textual rows and columns, location of border objects, and an analysis of projection profiles and crossing counts of textual squares	It uses several heuristics and it is based on several assumptions
Stamatopoulos et al. [62]	Borders or parts of adjacent page – projection profiles	It is based on projection profiles combined with a connected component labeling process. Signal cross-correlation is also used for verification	Experimental results on several camera document images, mainly historical
Haji et al. [29]	Borders or parts of adjacent page – projection profiles	A simultaneous document margin removal and skew correction based on corner detection in projection profiles	Applied to a collection of document images with different types of margin noise, layouts, and intensity levels
Avila and Lins [4]	Borders or parts of adjacent page – flood-fill	Invading and non-invading border detection algorithms. The non-invading border algorithm takes into account two parameters related to the nature of the documents in order to restrain flooding in the whole connected area	Tested on over 20,000 images and compared with several commercial tools (Scanfix, Leadtools, BlackIce, and Skyline Tools)
Avila and Lins [5]	Borders or parts of adjacent page – flood-fill	It is based on “flood-fill” component labeling and region adjacency graphs for removing noisy black borders in monochromatic images	Works even on images of torn-off documents or in the presence of irregular white stripes on the noise border frame
Fan et al. [24]	Borders or parts of adjacent page	It removes the black borders of scanned documents by reducing the resolution of the document image	It does not consider noisy text regions
Shafait et al. [59]	Borders or parts of adjacent page	It performs document image cleanup by detecting the page frame of the document	It can handle documents with a very large amount of noise with reasonable accuracy

Fadura et al. [23] for the enhancement of historical document images suffering from the bleed-through effect. This technique does not require any specific learning process or any input parameters. The stopping criterion for the proposed recursive approach has been determined empirically and set to a fixed number of iterations.

Recently, A. Tonazzini [65] has proposed some simple and fast procedures, based on the projection of the RGB image of a document into alternative color

spaces. These color spaces, both fixed and self-adaptive, in many cases allow for the enhancement of the main text and the extraction of various features of the document. It is shown that even simpler arithmetic operations among the color channels can be effective for removing bleed-through, refocusing and improving the contrast of the foreground text.

### **Registration (Non-blind)-Based Methods**

Registration-based methods require that the verso image of a page is available and precisely registered to the recto image. In order to register the verso with the recto image, several techniques have been proposed based on (i) intensity characteristics, intensity patterns are compared using correlation metrics; (ii) feature characteristics, correspondence between features such as points, lines, and contours is searched; (iii) transformation models; and (iv) spatial and frequency domain. In Dubois and Pathak's approach [22], an affine transformation is used to register the two sides, determining the parameters using an optimization method. Once the two sides have been registered, areas consisting primarily of bleed-through are identified and replaced by the background color or intensity. A new variational model has been introduced by Moghaddam and Cheriet [49] for the enhancement of degraded double-sided document images. This model has an extra term for reverse diffusion between the two sides of a document. By transferring the model to the wavelet domain and using the hard wavelet shrinkage, the solution of the model is obtained in a single step. The transformation to the wavelet domain is performed using the dual-tree complex wavelet transform which is a shift-invariant discrete wavelet transform.

### **Damaged Characters or Noisy Background**

The enhancement of character areas and the elimination of noise in the background are usually a post-processing step of the binarization algorithms [27, 77]. Common operations that are applied in order to enhance the quality of a binary document image include masks, connected component analysis, morphological operations, as well as of shrink and swell operations.

Concerning the enhancement of grayscale or color images, Drira et al. [21] presented a study for restoring degraded text characters which consists of repairing the shapes of the features as well as extrapolating lost information. This study suggests the application of the Partial Diffusion Equation (PDE) for enhancing text in degraded document images. Ajayi et al. [52] introduced an automated adaptive system to correct text degradations in typewritten documents. It is based on lookup table classification algorithms and learns the corrections of patterns of text degradation in document images. Actual degraded historical documents are used for training a degradation model. The main limitation of this approach is that it is dependent on generating good quality ground truth images for the document collection to which it will be applied.

## Borders or Parts of Adjacent Page

Border removal methods fall into the following categories:

### Connected Component Analysis-Based Methods

The most common and easy to implement approach to eliminate marginal noise is to perform document cleaning by filtering out connected components based on their size and aspect ratio [53, 58]. However, when characters from the adjacent page are also present, they usually cannot be filtered out using only these features.

### Projection Profile-Based Methods

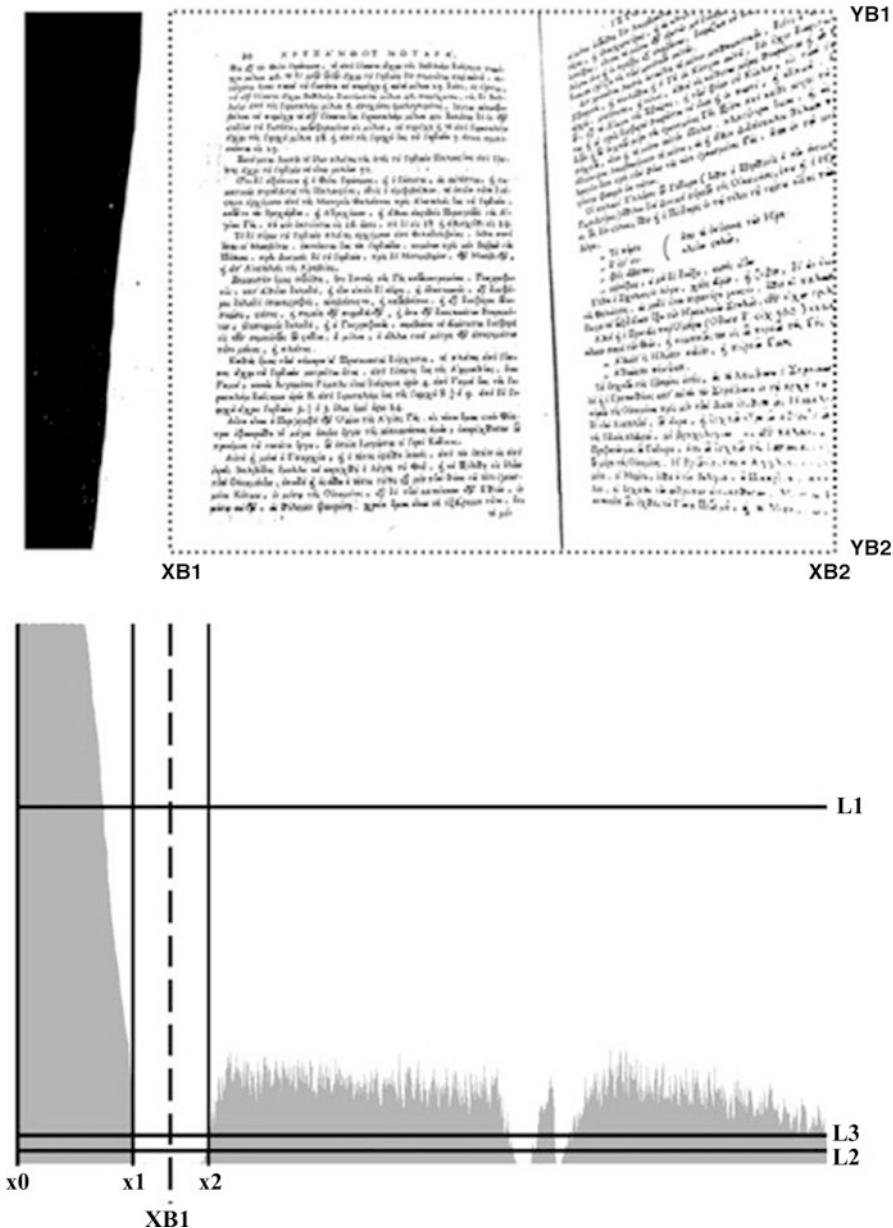
Le et al. [38] proposed a method for border removal which is based on classification of blank, textual, and non-textual rows and columns, location of border objects, and an analysis of projection profiles and crossing counts of textual squares. Their approach uses several heuristics, and also it is based on the assumption that the page borders are very close to edges of images and borders are separated from image contents by a white space. However, this assumption is often violated.

Stamatopoulos et al. [62] have presented a technique for enhancing the document images captured by a digital camera by automatically detecting the document borders and cutting out noisy black borders as well as noisy text regions appearing from neighboring pages. This methodology is based on projection profiles combined with a connected component labeling process (see Fig. 4.12). Signal cross-correlation is also used in order to verify the detected noisy text areas.

A simultaneous document margin removal and skew correction based on corner detection in projection profiles has been proposed by Haji et al. [29]. The basic function of the algorithm is to find the corners which correspond to the page margins from the projection profiles of the input image. For a straight page, the leftmost and rightmost sharp corners in the horizontal profile of the image correspond to the left and right margins, and the leftmost and rightmost sharp corners in the vertical profile of the image correspond to the upper and lower margins. For skewed pages, we observe that horizontal and vertical projection profiles have an isosceles trapezoidal shape.

### “Flood-Fill” Algorithms

Avila and Lins [4] proposed the invading and non-invading border detection algorithms which work as “flood-fill” algorithms. The invading algorithm assumes that the noisy black border does not invade the black areas of the document. It moves outside from the noisy surrounding borders towards the document. In the case that the document text region is merged to noisy black borders, the whole area, including the part of text region, is flooded and removed. Contrarily, the non-invading border algorithm assumes that noisy black border merges with document information. In order to restrain flooding in the whole connected area, it takes into account two parameters related to the nature of the documents. These are the maximum size of a segment belonging to a document and the maximum distance between lines. Also, Avila and Lins [5] proposed an algorithm based on “flood-fill” component labeling



**Fig. 4.12** Example of vertical projection profile used for border removal

and region adjacency graphs for removing noisy black borders in monochromatic images. The proposed algorithm encompassed five steps: flooding, segmentation, component labeling, region adjacency graph generation, and noise black border removal.

## Other Methods

Fan et al. [24] proposed a method for removing noisy black regions overlapping the text region, but do not consider noisy text regions. They proposed a scheme to remove the black borders of scanned documents by reducing the resolution of the document image. This approach consists of two steps, marginal noise detection and marginal noise deletion. Marginal noise detection consists of three steps: (1) resolution reduction, (2) block splitting, and (3) block identification. Marginal noise detection makes the textual part of the document disappear leaving only blocks to be classified either as images or borders by a threshold filter. Marginal noise has to be deleted after it has been detected. The deletion process should be performed on the original image instead of the reduced image. The block classification is used to segment the original image, removing the noisy black borders.

Shafait et al. [59] proposed a new perspective for document image cleanup by detecting the page frame of the document. The goal of page frame detection is to find the actual page contents area, ignoring marginal noise along the page border (see Fig. 4.13). First, a geometric model is built for the page frame of a scanned structured document (journal article, book, magazine). Then, a geometric matching method is used to find the globally optimal page frame with respect to a defined quality function.

---

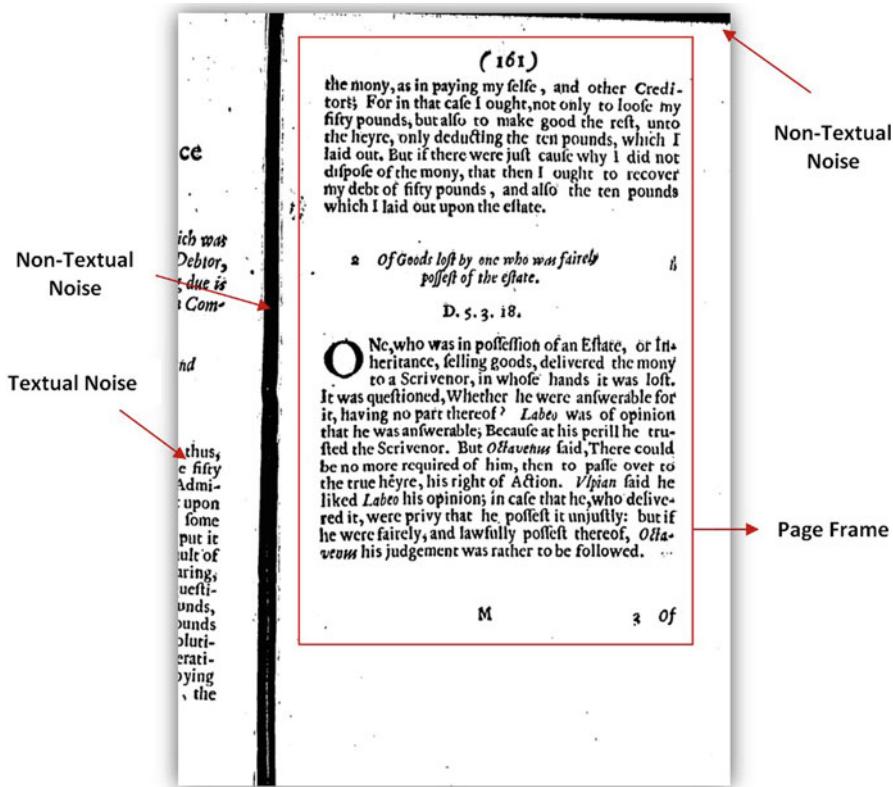
## Document Image Normalization

Document digitization with either flatbed scanners or camera-based systems results in document images which often suffer from skew, warping, and perspective distortions that deteriorate the performance of current OCR approaches. On the other hand, document orientation is not known *a priori*, and as a result, the text may be rotated by  $90^\circ$  or even be upside down. To this end, a document image normalization step is imperative in order to restore text areas horizontally aligned without any distortions as well as in  $0^\circ$  angle. This step also includes slant correction for the case of handwritten documents where there is often a deviation of the near-vertical strokes from the vertical direction.

## Page Orientation

The first step of document image normalization concerns the detection of text orientation. Usually, page orientation techniques involve the detection of portrait or landscape orientation, where the page has a skew of around  $0^\circ$  or  $90^\circ$ , correspondingly, as well as the detection of right side up or upside down orientation where the page has a skew of around  $0^\circ$  or  $180^\circ$ , correspondingly. An overview of the key page orientation estimation techniques is given in Table 4.3.

Portrait/landscape orientation detection is mainly accomplished by using projection histograms as well as by counting black-to-white transitions.



**Fig. 4.13** An example of a page frame along with textual and non-textual noise

In [39], Le et al. use local analysis and projection histograms. The proposed methodology consists of three steps: textual and non-textual squares classification, textual squares page orientation estimation, and squares grouping. At the textual and non-textual squares classification step, the entire binary image is divided into squares based on a pyramidal image date structure, and each square is then classified as textual or non-textual based on several empirical conditions. At the textual squares page orientation step, the page orientation of each textual square is estimated by using a projection profiles or a square difference method. The projection profile method is based on an analysis of shapes of horizontal and vertical projection histograms, while the square difference method is based on the comparison between squared sums of horizontal and vertical projection histograms. According to the square difference method, text orientation is defined as follows:

$$\begin{aligned}
 S_H \geq aS_v &\Rightarrow \text{text orientation} = \text{portrait} \\
 S_H < bS_v &\Rightarrow \text{text orientation} = \text{landscape} \\
 bS_v \geq S_H > aS_v &\Rightarrow \text{text orientation} = \text{uncertain}
 \end{aligned} \tag{4.21}$$

**Table 4.3** Overview of key page orientation estimation techniques

Reference	Category	Short description	Remarks
Le et al. [39]	Portrait/ landscape orientation detection	Consists of three steps: textual and non-textual squares classification, textual squares page orientation estimation, and squares grouping. Orientation is estimated using a projection profile as well as a square difference method	Performs an accuracy rate of 99.9 % on a variety of more than 10,000 images
Yin [78]	Portrait/ landscape orientation detection	Counts the black-to-white transitions vertically and horizontally in the document image after applying a horizontal and vertical smoothing	Part of a skew detection and block classification process
Caprari [13]	Text page up/down orientation determination	Exploits an up/down asymmetry of passages of text composed of roman letters and Arabic numerals. Calculates the statistical excess of ascending over descending characters in the text areas	Tested on 226 pages (some of them containing noise) with 100 % success
Aradhye [3]	Text page up/down orientation determination	It analyzes the “open” portions of text blobs to determine the direction in which the open portions face	Adapted for roman and non-roman scripts, automated mail processing, checks, envelops, etc.
van Beusekom et al. [70]	One-step skew and orientation detection	A one-step skew and orientation detection method using a well-established geometric text line model	The effectiveness of the method is demonstrated using a publicly available dataset

where  $S_H$  and  $S_V$  the squared sums of horizontal and vertical projection histograms and  $a, b$  the upper and lower bound user-defined parameters. Each textual square is then assigned its mode weight as the total black pixels of a portrait or landscape mode textual square. At the squares grouping step, the page orientation of a binary image is determined using the notion of a pyramidal image data structure.

In method [78] proposed by Yin, the determination of text orientation is accomplished by counting the black-to-white transitions vertically and horizontally in the document image after applying a horizontal and vertical smoothing based on the run-length smoothing (RLSA) procedure [73]. As it is demonstrated in Fig. 4.14, we first apply a horizontal smoothing and calculate the black-to-white transitions in vertical direction ( $TPix1$ ). Then, we first apply a vertical smoothing and calculate the black-to-white transitions in horizontal direction ( $TPix2$ ). Text orientation is defined as follows:

$$\begin{aligned} TPix1 < TPix2 &\Rightarrow \text{text orientation} = \text{portrait} \\ TPix1 \geq TPix2 &\Rightarrow \text{text orientation} = \text{landscape} \end{aligned} \quad (4.22)$$

The orientation of the document image of Fig. 4.14 is detected as portrait since  $TPix1 = 83,044 < TPix2 = 141,782$ .

An algorithm for text page up/down orientation determination is presented in [13] by Caprari. The algorithm exploits an up/down asymmetry of passages of text

**a**

- 34 -

L'ordre de lancement et de réalisation des applications fait l'objet de décisions au plus haut niveau de la Direction Générale des Télécommunications. Il n'est certes pas question de construire ce système intégré "en bloc" mais bien au contraire de procéder par étapes, par paliers successifs. Certaines applications, dont la rentabilité ne pourra être assurée, ne seront pas entreprises. Actuellement, sur trente applications qui ont pu être globalement définies, six en sont au stade de l'exploitation, six autres se sont vu donner la priorité pour leur réalisation.

Chaque application est confiée à un "chef de projet", responsable successivement de sa conception, de son analyse-programmation et de sa mise en oeuvre dans une région-pilote. La généralisation ultérieure de l'application réalisée dans cette région-pilote dépend des résultats obtenus et fait l'objet d'une décision de la Direction Générale. Néanmoins, le chef de projet doit dès le départ considérer que son activité a une vocation nationale donc refuser tout particularisme régional. Il est aidé d'une équipe d'analystes-programmeurs et entouré d'un "groupe de conception" chargé de rédiger le document de "définition des objectifs globaux" puis le "cahier des charges" de l'application, qui sont adressés pour avis à tous les services utilisateurs potentiels et aux chefs de projet des autres applications. Le groupe de conception comprend 6 à 10 personnes représentant les services les plus divers concernés par le projet, et comporte obligatoirement un bon analyste attaché à l'application.

## II - L'IMPLANTATION GEOGRAPHIQUE D'UN RESEAU INFORMATIQUE PERFORMANT

L'organisation de l'entreprise française des télécommunications repose sur l'existence de 20 régions. Des calculateurs ont été implantés dans le passé au moins dans toutes les plus importantes. On trouve ainsi des machines Bull Gamma 30 à Lyon et Marseille, des GE 425 à Lille, Bordeaux, Toulouse et Montpellier, un GE 437 à Massy, enfin quelques machines Bull 300 TI à programmes câblés étaient récemment ou sont encore en service dans les régions de Nancy, Nantes, Limoges, Poitiers et Rouen ; ce parc est essentiellement utilisé pour la comptabilité téléphonique.

A l'avenir, si la plupart des fichiers nécessaires aux applications décrites plus haut peuvent être gérés en temps différé, un certain nombre d'entre eux devront nécessairement être accessibles, voire mis à jour en temps réel : parmi ces derniers le fichier commercial des abonnés, le fichier des renseignements, le fichier des circuits, le fichier technique des abonnés contiendront des quantités considérables d'informations.

Le volume total de caractères à gérer en phase finale sur un ordinateur ayant en charge quelques 500 000 abonnés a été estimé à un milliard de caractères au moins. Au moins le tiers des données seront concernés par des traitements en temps réel.

Aucun des calculateurs énumérés plus haut ne permettait d'envisager de tels traitements. L'intégration progressive de toutes les applications suppose la création d'un support commun pour toutes les informations, une véritable "Banque de données", répartie sur des moyens de traitement nationaux et régionaux, et qui devra rester alimentée, mise à jour en permanence, à partir de la base de l'entreprise, c'est-à-dire les chantiers, les magasins, les guichets des services d'abonnement, les services de personnel etc.

L'étude des différents fichiers à constituer a donc permis de définir les principales caractéristiques du réseau d'ordinateurs nouveaux à mettre en place pour aborder la réalisation du système informatif. L'obligation de faire appel à des ordinateurs de troisième génération, très puissants et dotés de volumineuses mémoires de masse, a conduit à en réduire substantiellement le nombre.

L'implantation de sept centres de calcul interrégionaux constituera un compromis entre : d'une part le désir de réduire le coût économique de l'ensemble, de faciliter la coordination des équipes d'informatiens; et d'autre part le refus de créer des centres trop importants difficiles à gérer et à diriger, et posant des problèmes délicats de sécurité. Le regroupement des traitements relatifs à plusieurs régions sur chacun de ces sept centres permettra de leur donner une taille relativement homogène. Chaque centre "gerera" environ un million d'abonnés à la fin du VIème Plan.

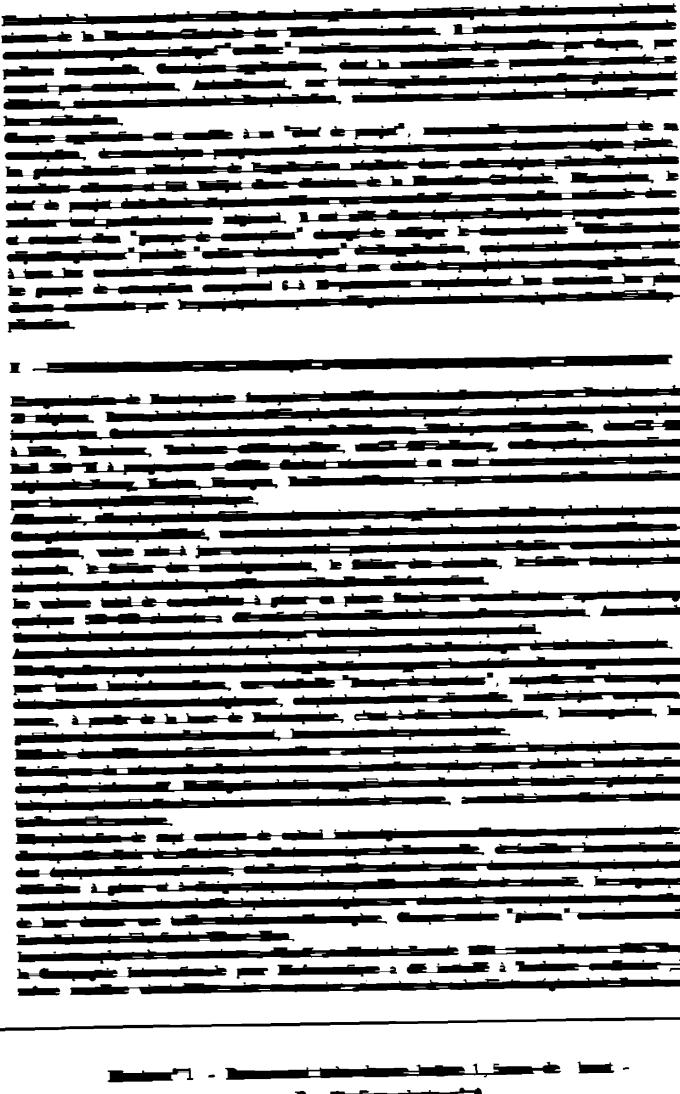
La mise en place de ces centres a débuté au début de l'année 1971 : un ordinateur IRIS 50 de la Compagnie Internationale pour l'Informatique a été installé à Toulouse en février ; la même machine vient d'être mise en service au centre de calcul interrégional de Bordeaux,

Photo n° 1 - Document très dense lettre 1,5mm de haut -  
Restitution photo n° 9

**Fig. 4.14** (continued)

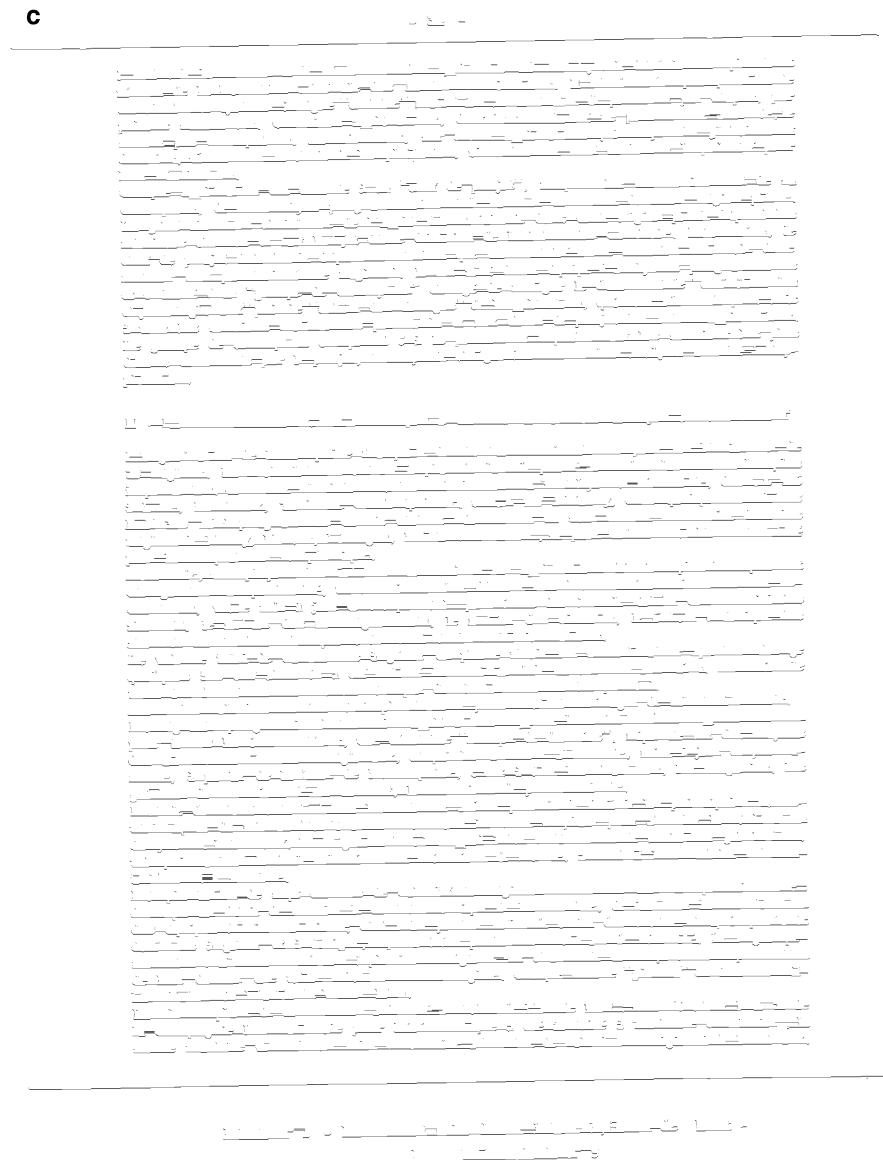
composed of roman letters and Arabic numerals. Up/down text asymmetry is a result of the statistical excess of ascending characters over descending characters in the text areas. Specifically, for lower-case letters in common written English, the frequencies of occurrence of letters in the top, middle, and bottom rows are 26.5, 67.25, and 6.25 %, respectively. In order to calculate the document up/down

b



**Fig. 4.14** (continued)

asymmetry, horizontal projections are used. A method for determining up/down orientation of text in roman and non-roman scripts is presented in [3] by Aradhye. The method analyzes the “open” portions of text blobs to determine the direction in which the open portions face. By determining the respective densities of blobs

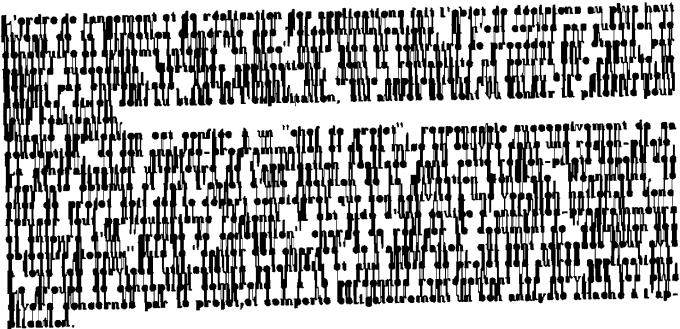


**Fig. 4.14** (continued)

opening in a pair of opposite directions (e.g., right or left), the method can establish the direction in which the text as a whole is oriented. First, a method for determining the up/down orientation of roman text based on the asymmetry in the openness of most roman letters in the horizontal direction is presented. For non-roman text,

d

- 84 -



■ - L'IMPLANTATION GEOGRAPHIQUE D'UN RESEAU INFORMATIQUE PERFORMANT

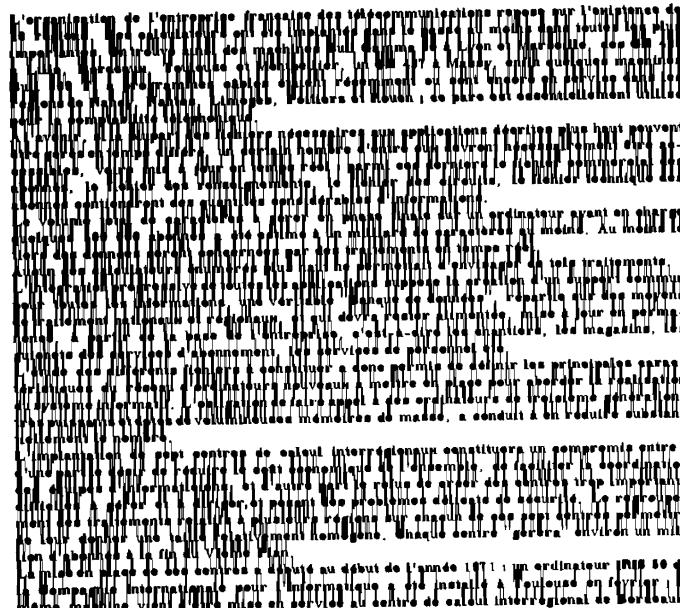
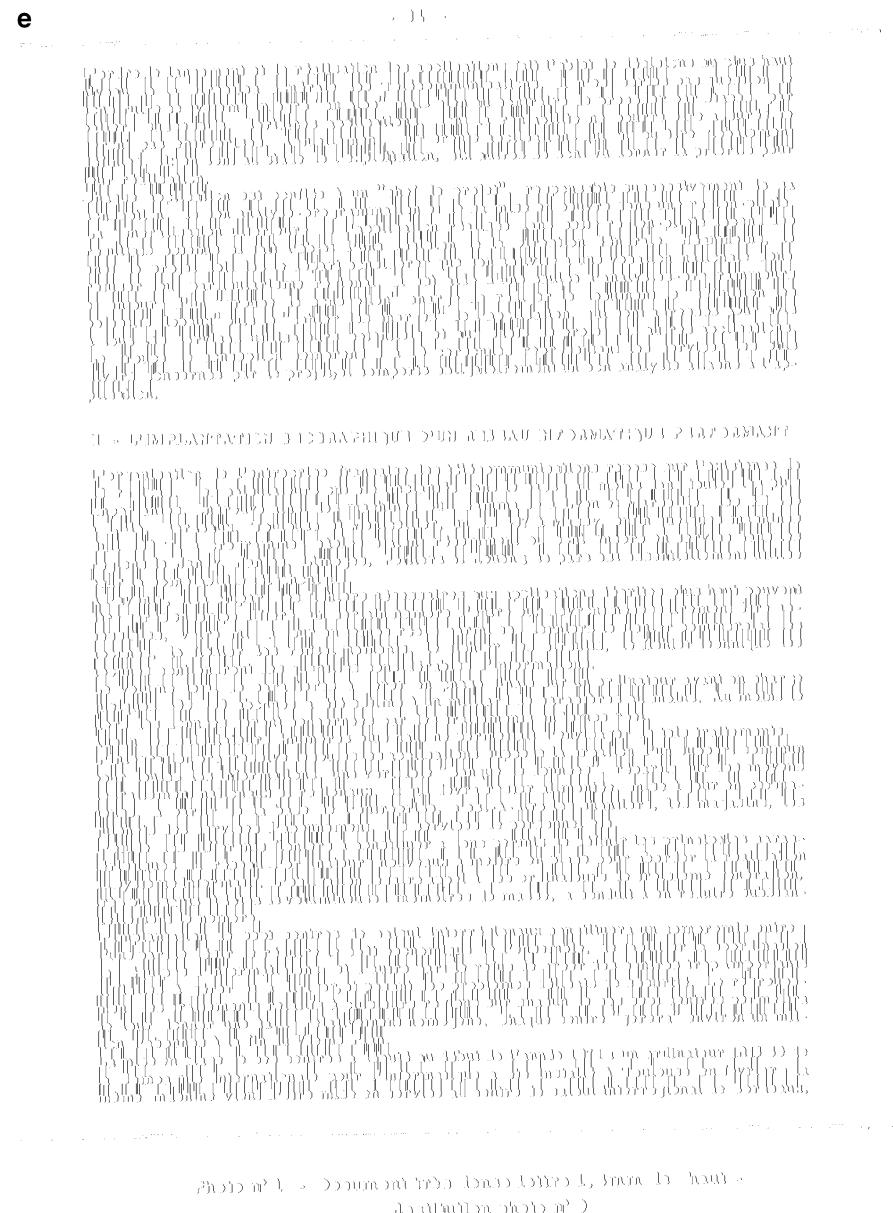


Photo n° 1 - Document très dense lettre 1,5mm de haut -  
Restitution photo n° 9

**Fig. 4.14** (continued)



**Fig. 4.14** Page orientation detection based on black-to-white transitions [78]. (a) Original binary image; (b) image after horizontal smoothing; (c) pixels that correspond to black-to-white transitions in vertical direction of the horizontally smoothed image ( $TPix1 =$  total number of pixels = 83,044); (d) image after vertical smoothing; (e) pixels that correspond to black-to-white transitions in horizontal direction of the vertically smoothed image ( $TPix2 =$  total number of pixels = 141,782)

a method that determines a direction that is the most asymmetric is provided. This direction is the most useful for the determination of text orientation, given a training dataset of documents of known orientation. This work can be adapted for use in automated mail processing or to determine the orientation of checks in automated teller machine envelopes, scanned or copied documents, documents sent via facsimile, etc.

A one-step skew and orientation detection method has been also proposed based on maximization of variance of transition counts (pixel from black to white or white to black) or on text line detection using a geometric text line model ([70] by van Beusekom et al.).

## Deskew and Deslant

Document skew is often introduced during the document capturing process and may seriously affect the performance of subsequent stages of segmentation and recognition. An overview of the key document image skew detection techniques is given in Table 4.4.

Document skew detection methodologies proposed in the literature (see survey paper [31]) usually assume the skew angle range to be from  $-5^\circ$  to  $5^\circ$  and fall broadly into the following categories:

### Projection Profile-Based Skew Detection Methods

According to these techniques, a series of horizontal projection profiles are calculated as we rotate the document page at a range of angles. The optimization of an objective function for a given skew angle leads to the actual document skew. As it can be observed in Fig. 4.15, the horizontal profiles tend to have consecutive local minima and maxima (discrete hills that correspond to text lines) as the document is rotated near  $-\theta_d$  angle, where  $\theta_d$  is the skew of the original document image. As an objective function that needs to be minimized, we can use the energy function  $A(\theta)$  which is defined as follows:

$$A(\theta) = \sum_{i=1}^m c_i^2(\theta) \quad (4.23)$$

where  $m$  is the number of horizontal profile bins and  $\theta$  the angle we rotate the document. At the example of Fig. 4.15, the original image has a skew of  $\sim -3^\circ$  and we calculate  $A(0^\circ) = 5.397.158$ ,  $A(1^\circ) = 5.537.790$ ,  $A(2^\circ) = 6.890.72653$ , and  $A(3^\circ) = 8.927.232$ .

In order to reduce high computational cost as well as to optimize search strategy, several variations of this method have been proposed. Baird [8] proposed a technique to minimize the number of points to be projected. For each connected component, only the midpoint of the bottom side of the bounding box is projected. The sum of the squares of the profile bins is used as the energy function to be

**Table 4.4** Overview of key document image skew detection techniques

Reference	Category	Short description	Remarks
Baird [8]	Projection profiles	For each connected component, only the midpoint of the bottom side of the bounding box is projected. The sum of the squares of the profile bins is used as the energy function to be optimized	An iterative coarse-to-fine process was also introduced in order to speed up the process
Akiyama and Hagita [1]	Projection profiles	The image is partitioned vertically into a number of strips. The horizontal projection profile is then calculated for each strip. Correlation of the profiles of the neighboring strips	Fast but not so accurate results
Li et al. [41]	Projection profiles	Using wavelet decomposition, the horizontal structure of the document image is extracted and emphasized, which improves the accuracy of profile projection method	Fast, language independent, and can deal with document images with complex layouts and multiple font size
Amin and Fischer [2]	Hough transform	Connected components of similar dimensions are grouped together. Hough transform is performed after dividing each group of connected components into vertical segments	Fast and accurate for angles of up to 45°
Singh et al. [60]	Hough transform	Accelerates the application of Hough transform by reducing the number of image pixels using a modified form of block adjacency graph	Significant enhancement of the speed without affecting the skew detection accuracy
O'Gorman [53]	Nearest-neighboring clustering	Nearest-neighbor clustering uses the K-neighbors for each connected component	The calculated histogram peak may not be very accurate
Lu and Tan [45]	Nearest-neighboring clustering	Size restriction is introduced to the detection of nearest-neighbor pairs. The slopes of the chains with a largest possible number of nearest-neighbor pairs are calculated	Able to deal with documents of different scripts such as English, Tamil, and Chinese
Cao and Li [11]	Nearest-neighboring clustering	The bottom center of the bounding box of a connected component is regarded as an eigen-point. Skew is determined according to the relations between the successive eigen-points in every text line	It reduces the computing complexity and achieves better precision

(continued)

**Table 4.4** (continued)

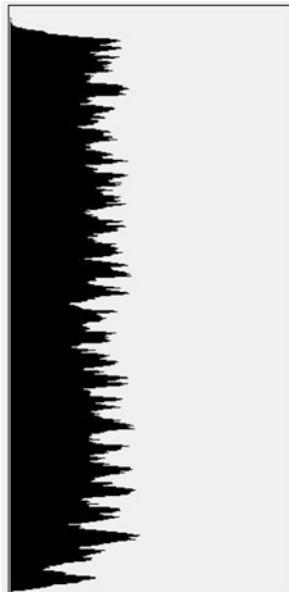
Reference	Category	Short description	Remarks
Yan [76]	Cross-correlation	The accumulated correlation for many pairs of lines is calculated. The optimal shift and the distance between each pair of lines are used for skew estimation	It can be used for binary, grayscale, and color images
Gatos et al. [26]	Cross-correlation	The image is first smoothed and then only the information existing in two or more vertical lines is used for skew detection	It can also be used for text line detection
Liu et al. [44]	Segmentation	It is based on detecting the borderlines of objects, either in text or non-text regions. Borderlines are extracted from the borders of large connected components by using a run-length-based method	Efficient for complex documents with horizontal and vertical text layout, predominant non-text regions, or sparse text regions
Chou et al. [17]	Segmentation	It is based on piecewise coverings of objects by parallelograms. The angle at which objects are best covered corresponds to the document skew angle	It is vulnerable to images that are rich in background noise
Fan et al. [25]	Segmentation	The Rectangular Active Contour Model (RAC Model) is used by assuming the boundary of a document image's content area as a rectangle represented by a five-tuple	Robust to noise and does not require any de-noising preprocessing
Chen and Wang [15]	Color documents	It determines the variation of color-transition count at each angle and the angle of maximal variation is regarded as the skew angle	Tested on 100 color images rotated from $-45^\circ$ to $45^\circ$
Makridis et al. [47]	Color documents	It consists of four main stages: color reduction, text localization, document binarization, and skew correction	It can be applied to color, grayscale, and binary documents

optimized. In order to speed up the search of the optimum angle, an iterative coarse-to-fine process was also introduced. In [1], Akiyama and Hagita proposed to partition the image vertically into a number of strips. The horizontal projection profile is then calculated for each strip. The skew angle is estimated from the correlation of the profiles of the neighboring strips. This yields fast but not so accurate result. A projection profile-based method using wavelet decompositions is presented in [41] by Li et al. Using the wavelet decomposition, the horizontal structure of the document image can be extracted and emphasized, which improves

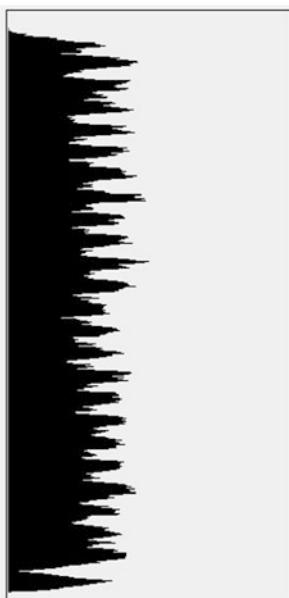
the accuracy of profile projection method. On the other hand, the computational cost is less than one tenth of that of the original after 2-level wavelet decomposition. Experimental results show that the proposed algorithm is language independent and can deal with document images with complex layouts and multiple font size.

**a***Gentlemen of the Society,*

While hesitating in the performance of your demand, I could not forget the great injunction, "never shrink from duty," which should govern every man, in whatever situation he may be placed, either by the allotment of Providence, or the suffrage of friends; but when I reflect on how little can be said upon the usual topic, that is new, interesting or instructive, after what has already been advanced in the numerous addresses which have poured in from all quarters—from not only the first farmers, but the first statesmen and literary characters of the nation, I become sensible of the difficulty you have laid me under.—Eminent men, with President Madison at their head, have led up the genius of our art, and placed her upon that lofty eminence, on which she ought long ago to have been seated. In their course thither, they have strewed the path with some of the finest flowers of eloquence; the distinguished virtue, happiness, and independence of the Farmer's vocation, and the utility which flows from our societies, have received at their hands, a coloring so just, that I shall not presume to retouch these subjects.

**b***Gentlemen of the Society,*

While hesitating in the performance of your demand, I could not forget the great injunction, "never shrink from duty," which should govern every man, in whatever situation he may be placed, either by the allotment of Providence, or the suffrage of friends; but when I reflect on how little can be said upon the usual topic, that is new, interesting or instructive, after what has already been advanced in the numerous addresses which have poured in from all quarters—from not only the first farmers, but the first statesmen and literary characters of the nation, I become sensible of the difficulty you have laid me under.—Eminent men, with President Madison at their head, have led up the genius of our art, and placed her upon that lofty eminence, on which she ought long ago to have been seated. In their course thither, they have strewed the path with some of the finest flowers of eloquence; the distinguished virtue, happiness, and independence of the Farmer's vocation, and the utility which flows from our societies, have received at their hands, a coloring so just, that I shall not presume to retouch these subjects.

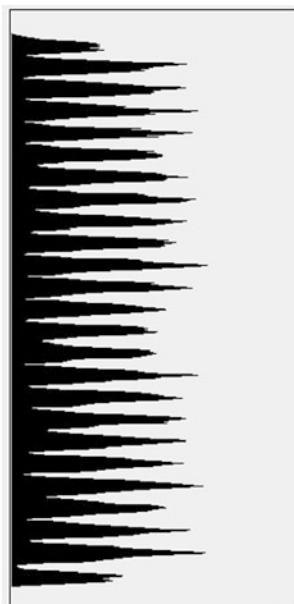


**Fig. 4.15** (continued)

**c**

*Gentlemen of the Society,*

While hesitating in the performance of your demand, I could not forget the great injunction, "never shrink from duty," which should govern every man, in whatever situation he may be placed, either by the allotment of Providence, or the suffrage of friends; but when I reflect on how little can be said upon the usual topick, that is new, interesting or instructive, after what has already been advanced in the numerous addresses which have poured in from all quarters—from not only the first farmers, but the first statesmen and literary characters of the nation, I become sensible of the difficulty you have laid me under.—Eminent men, with President Madison at their head, have led up the genius of our art, and placed her upon that lofty eminence, on which she ought long ago to have been seated. In their course thither, they have strewed the path with some of the finest flowers of eloquence; the distinguished virtue, happiness, and independence of the Farmer's vocation, and the utility which flows from our societies, have received at their hands, a coloring so just, that I shall not presume to retouch these subjects.

**d**

*Gentlemen of the Society,*

While hesitating in the performance of your demand, I could not forget the great injunction, "never shrink from duty," which should govern every man, in whatever situation he may be placed, either by the allotment of Providence, or the suffrage of friends; but when I reflect on how little can be said upon the usual topick, that is new, interesting or instructive, after what has already been advanced in the numerous addresses which have poured in from all quarters—from not only the first farmers, but the first statesmen and literary characters of the nation, I become sensible of the difficulty you have laid me under.—Eminent men, with President Madison at their head, have led up the genius of our art, and placed her upon that lofty eminence, on which she ought long ago to have been seated. In their course thither, they have strewed the path with some of the finest flowers of eloquence; the distinguished virtue, happiness, and independence of the Farmer's vocation, and the utility which flows from our societies, have received at their hands, a coloring so just, that I shall not presume to retouch these subjects.



**Fig. 4.15** Horizontal projection profiles of a document image rotated at angle  $\theta$ : (a) original image having a skew of  $\sim -3^\circ$  and rotated at  $0^\circ$ , (b) original image rotated at  $1^\circ$ , (c) original image rotated at  $2^\circ$ , and (d) original image rotated at  $3^\circ$

## Hough Transform-Based Skew Detection Methods

Since usually the document skew is also the skew of the document text lines, the use of the Hough transform (HT) for detecting straight lines has been proposed as a skew detection tool (see survey paper [31]). Each black pixel is mapped to the Hough space ( $\rho, \theta$ ) and the skew is estimated as the angle in the parameter space that gives the maximum sum of squares of the gradient along the  $\rho$  component. Several research efforts focus on improving the computational efficiency of this method. The image can be downsampled and transformed into a “burst image.” The “burst image” is build by replacing each vertical black run with its length and placing this value in the bottommost pixel of the run. HT is then applied only to the pixels of the burst image that have value less than a threshold in order to discard non-text components. HT can be performed on a selected square of the document where only bottom pixels of candidate objects are preserved. According to the skew detection methodology proposed in [2] by Amin and Fischer, connected components of similar dimensions are grouped together and a skew angle is calculated for each group. HT is performed after dividing each group of connected components into vertical segments of approximately the width of one connected component and stores only the bottom rectangle in each segment. The skew angle for the entire page is then determined by choosing the most frequent angle, after applying a weighting factor to each angle to account for the number of points used in its calculation. In [60], the two major issues of the Hough transform, slow speed and high memory requirement, are addressed by Singh et al. The overall speed of skew detection is enhanced by using a preprocessing stage where the number of image pixels is greatly reduced using a modified form of block adjacency graph.

## Nearest-Neighbor Clustering-Based Skew Detection Methods

According to these approaches, spatial relationships and mutual distances of connected components are used to estimate the page skew. The direction vector of all nearest-neighbor pairs of connected components is accumulated in a histogram and the peak in the histogram gives the dominant skew. This method is generalized in [53] by O’Gorman where nearest-neighbor clustering uses the K-neighbors for each connected component. Since connections between neighboring components may be made both within and across text lines, the calculated histogram peak may not be very accurate. An improved nearest-neighbor-based approach for skew estimation is presented in [45] by Lu and Tan. According to this approach, size restriction is introduced to the detection of nearest-neighbor pairs. Then, the chains with a largest possible number of nearest-neighbor pairs are selected, and their slopes are computed to give the skew angle of document image. In [11] by Cao and Li, the bottom center of the bounding box of a connected component is regarded as an eigen-point. According to the relations between the neighboring eigen-points in every text line, the eigen-points laid on the baseline are extracted as sample points. Then, these samples are adopted by the least squares method to calculate the baseline direction. The average of these baseline directions corresponds to the skew angle of the whole document image. To reduce the computing cost, a suitable subregion that only contains the pure text content is selected for the skew detection method.

### Cross-Correlation-Based Skew Detection Methods

These approaches are based on measuring vertical deviations among foreground pixels along the document image in order to detect the page skew. Cross-correlation between lines at a fixed distance is used in [76] by Yan. This method is developed based on the following observation. The correlation between two vertical lines of a skewed document image is maximized if one line is shifted relatively to the other such that the character baseline levels for the two lines are coincident. The accumulated correlation for many pairs of lines can provide a precise estimation of the page skew. The skew angle can be determined from the optimal shift and the distance between each pair of lines. Cross-correlation  $R_1(x, s)$  can be defined as

$$R_1(x, s) = \sum_{y=s}^{I_y-S} B(x + d, y + s)B(x, y) \quad (4.24)$$

where  $B(x, y)$  the binary image with  $B(x, y) = 1$  represents a dark pixel in a character and  $B(x, y) = 0$  a bright pixel in the background,  $x \in [0, I_x]$ ,  $y \in [0, I_y]$ ,  $s$  the shift between two vertical lines of the image at  $x$  and  $x + d$ , and  $s \in [-S, S]$ . The accumulated cross-correlated function  $R(s)$  is defined as follows:

$$R(s) = \sum_{x=0}^{I_x-d} R_1(x, s) \quad (4.25)$$

In [26] by Gatos et al., the image is first smoothed by a horizontal run-length algorithm [73], and then, only the information existing in two or more vertical lines is used for skew detection. Based only on this information, a correlation matrix is constructed. The skew angle is determined from the global maximum of a projection derived from this matrix. After the skew angle is determined, a text line detection function is also defined and its local maxima give the positions of the text lines.

### Segmentation-Based Skew Detection Methods

These methodologies first proceed to document image segmentation in order to detect image objects (e.g., text or non-text regions, text lines) and then estimate the skew angle based on analyzing the skew characteristics of each object. A skew detection method for complex document images based on detecting the borderlines of objects, either in text regions or in non-text regions, is proposed in [44] by Liu et al. Borderlines are extracted from the borders of large connected components in a document image by using a run-length-based method [73]. After filtering out nonlinear borderlines, a fast iteration algorithm is applied to optimize each linear borderline's directional angle. Finally, the weighted median value of all the directional angles is calculated as the skew angle of the whole document. A skew estimation algorithm based on piecewise coverings of objects by parallelograms was proposed in [17] by Chou et al. In this approach, the document image is divided into several non-overlapping regions and the objects within each region

are covered by parallelograms at various angles. The angle at which objects are best covered corresponds to skew angle of the document. The Rectangular Active Contour Model (RAC Model) has been recently proposed in [25] by Fan et al. for skew detection by assuming the boundary of a document image's content area as a rectangle represented by a five-tuple. This assumption is satisfied with many document images, such as journals, magazines, and newspaper. In the model, a rectangular shape constraint is imposed on the zero-level set of a scalar Lipschitz continuous function in Chan–Vese Model (C–V Model). The variational method is applied to minimize the energy functional to get the parameters of the rectangle close to the boundary of the content region. The proposed algorithm estimates the skew angle with a global shape feature of content on a document image, so it is robust to noise and does not require any de-noising preprocessing.

A one-step skew and orientation detection method is proposed in [70] by van Beusekom et al. and is based on text line detection using a geometric text line model.

### **Skew Detection Methods for Color Documents**

The skew detection and correction approach of [15] by Chen and Wang has been designed for color documents. This approach first determines variation of color-transition count at each angle and the angle of maximal variation is regarded as the skew angle. Then, a scanning-line model reconstructs the image. A technique for global and local skew detection in complex color documents is proposed in [47] by Makridis et al. It consists of four main stages: color reduction, text localization, document binarization, and skew correction. Skew correction is achieved by detecting the direction of connection of the connected components in the binary images.

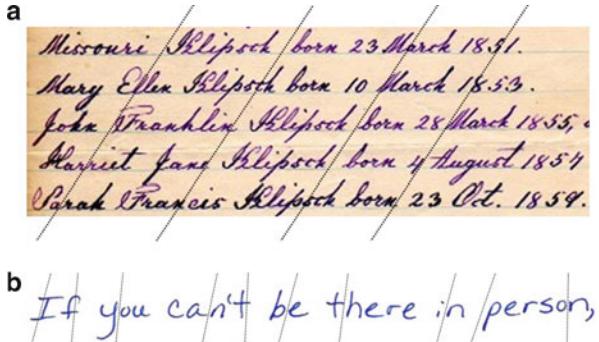
By the term “character slant,” we imply the angle in degrees clockwise from vertical at which the characters are drawn (see Fig. 4.16). Character slant estimation can be very helpful in handwritten text processing. Knowing the slant’s value, we can correct it in order to facilitate processing and recognition. In addition to that, the character slant is thought to be very important information which can help to point out the writer of a text. Document slant may be uniform (see Fig. 4.16a) or nonuniform (see Fig. 4.16b) along the text line or even within the same word. An overview of key slant estimation techniques is given in Table 4.5.

Slant estimation methodologies proposed in the literature fall broadly into the following main categories.

### **Slant Estimation Methods that Analyze Near-Vertical Strokes**

According to the approach of Bozinovich and Srihari [9], for a given word, all horizontal lines which contain at least one run of length greater than a parameter  $M$  (depending on the width of the letters) are removed. Additionally, all horizontal strips of small height are removed. By deleting these horizontal lines, only orthogonal window parts of the picture remain in the text. For each letter, the parts that remain (after the deletion) are those that contribute to the word slant. For each of these parts, we estimate the angle between the line indicated by the centers of gravity for its upper and lower halves and the vertical side of the page.

**Fig. 4.16** Slanted handwriting examples: (a) uniform slanted text and (b) nonuniform slanted text



The mean value of these slants is the overall text's character slant. In the approach of Kim and Govindaraju [34], vertical and near-vertical lines are extracted by tracing chain code components using a pair of one-dimensional filter. Coordinates of the start and end points of each vertical line extracted provide the slant angle. Global slant angle is the average of all the angle of the lines, weighted by their length direction since the longer line gives more accurate angle than the shorter one.

### Projection Profile-Based Slant Estimation Methods

Vinciarelli and Luettin [71] have proposed a deslanting technique based on the hypothesis that the word has no slant when the number of columns containing a continuous stroke is maximum. The word image is artificially slanted at different slant angles, and for each angle  $\alpha$ , the following vertical profile is calculated:

$$H_\alpha(m) = \frac{h_\alpha(m)}{\Delta y_\alpha(m)} \quad (4.26)$$

where  $h_\alpha(m)$  is the vertical density (number of foreground pixels per column) in column  $m$ , and  $\Delta y_\alpha(m)$  the distance between the highest and lowest pixel in the same column. If the column  $m$  contains a continuous stroke,  $H_\alpha(m) = 1$ , otherwise  $H_\alpha(m) \in [0,1]$ . For each angle  $\alpha$ , the following function is calculated:

$$S(\alpha) = \sum_{\{i : H_\alpha(i)=1\}} h_\alpha(i)^2 \quad (4.27)$$

The angle  $\alpha$  giving the highest value of  $S$  is taken as the slant estimate.

Kavallieratou et al. [33] have proposed a slant removal algorithm based on the use of the vertical projection profile of word images and the Wigner–Ville distribution (WVD). The word image is artificially slanted, and for each of the extracted word images, the vertical histogram as well as the WVD of these histograms is calculated. The curve of maximum intensity of the WVDs corresponds to the histogram with the most intense alternations and as a result to the dominant word slant.

**Table 4.5** Overview of key slant estimation techniques

Reference	Category	Short description	Remarks
Bozinovich and Srihari [9]	Analysis of near-vertical strokes	After removing character parts that do not contribute to the word slant, the angle calculated in several image parts is used to estimate the global word slant	Several parameters have to be manually tuned
Kim and Govindaraju [34]	Analysis of near-vertical strokes	Vertical and near-vertical lines are extracted by tracing chain code components using a pair of one-dimensional filter. Global word angle is calculated by the weighted average of the angle of these lines	Part of a chain code-based handwritten word recognition system
Vinciarelli and Luettin [71]	Projection profiles	It is based on the hypothesis that the word has no slant when the number of columns containing a continuous stroke is maximum	It is shown to improve the recognition rate by 10 % relative to other normalization methods
Kavallieratou et al. [33]	Projection profiles	It is based on the use of the vertical projection profile of word images and the Wigner–Ville distribution (WVD) of this profile	Tested in English and Modern Greek samples of more than 500 writers
Kimura et al. [36]	Chain code	The average slant of the handwritten word is estimated using the chain code histogram of border pixels	The slant tends to be underestimated when the absolute of the slant is close or greater than 45°
Ding et al. [19]	Chain code	A modification of [36] using an 8-directional chain code	12- and 16-directional chain code are also examined but it is shown that the slant is overestimated
Ding et al. [20]	Chain code	Cumulative frequency distribution of chain code is used for local slant estimation	Several variations are also proposed in order to improve the accuracy

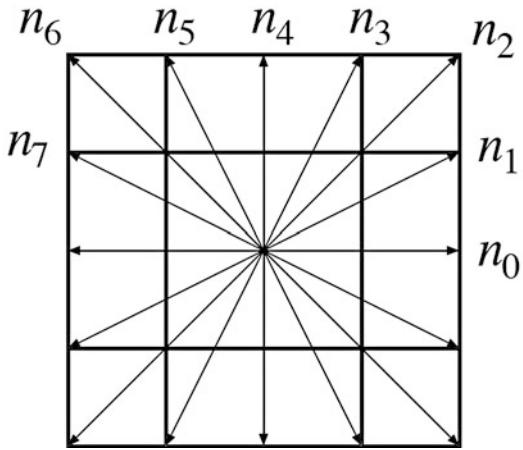
### Chain Code-Based Slant Estimation Methods

Average slant of a handwritten word can be easily estimated using the chain code histogram of border pixels. According to Kimura et al. [36], the slant  $\theta$  of a handwritten word can be estimated as follows:

$$\theta = \tan^{-1} \left( \frac{n_1 - n_3}{n_1 + n_2 + n_3} \right) \quad (4.28)$$

where  $n_i$  is the number of chain elements at an angle of  $i \times 45^\circ$  (“/” or “|” or “\”). Although this method usually gives good estimate of the word slant, the slant tends to be underestimated when the absolute of the slant is close or greater than 45°.

**Fig. 4.17** 8-directional quantization of chain code



To solve the problem, Ding et al. [19] proposed a slant detection method using an 8-directional chain code. Obtained chain code is then quantized to eight directions as shown in Fig. 4.17. Each direction is numbered 0–7 in counterclockwise, and  $n_i$  denotes the number of chain code elements in direction  $i$  (see Fig. 4.17). The slant estimator of the 8-directional chain code method is given as follows:

$$\theta = \tan^{-1} \left\{ \frac{(2n_1 + 2n_2 + n_3) - (n_5 + 2n_6 + 2n_7)}{(n_1 + 2n_2 + 2n_3) + 2n_4 + (2n_5 + 2n_6 + n_7)} \right\} \quad (4.29)$$

where  $(2n_1 + 2n_2 + n_3)$  is the sum of horizontal projection of the element 1, 2, 3;  $(n_5 + 2n_6 + 2n_7)$  is the sum of horizontal projection of the element 5, 6, 7; and the denominator is the sum of vertical projection of the element 1–7. Chain code methods of 12 and 16 directions are also examined in [19], but the experimental results show that these methods tend to overestimate the slant.

A local slant estimation using cumulative frequency distribution of chain code has been proposed by Ding et al. [20]. Local slant estimator is defined as a function of horizontal coordinate  $x$  as follows:

$$\theta(x) = \tan^{-1} \left[ \frac{n_1(x) - n_3(x)}{n_1(x) + n_2(x) + n_3(x)} \right] \quad (4.30)$$

where  $n_i(x)$  ( $i = 1, 2, 3$ ) is the frequency distribution of chain code elements at direction of  $i \times 45^\circ$  in  $[x - \delta_x, x + \delta_x]$  calculated as follows:

$$n_i(x) = s_i(x + \delta_x) - s_i(x - \delta_x - 1) \quad (4.31)$$

where  $s_i(x)$  is the cumulative frequency distribution number of chain code elements at direction of  $i \times 45^\circ$  in  $[0, x]$ . Parameter  $\delta_x$  is determined experimentally depending on the input image. Then,  $\tan \theta(x)$  is smoothed by the mean filter of adjacent three pixels. To improve the accuracy of local slant estimation, three variations of the

above idea (simple iterative method, high speed iterative method, and 8-directional chain code method) are also proposed in [20].

## Dewarping

Document image acquisition by a flatbed scanner or a digital camera often results in several unavoidable image distortions (see Fig. 4.18) due to the form of printed material (e.g., bounded volumes), the camera setup, or environmental conditions (e.g., humidity that causes page shrinking). Text distortions not only reduce document readability but also affect the performance of subsequent processing such as document layout analysis and character recognition. Many different techniques have been proposed for document image rectification that can be classified into two main categories based on (1) 3-D document shape reconstruction [12,42,80] and (2) 2-D document image processing [10,37,46,48,63,75,79]. Techniques of the former category obtain the 3-D information of the document image using special setup or reconstruct the 3-D model from information existing in document images. On the other hand, techniques in the latter category do not depend on auxiliary hardware or prior information but they only rely on 2-D information. An overview of document image dewarping techniques is given in Table 4.6.

### 3-D Document Shape-Based Methods

In this category, rectification techniques rely upon extraction of the 3-D information of the document and they can be further divided into two subcategories. Techniques of the first subcategory obtain the 3-D shape of the document image using special equipment such as laser scanners, stereo cameras, or structured light setups. The dependence on special equipment prevents these techniques from being used in an unconstrained environment. On the other hand, techniques of the second subcategory reconstruct the 3-D model from information existing in document images. Cao et al. [12] propose a method to rectify warping distortions in document images by constructing a cylinder model. Apart from the cylinder shape assumption, they also have a limitation on the pose that requires the image plane to be parallel to the generatrix of the page cylinder. Liang et al. [42] model the page surface by curved developable surfaces to estimate the 3-D shape of the page using texture flow fields. This method is based on the assumptions that the document is either flat or smoothly curved and the camera is a standard pinhole camera. Finally, some methods are based on a shape-from-shading formulation in order to reconstruct the 3-D shape of the document's surface (e.g., L. Zhang et al. [80]). These techniques require knowledge of lighting which in most of the cases is unknown.

### 2-D Document Image Processing-Based Methods

In this category, rectification techniques rely on the use of 2-D information available in document images. The majority of these rectification techniques are based on the detection of distorted text lines at the original document image which is a well-known hard task. Some of these techniques propose a method to straighten distorted text lines by fitting a model to each text line. Lavialle et al. [37] use an analytical

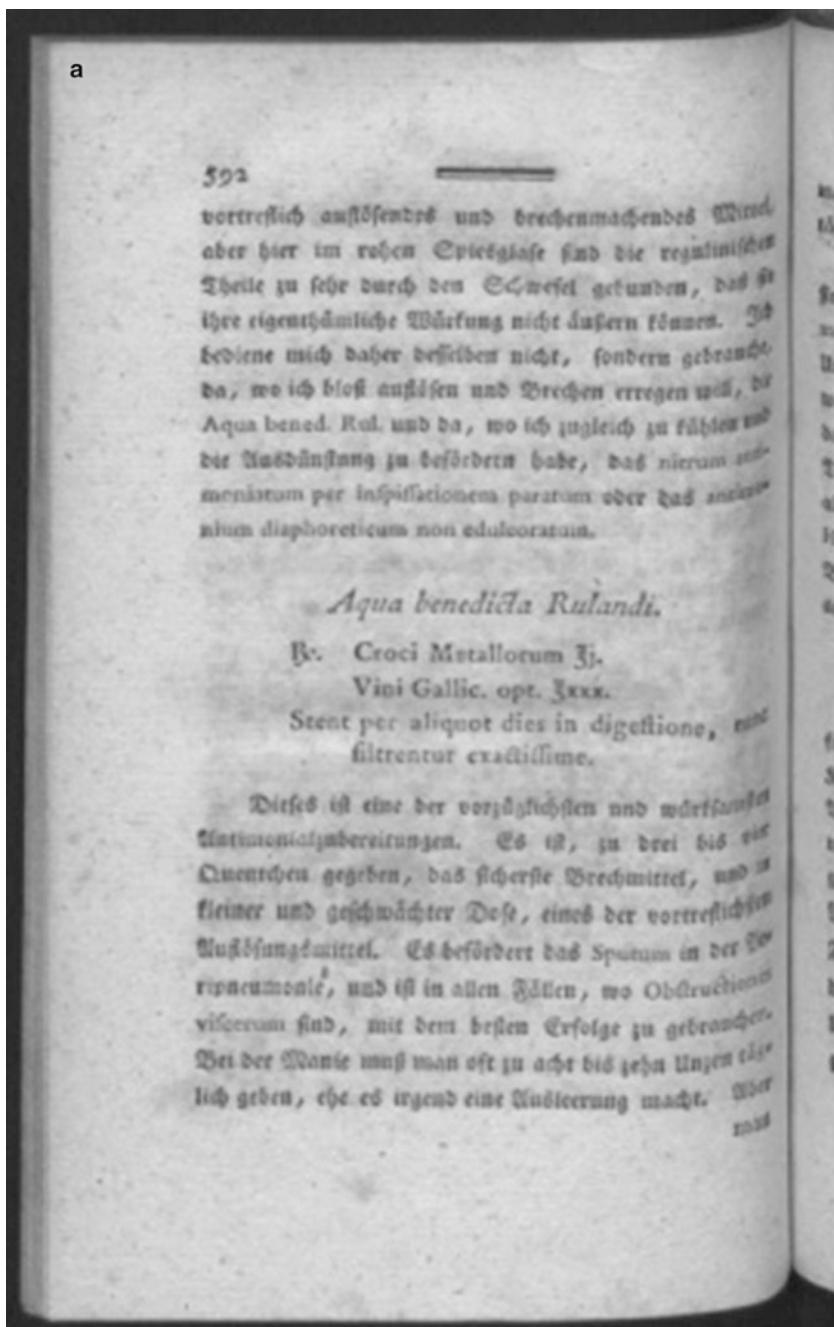
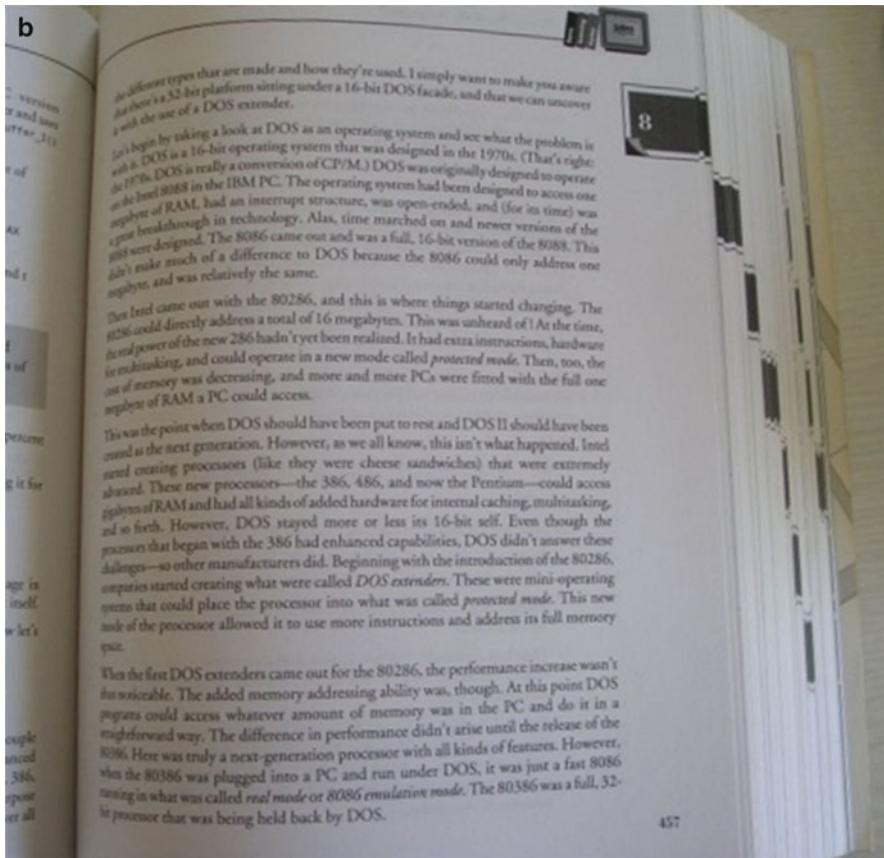


Fig. 4.18 (continued)



**Fig. 4.18** Examples of document images captured by (a) flatbed scanner and (b) digital camera

model with cubic B-splines, Wu and Agam [75] use a nonlinear curve for each text line, and L. Zhang and Tan [79] represent the text lines using natural cubic splines. The above mentioned techniques suffer from various limitations. Specifically, the approach in [37] is not efficient in the case of inhomogeneous line spacing and method [75] is based on several heuristics, while it requires that the user should interactively specify the four corner points of the warped image which is not practical and cannot handle nonuniform columns in the target mesh, as well. In [79], L. Zhang and Tan assume that the book spine is found along iso-parametric lines. In a recent work of Stamatopoulos et al. [63], a goal-oriented rectification methodology to compensate for undesirable document image distortions aiming to improve the OCR result is presented. The approach relies upon a coarse-to-fine strategy. It first detects words and text lines to rectify the document image in a coarse scale and then further normalize individual words in finer detail using baseline correction. Coarse rectification is

**Table 4.6** Overview of key document image dewarping techniques

Reference	Category	Short description	Remarks
Cao et al. [12]	3-D document shape	Rectifies warping distortions in document images by constructing a cylinder model. The skeleton of each horizontal text line is extracted to help estimate the parameter of the model and rectify the images	It does not require any information about the image formation process or any specific stereographic device
Liang et al. [42]	3-D document shape	It models the page surface by curved developable surfaces to estimate the 3-D shape of the page using texture flow fields	The document has to be either flat or smoothly curved and the camera a standard pinhole camera
Zhang et al. [80]	3-D document shape	It is based on a shape-from-shading formulation in order to reconstruct the 3-D shape of the document's surface	Does not produce satisfactory results in some cases of images with degradations
Lavialle et al. [37]	2-D document image processing	It is based on the detection of distorted text lines at the original document image using an analytical model with cubic B-splines	Not efficient in the case of inhomogeneous line spacing
Wu and Agam [75]	2-D document image processing	It is based on the detection of distorted text lines using a nonlinear curve for each text line	Based on several heuristics and requires user interaction
L. Zhang and Tan [79]	2-D document image processing	It represents the text lines using natural cubic splines	Assumes that the book spine is found along iso-parametric lines
Stamatopoulos et al. [63]	2-D document image processing	First detects words and text lines to rectify the document image in a coarse scale and then further normalize individual words in finer detail using baseline correction	A goal-oriented rectification aiming to improve the OCR result
Ulges et al. [69]	2-D document image processing	Estimates quadrilateral cell for each letter based on local baseline finding and then maps to a rectangle of corrected size and position in the dewarped image	Only for documents with straight lines that are approximately equally spaced and sized and the spacing between words is not large
Lu et al. [46]	2-D document image processing	Divides images into multiple quadrilateral patches based on the exploitation of the vertical stroke boundaries and text baselines	Based on several heuristics and limited on documents printed in Latin-based languages
Brown and Tsoi [10]	2-D document image processing	It uses document boundary interpolation to correct geometric distortions and shading artifacts present in images of art-like materials	Limited to some specific document distortions
Masalovitch and Mestetskiy [48]	2-D document image processing	Approximates deformation of interlinear spaces of image based on elements of image's skeleton that lie between the text lines	Sensitive to the approximation of vertical border deformation in text blocks

accomplished with the aid of a computationally low-cost transformation which addresses the projection of a curved surface to a 2-D rectangular area.

A few more rectification techniques also rely on text line detection but, in contrast with the above mentioned techniques, they emphasize on baseline finding. Ulges et al. [69] estimate quadrilateral cell for each letter based on local baseline finding and then map to a rectangle of corrected size and position in the dewarped image. Their method is not generic since it is based on the assumption that the original page contains only straight lines that are approximately equally spaced and sized and spacing between words is not large. Lu et al. [46] restore the document by dividing images into multiple quadrilateral patches based on the exploitation of the vertical stroke boundaries (VSBS) and text baselines. This method is based on several heuristics and is limited on documents printed in Latin-based languages.

There are also rectification techniques that do not rely on the detection of distorted text lines but they aim to find spatial transformations between the warped and dewarped document images by analyzing the 2-D content such as document boundaries or known reference points. Brown and Tsoi [10] use document boundary interpolation to correct geometric distortions and shading artifacts present in images of art-like materials. They use a physical pattern to guide the uniform parameterization, so it is limited to some specific document distortions. Masalovitch and Mestetskiy [48] approximate deformation of interlinear spaces of image based on elements of image's skeleton that lie between the text lines. This method is sensitive to the approximation of vertical border deformation in text blocks, which diminish accuracy.

---

## Conclusion

In this chapter, we introduced basic image processing algorithms used in document image analysis while the focus was given on the techniques used for document image binarization, enhancement, and normalization. Document image binarization is used to separate the text from the background regions. Document image enhancement aims to improve the quality of document images by diminishing artifacts such as low contrast and uneven background illumination, bleed-through and shadow effects, damaged characters, and noisy black borders. Document image normalization refers to the task of restoring the document image horizontal alignment after correcting possible page skew, character slant, warping, and perspective distortions. For all these tasks, we categorized the main approaches while we presented an overview of key techniques using comparative tables.

---

## Cross-References

- ▶ [Document Creation, Image Acquisition and Document Quality](#)
- ▶ [Handprinted Character and Word Recognition](#)
- ▶ [Page Segmentation Techniques in Document Analysis](#)
- ▶ [Text Segmentation for Document Recognition](#)

## References

1. Akiyama T, Hagita N (1990) Automated entry system for printed documents. *Pattern Recognit* 23(11):1141–1154
2. Amin A, Fischer S (2000) A document skew detection method using the Hough transform. *Pattern Anal Appl* 3(3):243–253
3. Aradhye HB (2005) A generic method for determining up/down orientation of text in roman and non-roman scripts. *Pattern Recognit* 38:2114–2131
4. Avila BT, Lins RD (2004) A new algorithm for removing noisy borders from monochromatic documents. In: ACM-SAC'2004, Cyprus, Mar 2004. ACM, pp 1219–1225
5. Avila BT, Lins RD (2004) Efficient removal of noisy borders from monochromatic documents. In: ICIAR 2004, Porto, Portugal. LNCS 3212, pp 249–256
6. Badekas E, Papamarkos N (2007) Optimal combination of document binarization techniques using a self-organizing map neural network. *Eng Appl Artif Intell* (Elsevier) 20:11–24
7. Badekas E, Nikolaou N, Papamarkos N (2006) Text binarization in color documents. *Int J Imaging Syst Technol* 16(6):262–274
8. Baird HS (1987) The skew angle of printed documents. In: SPSE 40th conference and symposium on hybrid imaging systems, Rochester, pp 21–24
9. Bozinovich A, Srihari A (1989) Off-line cursive script word recognition. *Trans Pattern Anal Mach Intell* II(1):69–82
10. Brown MS, Tsoi YC (2006) Geometric and shading correction for images of printed materials using boundary. *IEEE Trans Image Process* 15(6):1544–1554
11. Cao Y, Li H (2003) Skew detection and correction in document images based on straight-line fitting. *Pattern Recognit Lett* 24(12):1871–1879
12. Cao H, Ding X, Liu C (2003) Rectifying the bound document image captured by the camera: a model based approach. In: 7th international conference on document analysis and recognition, Edinburgh, pp 71–75
13. Caprari RS (2000) Algorithm for text page up/down orientation determination. *Pattern Recognit Lett* 21:311–317
14. Chen Y, Leedham G (2005) Decompose algorithm for thresholding degraded historical document images. *IEE Vis Image Signal Process* 152(6):702–714
15. Chen YK, Wang JF (2001) Skew detection and reconstruction of color-printed document images. *IEICE Trans Inf Syst* E84-D(8):1018–1024
16. Cheriet M, Said JN, Suen CY (1998) A recursive thresholding technique for image segmentation. *IEEE Trans Image Process* 7(6):918–921
17. Chou CH, Chu SY, Chang F (2007) Estimation of skew angles for scanned documents based on piecewise covering by parallelograms. *Pattern Recognit* 40:443–455
18. Chou C-H, Lin W-H, Chang F (2010) A binarization method with learning-built rules for document images produced by cameras. *Pattern Recognit* 43(4):1518–1530
19. Ding Y, Kimura F, Miyake Y (2000) Slant estimation for handwritten words by directionally refined chain code. In: 7th international workshop on Frontiers in handwriting recognition (IWFHR 2000), Amsterdam, pp 53–62
20. Ding Y, Ohyama W, Kimura F, Shridhar M (2004) Local Slant estimation for handwritten English words. In: 9th international workshop on Frontiers in handwriting recognition (IWFHR 2004), Tokyo, Japan, pp 328–333,
21. Drira F, LeBourgeois F, Emptoz H (2011) A new PDE-based approach for singularity-preserving regularization: application to degraded characters restoration. *Int J Doc Anal Recognit.* doi:10.1007/s10032-011-0165-5
22. Dubois E, Pathak A (2001) Reduction of bleed-through in scanned manuscript documents. In: Proceedings of the image processing, image quality, image capture systems conference, Apr 2001, pp 177–180
23. Fadoua D, Le Bourgeois F, Emptoz H (2006) Restoring ink bleed-through degraded document images using a recursive unsupervised classification technique. In: 7th international workshop on document analysis systems, (DAS 2006), Nelson, New Zealand, pp 38–49

24. Fan KC, Wang YK, Lay TR (2002) Marginal noise removal of document images. *Pattern Recognit* 35(11):2593–2611
25. Fan H, Zhu L, Tang Y (2010) Skew detection in document images based on rectangular active contour. *Int J Doc Anal Recognit* 13(4):261–269
26. Gatos B, Papamarkos N, Chamzas C (1997) Skew detection and text line position determination in digitized documents. *Pattern Recognit* 30(9):1505–1519
27. Gatos B, Pratikakis I, Perantonis SJ (2006) Adaptive degraded document image binarization. *Pattern Recognit* 39:317–327
28. Gatos B, Pratikakis I, Perantonis SJ (2008) Efficient binarization of historical and degraded document images. In: 8th international workshop on document analysis systems (DAS'08), Nara, Sept 2008, pp 447–454
29. Haji MM, Bui TD, Suen CY (2009) Simultaneous document margin removal and skew correction based on corner detection in projection profiles. In: ICIAP 2009, Solerno, Italy. LNCS 5716, pp 1025–1034
30. Huang S, Ahmadi M, Sid-Ahmed MA (2008) A hidden Markov model-based character extraction method. *Pattern Recognit* 41(9):2890–2900
31. Hull JJ (1998) Document image skew detection: survey and annotated bibliography. In: Hull JJ, Taylor SL (eds) Document analysis systems II. World Scientific, Singapore/River Edge, pp 40–64
32. Kamel M, Zhao A (1993) Extraction of binary character/graphics images from grayscale document images. *Comput Vis Graph Image Process* 55:203–217
33. Kavallieratou E, Fakotakis N, Kokkinakis G (2001) Slant estimation algorithm for OCR systems. *Pattern Recognit* 34:2515–2522
34. Kim G, Govindaraju V (1997) A Lexicon driven approach to handwritten word recognition for real-time applications. *IEEE Trans Pattern Anal Mach Intell* 19(4):366–379
35. Kim I-K, Jung D-W, Park R-H (2002) Document image binarization based on topographic analysis using a water flow model. *Pattern Recognit* 35:265–277
36. Kimura F, Shridhar M, Chen Z (1993) Improvements of a lexicon directed algorithm for recognition of unconstrained handwritten words. In: 2nd international conference on document analysis and recognition (ICDAR 1993), Tsukuba City, Japan. Oct 1993, pp 18–22
37. Lavialle O, Molines X, Angella F, Baylou P (2001) Active contours network to straighten distorted text lines. In: International conference on image processing, Thessaloniki, pp 748–751
38. Le DX, Thoma GR (1996) Automated borders detection and adaptive segmentation for binary document images. In: International conference on pattern recognition, (ICPR 1996), Vienna, Austria, p III: 737–741
39. Le DS, Thoma GR, Wechsler H (1994) Automated page orientation and skew angle detection for binary document images. *Pattern Recognit* 27(10):1325–1344
40. Leung CC, Chan KS, Chan HM, Tsui WK (2005) A new approach for image enhancement applied to low-contrast-low-illumination IC and document images. *Pattern Recognit Lett* 26:769–778
41. Li S, Shen Q, Sun J (2007) Skew detection using wavelet decomposition and projection profile analysis. *Pattern Recognit Lett* 28:555–562
42. Liang J, DeMenthon D, Doermann D (2008) Geometric rectification of camera-captured document images. *IEEE Trans PAMI* 30(4):591–605
43. Likforman-Sulem L, Darbon J, Barney Smith EH (2011) Enhancement of historical printed document images by combining total variation regularization and Non-local Means filtering. *Image Vis Comput J* 29(5):351–363
44. Liu H, Wua Q, Zha H, Liu X (2008) Skew detection for complex document images using robust borderlines in both text and non-text regions. *Pattern Recognit Lett* 29:1893–1900
45. Lu Y, Tan CL (2003) A nearest-neighbor chain based approach to skew estimation in document images. *Pattern Recognit Lett* 24:2315–2323
46. Lu SJ, Chen BM, Ko CC (2006) A partition approach for the restoration of camera images of planar and curled document. *Image Vis Comput* 24(8):837–848

47. Makridis M, Nikolaou N, Papamarkos N (2010) An adaptive technique for global and local skew correction in color documents. *Expert Syst Appl* 37(10):6832–6843
48. Masalovitch A, Mestetskiy L (2007) Usage of continuous skeletal image representation for document images de-warping. In: International workshop on camera-based document analysis and recognition, Curitiba, pp 45–53
49. Moghaddam RF, Cheriet M (2010) A variational approach to degraded document enhancement. *IEEE Trans Pattern Anal Mach Intell* 32(8):1347–1361
50. Niblack W (1986) An introduction to digital image processing. Prentice-Hall, Englewood Cliffs, pp 115–116
51. Nomura S, Yamanaka K, Shiose T, Kawakami H, Katai O (2009) Morphological preprocessing method to thresholding degraded word images. *Pattern Recognit Lett* 30(8):729–744
52. Obafemi-Ajayi T, Agam G, Frieder O (2010) Historical document enhancement using LUT classification. *Int J Doc Anal Recognit* 13:1–17
53. O’Gorman L (1993) The document spectrum for page layout analysis. *IEEE Trans Pattern Anal Mach Intell* 15(11):1162–1173
54. Otsu N (1979) A threshold selection method from Gray-level histograms. *IEEE Trans Syst Man Cybern* 9(1):377–393
55. Pratt WK (2012) Digital image processing. Wiley, New York
56. Sauvola J, Pietikainen M (2000) Adaptive document image binarization. *Pattern Recognit* 33:225–236
57. Sezgin M, Sankur B (2004) Survey over image thresholding techniques and quantitative performance evaluation. *J Electron Imaging* 13(1):146–165
58. Shafait F, Breuel TM (2009) A simple and effective approach for border noise removal from document images. In: 13th IEEE international multi-topic conference, Islamabad, Dec 2009, pp 1–5
59. Shafait F, van Beusekom J, Keysers D, Breuel TM (2008) Document cleanup using page frame detection. *Int J Doc Anal Recognit* 11:81–96
60. Singh C, Bhatia N, Kaur A (2008) Hough transform based fast skew detection and accurate skew correction methods. *Pattern Recognit* 41(12):3528–3546
61. Solihin Y, Leedham C (1999) Integral ratio: a new class of global thresholding techniques for handwriting images. *IEEE Trans Pattern Anal Mach Intell* 21:761–768
62. Stamatopoulos N, Gatos B, Kesidis A (2007) Automatic borders detection of camera document images. In: 2nd international workshop on camera-based document analysis and recognition (CBDAR’07), Curitiba, Sept 2007, pp 71–78
63. Stamatopoulos N, Gatos B, Pratikakis I, Perantonis SJ (2011) Goal-oriented rectification of camera-based document images. *IEEE Trans Image Process* 20(4):910–920
64. Su B, Lu S, Tan CL (2011) Combination of document image binarization techniques. In: 11th international conference on document analysis and recognition, Beijing, 18–21 Sept 2011
65. Tonazzini A (2010) Color space transformations for analysis and enhancement of ancient degraded manuscripts. *Pattern Recognit Image Anal* 20(3):404–417
66. Trier ØD, Taxt T (1995) Evaluation of binarization methods for document images for document images. *IEEE Trans Pattern Anal Mach Intell* 17(3):312–315
67. Tsai CM, Lee HJ (2002) Binarization of color document images via luminance and saturation color features. *IEEE Trans Image Process* 11(4):434–451
68. Tseng YH, Lee HJ (2008) Document image binarization by two-stage block extraction and background intensity determination. *Pattern Anal Appl* 11:33–44
69. Ulges A, Lampert CH, Breuel TM (2005) Document image dewarping using robust estimation of curled text lines. In: 8th international conference on document analysis and recognition, Seoul, pp 1001–1005
70. van Beusekom J, Shafait F, Breuel TM (2010) Combined orientation and skew detection using geometric text-line modeling. *Int J Doc Anal Recognit* 13(2):79–92
71. Vinciarelli A, Luettin J (2001) A new normalization technique for cursive handwritten words. *Pattern Recognit Lett* 22(9):1043–1050

72. Vonikakis V, Andreadis I, Papamarkos N (2011) Robust document binarization with OFF center-surround cells. *Pattern Anal Appl* 14:219–234
73. Wahl FM, Wong KY, Casey RG (1982) Block segmentation and text extraction in mixed text/image documents. *Comput Graph Image Process* 20:375–390
74. Wang B, Li XF, Liu F, Hu FQ (2005) Color text image binarization based on binary texture analysis. *Pattern Recognit Lett* 26:1650–1657
75. Wu C, Agam G (2002) Document image De-warping for text/graphics recognition. In: Joint IAPR international workshop on structural, syntactic and statistical pattern recognition, Windsor, pp 348–357
76. Yan H (1993) Skew correction of document images using interline cross-correlation. *Graph Models Image Process* 55(6):538–543
77. Yang Y, Yan H (2000) An adaptive logical method for binarization of degraded document images. *Pattern Recognit* 33:787–807
78. Yin PY (2001) Skew detection and block classification of printed documents. *Image Vis Comput* 19(8):567–579
79. Zhang L, Tan CL (2005) Warped image restoration with applications to digital libraries. In: 8th international conference on document analysis and recognition, Seoul, pp 192–196
80. Zhang L, Yip AM, Brown MS, Tan CL (2009) A unified framework for document restoration using in painting and shape-from-shading. *Pattern Recognit J* 42(11):2961–2978

## Further Reading

- Gonzalez RC, Woods RE (2007) Digital image processing, 3rd edn  
Petrou M, Petrou C (2010) Image processing: the fundamentals, 2nd edn  
Pratt WK (2012) Digital Image Processing

---

## Part B

# Page Analysis

Historically, documents in printed form have been organized into pages in a way that facilitates sequential browsing, reading and/or search. The structure of these documents varies by genre; there is no doubt that there is an implicit organization that helps readers navigate the content in order to enable the transfer of information from the author to the reader. Scientific articles first provide basic metadata about the subject and authors, phone books provide indexing information and are ordered alphabetically, business correspondence follows established protocols, and tables organized implicitly in row and column structures provide an efficient representation of information.

As authors, humans have a canny ability to tweak the basic rules of organization to enhance the experience of the reader, and as readers we are able to use this information implicitly to comprehend what we are reading. While ideally we should be teaching our document analysis systems to be able to parse structures and use them to their advantage, traditionally page analysis has simply been a method to divide up content in a way that enables content analysis algorithms to perform more efficiently. Pages are divided into zones and these zones labeled and analysis by content-specific routines. Zones of text are divided into lines, lines of text into words, etc. As part of interpretation, the logical relationships between zones or regions on the page can be analyzed for reading order or as functional units of organization.

► **Part B** (Page Analysis) focuses on the fundamentals of this process and is divided into three chapters. In ►[Chap. 5](#) (Page Segmentation Techniques in Document Analysis), Koichi Kise addresses the first of these problems, page segmentation. Prior to the widespread deployment of word processing systems, textual documents tended to be fairly simple with text and illustrations falling into overlapping, homogeneous regions that could be segmented with basic image analysis techniques. However, as more complex layouts evolved and the possibility of processing less constrained handwritten material make this problem especially challenging. Robust solutions are essential and this chapter provides a firm baseline for understanding what is required.

► [Chapter 6](#) (Analysis of the Logical Layout of Documents) addresses the problem of analyzing the logical layout of documents. As previously suggested, this problem is often highly dependent on the type of document, so assumptions are made about genre. Andreas Dengel and Faisal Shaifait have a significant history with this problem and provide an in-depth discussion of the issues and solutions of logical analysis. They provide some important practical recommendations for those needing to design or incorporate their own logical analysis capabilities.

► [Chapter 7](#) (Page Similarity and Classification) takes a step back and examines the higher level of the classification of pages and the computation of a page similarity measure. Simone Marinai motivates this chapter as a step required to take input from a heterogeneous workflow and classify the document with respect to genre in order to allow subsequent genre-specific analysis. The related problem of determining similarity addresses both content and structural features. As previously suggested, layout information often provides important clues and user are more and more interested in grouping or searching document collections by document type. Providing robust similarity measures is an essential step toward this goal.

---

# Page Segmentation Techniques in Document Analysis

5

Koichi Kise

## Contents

Introduction.....	136
History and Importance.....	136
Evolution of the Problem.....	137
Applications.....	137
Main Difficulties.....	138
Summary of the State of the Art.....	138
Components of a Document.....	139
Classification of Pages and Their Layouts.....	139
Text-Block Level.....	140
Text-Line Level.....	142
Colors and Backgrounds.....	143
Other Factors.....	143
Classification of Methods.....	144
Object to Be Analyzed: Foreground or Background.....	144
Primitives of Analysis.....	145
Strategy of Analysis.....	147
Analysis of Pages with Nonoverlapping Layout.....	148
Projection-Based Methods.....	148
Smearing-Based Methods.....	151
Connected Component-Based Methods.....	154
Background Analysis Methods.....	158
Comparison of Methods.....	161
Analysis of Gray-Level and Color Pages.....	164
Analysis of Pages with Overlapping Layout.....	165
Analysis of Handwritten Pages.....	172
Conclusion.....	172
Cross-References.....	172
References.....	173
Further Reading.....	174

---

K. Kise

Department of Computer Science and Intelligent Systems, Graduate School of Engineering,  
Osaka Prefecture University, Sakai, Osaka, Japan  
e-mail: [kise@cs.osakafu-u.ac.jp](mailto:kise@cs.osakafu-u.ac.jp)

**Abstract**

In this chapter, we describe various notions and methods of page segmentation, which is to segment page images into homogeneous components such as text blocks, figures, and tables. It constitutes the whole process called layout analysis along with the classification of segmented components described in ►Chap. 7 (Page Similarity and Classification). This chapter starts with classification of page layout structures from various viewpoints including different levels of components and printing colors. Then we classify methods to handle each class of layout. This is done based on three viewpoints: (1) objects to be analyzed, foreground or background; (2) primitives of analysis, pixels, connected components, maximal empty rectangles, etc.; (3) strategy of analysis, top-down and bottom-up. The details of classified methods are described and compared with one another to know pros and cons of these methods.

**Keywords**

Background analysis • Bottom-up analysis • Connected component  
• Docstrum • Foreground analysis • Gabor filters • Layout analysis •  
Manhattan layout • Mathematical morphology • Maximal empty rectangle •  
Non-Manhattan layout • Page segmentation • Projection profile • Rectangular  
layout • Smearing • Top-down analysis • Voronoi diagram • Wavelet  
analysis • White tile

---

## Introduction

Page segmentation is a task of extracting homogeneous components from page images. As components, we often consider text blocks or zones, text-lines, graphics, tables, and pictures. The task of page segmentation does not include the classification of components, which is described in ►Chap. 7 (Page Similarity and Classification). It is important to understand that these tasks may not be separated; they are sometimes thought of as different sides of the same coin. Actually the task of both page segmentation and classification is often called “(physical) layout analysis.” However, some methods are designed to work without classification. In this chapter, we put the main focus on such methods and touch to methods with classification if needed.

The main target of page segmentation described in this chapter is machine-printed pages with a certain layout, since the majority of efforts have been devoted to their analysis. Other targets of page segmentation are handwritten pages, which are also described shortly at the last part of this chapter.

## History and Importance

The history of page segmentation is relatively long in the field of document analysis. After exploring the problem of character recognition, researchers started to address

the problem of character segmentation from a single text-line. This was around late 1970s and the beginning of 1980s. The need of page segmentation was recognized at about this time, since text-lines to be recognized are often laid out in pages, and thus it is necessary to extract text-lines from page images. In other words, in order to let character segmentation and recognition work, it is mandatory to apply layout analysis including page segmentation.

## **Evolution of the Problem**

At very early stages of page segmentation, it was described as a part of a whole system such as “document analysis systems” [1]. In such systems, layout of pages to be processed is not so complicated. However, as soon as researchers attempted to process a wider variety of pages, they realized that the problem of page segmentation was not easily solved. It was recognized that there were pages ranging from easy to hard. This led researchers to classify page layout. As described later, the most fundamental layout is “rectangular” which means that all components can be represented as nonoverlapping rectangles. The problem of page segmentation has been evolved to expand the class of layout to be analyzed. An important subclass of layout is called “Manhattan” which means that each component has boundaries consisting of line segments parallel or perpendicular with one another. In other words, regions of components are not necessarily convex. After much effort was made for addressing the problem of Manhattan layout, the focus of research was moved to process non-Manhattan layout and beyond.

Another problem evolution is about the primitive for page segmentation. The simplest one is connected components (CCs), under the assumption that there is no difficulty to separate the foreground from the background. However, this assumption does not hold for pages in which components are overlapped. In such a case, we need to take pixels as the primitive.

The last evolution is about the strategy of processing. Since layout of pages is hierarchical in such a way that text-lines consist of characters, one can have a strategy to extract larger components first and then divide each extracted component into smaller ones. To be precise, text blocks are first extracted, and then from each text block, text-lines are extracted and so on. This strategy is called splitting or top-down. Readers can easily imagine that we also have a reverse strategy, from the smallest to the largest, i.e., starting from the primitives, they are combined to form larger components. This is called merging or bottom-up.

## **Applications**

The most important application is to feed data to text-line extraction and character segmentation, since the task of page segmentation is designed to solve this problem. However, we also have several other applications, which are to use document images without recognition.

One important application is “reflow” of page contents. This is the process of generating a new image of a different size from its original. The purpose is to read the original document image on a smaller display available on a mobile phone. One idea to implement this functionality is to recognize the input document image to obtain the contents (symbol information) and display the contents on a smaller display by rendering them. However, this process is computationally expensive and not easy to be recovered from recognition errors. Another idea is not to recognize it but to apply “cut and paste” of the input document image to generate an image that fits into the new display. For this purpose, it is necessary to know the layout structure of the document image to apply correctly the cut and paste process.

Another important application is document image retrieval and clustering based on the layout of documents. In companies it is typical to store a large number of documents with the same layout but different contents, such as filled forms. It is also typical that in companies there are many documents with different layouts. Thus in order to deal with such documents efficiently, it is necessary to have a means to retrieve or to apply clustering based on the layout. The process of layout analysis is crucial to realize them.

## Main Difficulties

The main difficulty of the page segmentation lies in achieving the generality of processing without losing accuracy and efficiency. It is not so difficult to implement a method of page segmentation for documents with a specific layout. For example, if the layout is strictly fixed, perfect page segmentation is achieved by just cutting the image at the fixed positions. On the contrary, if the layout is totally free, it is necessary to understand the contents of the input document to obtain correct components. The task of page segmentation lies between these two extremes. Layouts with more constraints are easy to handle than those without them. Segmentation of pages with less constraints is harder and thus often more difficult to achieve high accuracy and efficiency.

Therefore it is reasonable to apply page segmentation methods that are capable of dealing with layouts to be analyzed. It may be wasteful to apply too general methods if the layout is with more constraints. For example, it may deteriorate the accuracy and efficiency of segmentation if one applies a page segmentation method for non-Manhattan layout to documents with rectangular layout. On the other hand, if the layout is beyond the one that can be handled by a method, the results of segmentation can be catastrophic. Thus another difficulty is to know the class of layout of documents to be processed.

## Summary of the State of the Art

Thanks to the effort that has been made, we can segment documents with constrained layouts such as rectangular and Manhattan. State-of-the-art technologies

of page segmentation are addressing the problem of segmentation of documents beyond Manhattan. As described later, the analysis of pages with layouts such as overlapping is still an open problem. Another important point is that the presence of noise makes the page segmentation difficult. This often occurs for analyzing old documents. Documents with less regularity such as handwriting are also hard to analyze.

To sum up, segmentation of pages with constrained layouts (up to Manhattan) and clean images has almost been solved. The analysis of pages beyond this, such as more general layout, noisy images, and/or presence of handwriting, is still under study.

In this chapter, we first define components of a document in the next section. Then we classify layouts and methods in sections “[Classification of Pages and Their Layouts](#)” and “[Classification of Methods](#).“ Next, we describe details of methods of each class. Finally, most difficult cases of overlapping layout and handwritten documents are described.

---

## Components of a Document

Let us start with the discussion on components of documents. They have a hierarchical structure as shown in Fig. 5.1. First, a document consists of pages each of which may include several components of a page. Typical components are text blocks, figures, tables, pictures, and ruled lines. A text block consists of text-lines, and a text-line is composed of characters. In this chapter, these components of a page are called “page components.”

The task of page segmentation is to extract these page components from input document images. In this chapter we consider mainly the extraction of page components larger than text-lines.

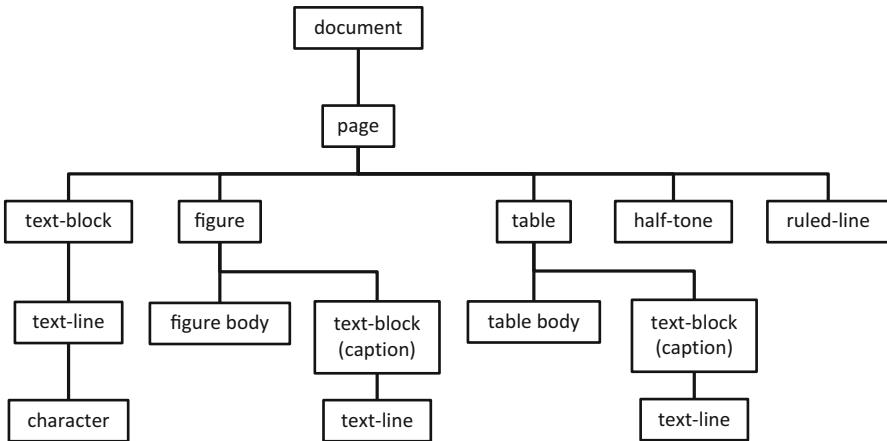
---

## Classification of Pages and Their Layouts

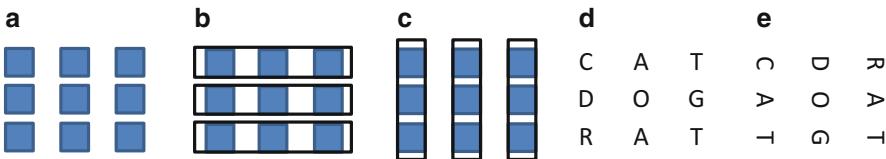
Next, let us consider the layout shown in Fig. 5.2 where filled squares indicate characters. For the squares in Fig. 5.2a, which of Fig. 5.2b or c do you think is the better interpretation of text-lines? As shown in Fig. 5.2b, c and d, e are both possible interpretations. This simple example indicates that the correct interpretation cannot be defined without the contents. Although this is an example of text-lines, the same holds for text blocks.

The above example suggests that, if we do not have an access to the contents, layout analysis, and thus page segmentation as its part, requires some assumptions on layout to obtain results of analysis. The purpose of this section is to describe such assumptions used especially in page segmentation.

Assumptions on layout can be classified into two levels: the text-block level and the text-line level. Each of them is described below.



**Fig. 5.1** Components of a document



**Fig. 5.2** Interpretation of layout

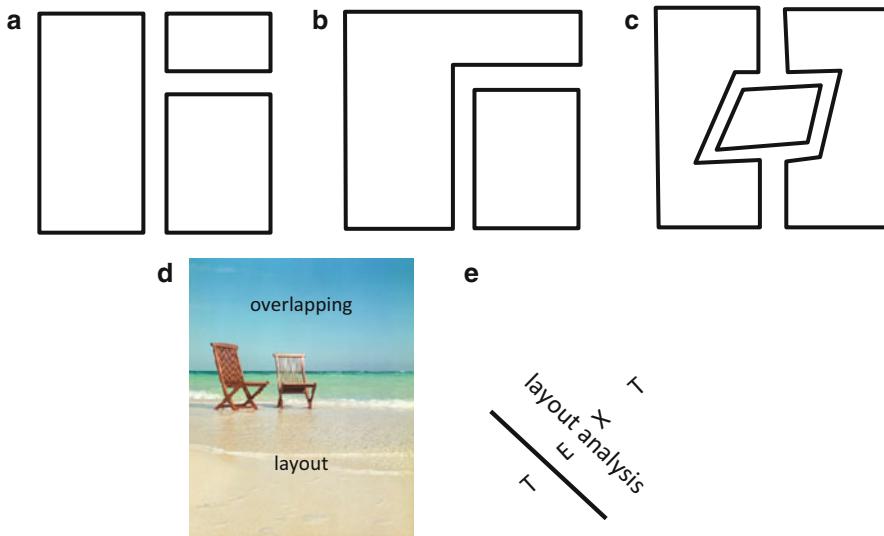
## Text-Block Level

### Rectangular Layout

The most fundamental layout is called “rectangular” which means all components are circumscribed by nonoverlapping rectangles whose sides are parallel or perpendicular to the border of a page. When pages are without skew, edges of rectangles are vertical or horizontal. Figure 5.3a shows examples of the rectangular layout. It is important to note that most of important publications such as business documents, scientific papers, and most of books are with this layout.

### Manhattan Layout

Some components cannot be represented by nonoverlapping rectangles because of their concave shape as shown in Fig. 5.3b. If regions of components are represented by sides parallel or perpendicular with one another as shown in this figure, the layout is called Manhattan. Note that by its definition Manhattan layout includes rectangular layout as its subclass. Documents with multiple columns such as newspapers and magazines are often with this class of layout.



**Fig. 5.3** Classes of layout. (a) Rectangular, (b) Manhattan, (c) non-Manhattan, (d) overlapping layout 1, and (e) overlapping layout 2

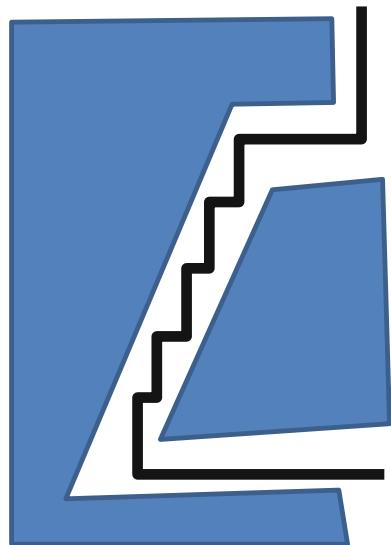
### Non-Manhattan Layout

There is a class of layout beyond Manhattan. For example, components with slant sides as shown in Fig. 5.3c are not circumscribed by parallel or perpendicular sides. As shown in this example, a class of layout beyond Manhattan but all components are with no overlapping regions is called non-Manhattan. Readers may think that slant borders as shown in Fig. 5.3c can also be represented by using many small parallel or perpendicular sides as in Fig. 5.4. In order to avoid such cases, it is necessary for Manhattan layout to have enough length of sides. As is the relation between rectangular and Manhattan, non-Manhattan layout includes Manhattan as its special case. It is typical to have this class of layout for magazines with larger figures and pictures.

### Overlapping Layout

All the layout classes introduced above are with components whose regions have no overlap. Although these classes cover most document publications, some documents such as graphical magazines are beyond these classes. For example, text can be laid out on a picture as shown in Fig. 5.3d. Another example is illustrated in Fig. 5.3e where text blocks intersect with each other. The former case is different from the latter. In the former case, there is no clear distinction between the foreground and the background; pixels of one component may be adjacent to those of others.

**Fig. 5.4** Separation of components of non-Manhattan layout with short edges



On the other hand, in the latter case, the background is clearly separated from the foreground, and pixels of components are adjacent only with those of the background.

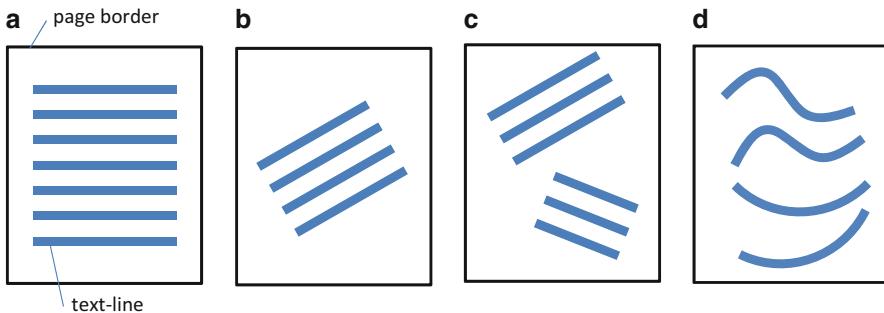
Although the former is quite common in modern publications such as advertisements, the latter is rare. An example can be found in maps, but they are out of the scope of this chapter. In the following, in contrast to the overlapping layout, the layouts, rectangular, Manhattan and non-Manhattan described above.

## Text-Line Level

Page layout can also be classified with respect to the layout of text-lines. Let us again begin with the most basic one and go to less restricted layouts. Figure 5.5 shows the layouts described below.

### Horizontal or Vertical Text-Lines

The most fundamental and thus common layout at the text-line level is that all text-lines are parallel to the horizontal side of the page as shown in Fig. 5.5a, where text-lines are represented as thick black lines. If the page image is without skew, the text-line is horizontal and thus we call this layout horizontal text-line layout. In some Asian languages, text-lines may be laid out vertically, which is also classified into this class. Most documents have this class of layout except for designed pages such as advertisements.



**Fig. 5.5** Layout of text-lines

### Parallel Text-Lines

Text-lines can be laid out parallel with one another but, as shown in Fig. 5.5b, may not be laid out horizontally. Another layout with more freedom is the layout with partially parallel text-lines as in Fig. 5.5c. These layouts are, however, not common and thus only available in advertisement documents.

### Arbitrary Text-Lines

With the highest freedom, text-line may have arbitrary shape such as illustrated in Fig. 5.5d. However, pages only with this layout are uncommon. It may be used with the horizontal text-line layout for their main part.

## Colors and Backgrounds

It is still major to have main text in black or dark color. In such cases, the background is often with white or light color. We call such documents black-and-white documents. The main text is also printed in black even in color-printed documents. Some parts are printed in different colors for emphasizing them. For advertisement documents, text may be printed totally in different colors with colored and/or textured background. A typical case would be text on a picture as shown in Fig. 5.3d. As a special case, a single text-line can be partly on a light background and the rest on a dark background. In this case the color of the text-line can be changed partly so as to make it distinguishable from the background.

## Other Factors

### Printed or Handwritten

Nowadays, printed documents are dominant in our daily life due to the extensive use of computers for their preparation. However, some documents are still in the mixture

of printed and handwritten or purely handwritten. The former case includes printed forms filled out with handwritten characters. The latter can be personal notes or historical documents. In this chapter, we mainly deal with printed documents, and at the end of the chapter, we touch the layout analysis of (partially) handwritten documents.

### Clean or Degraded Documents

Thus far, we implicitly assume that all documents are clean enough for the processing. However, this does not hold for certain documents such as historical documents. Pages are degraded and a lot of noise such as stain can be found. In addition to historical documents, modern documents can also suffer from a similar problem if they are captured in an uncontrolled environment.

We have discussed various types of pages with respect to layout. In the history of layout analysis, researchers started their research from the most basic one, i.e., clean documents with rectangular layout and horizontal text-lines, and then gradually remove some assumptions to make their algorithms more general.

---

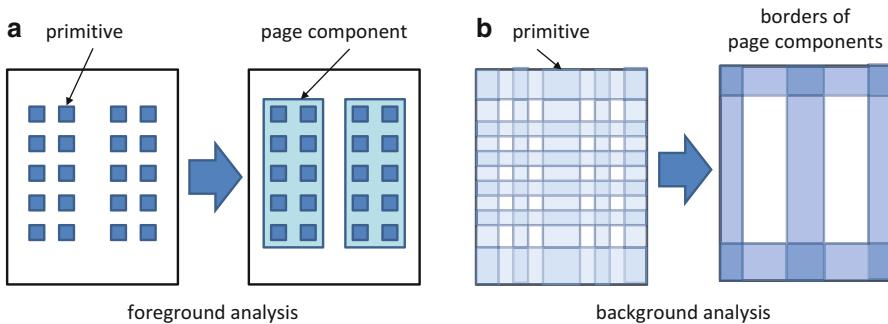
## Classification of Methods

Before going into the details of various methods of page segmentation, it would be helpful to have a bird's-eye view of these methods. There are several criteria for classifying methods, i.e., (1) an object to be analyzed, foreground or background; (2) a primitive of analysis, such as pixels and CCs; and (3) a strategy of analysis, i.e., top-down or bottom-up. In the following, we describe the details of these criteria focusing mainly on the analysis of black-and-white documents.

### Object to Be Analyzed: Foreground or Background

Suppose we analyze documents printed in black and white. It is typical that black parts represent the foreground such as characters and figures, while white parts correspond to the background. In this case, since there is a clear distinction between foreground and background, the analysis can be applied only to one of them. In this chapter the analysis of foreground and background is called foreground analysis and background analysis, respectively.

In both cases foreground and background are represented in terms of “primitives,” which will be described in the next subsection. Note that the primitive is an element of images that belongs exclusively to one of the page components or background. Thus there is no need to divide further the primitives in the process of analysis. As shown in Fig. 5.6a, the task of foreground analysis is, therefore, to group primitives to form page components. On the other hand background analysis is to group primitives to obtain borders of page components as shown in Fig. 5.6b.



**Fig. 5.6** (a) Foreground and (b) background analysis

If a document to be analyzed is not printed in black and white, the distinction of foreground and background is not trivial; the background of a page component can be the foreground of another page component, when documents have the overlapping layout. Even if a document is with a simple color print, foreground/background separation is not a trivial task; the same color can be used as both foreground and background. Thus the process of separation itself is a part of the goal of page segmentation.

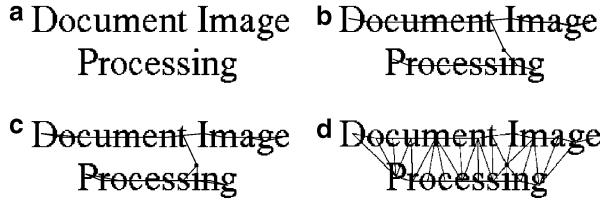
## Primitives of Analysis

The next criterion is about the primitive of analysis. Needless to say but the most fundamental and universal primitive is “pixel,” which can be used in both foreground and background analysis. However, documents with some classes of layout allow us to use larger primitives that reduce the burden of analysis thanks to their smaller number. In the field of computer vision, such primitives are generally called “superpixels.” For the task of page segmentation, superpixels can be of a variety of shapes depending on a class of layout.

For the case of black-and-white pages, an important primitive that can be used for nonoverlapping layout is “connected components” (CCs). As the connectivity for definition of CCs, it is common to employ 8-connectivity for black pixels. In this case, in order to avoid the contradiction of crossing connections, 4-connectivity is used for white pixels.

The burden of the processing can further be reduced by merging CCs. A natural way is to merge CCs if the distance between them is small enough. The question here is how to define the distance between CCs. There have been several methods for this purpose. The simplest is to measure the horizontal and vertical distance assuming that the skew is corrected [1]. A more general way is to use the morphological operation of “closing” that fills the gaps between black pixels. The distance can be defined as the Euclidean distance between centroids of CCs as well as the closest Euclidean distance between pixels of CCs.

**Fig. 5.7** Representations of adjacent relation among CCs.  
**(a)** Original image, **(b)** MST,  
**(c)**  $k$ -NN ( $k = 2$ ), and **(d)**  
 Delaunay triangulation

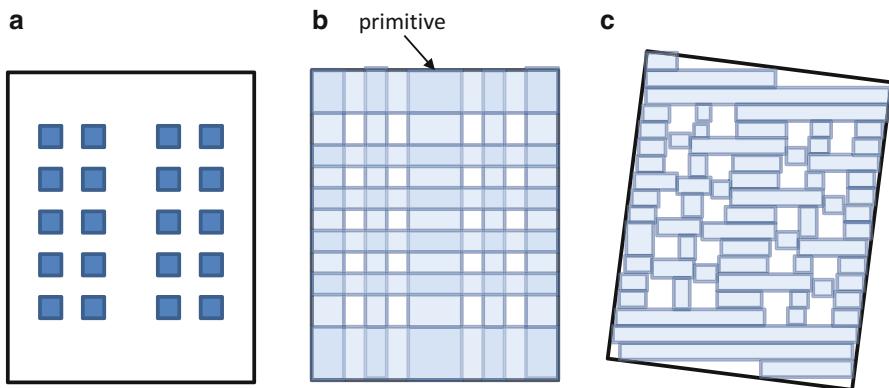


The above methods of merging CCs are done by just looking locally at the distance between CCs. However, it may depend on other CCs whether or not the CCs of interest should be merged. In order to evaluate such a “context” defined in relation to other CCs, we need a representation of relationship among CCs or a structure among CCs. For this purpose, there have been several methods proposed so far.

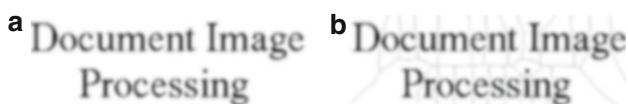
In general, a structure can well be represented by using a graph where nodes are CCs and edges are adjacent relations between them. The problem here is how to define the edges. Several ways have been proposed as shown in Fig. 5.7. Figure 5.7b is the minimum spanning tree (MST) that connects all CCs by minimizing the sum of distances of edges [2]. Figure 5.7c is  $k$ -nearest neighbors ( $k$ -NN) from each CC [3]. The last one, Fig. 5.7d is the Delaunay triangulation obtained from the CCs [4]. The details of these representation including pros and cons will be described later. One thing that needs to understand now is that once the page is represented as a graph of CCs, the task of page segmentation is to select necessary edges from the graph or equivalently to delete unnecessary edges from the graph.

The situation is different if we consider background analysis. It is typical that the background is a big single CC so that we need to utilize smaller primitives to form borders of page components. We have several ways to realize this as shown in Fig. 5.8. The most fundamental primitive is “maximal empty rectangles” [5] shown in Fig. 5.8b. A rectangle is empty if it only contains white pixels. An empty rectangle is maximal if it is unable to enlarge its area while keeping it empty. In other words, all sides of an empty rectangle touch to either the border of a page or black pixels. Note that the maximal empty rectangles have sides parallel or perpendicular to the side of a page, assuming that pages have been deskewed. Figure 5.8c shows a way to deal with skewed pages. This representation is called white tiles [6]. The white tiles are also empty rectangles but not maximal. They have variable heights with the maximal width. If the height is small enough, it can represent white space even for skewed pages as shown in Fig. 5.8c.

Similarly to the primitives of foreground, we can also think about the structure among background primitives. An important property is that the representation should keep the topology of background. The area Voronoi diagram [7] shown in Fig. 5.9b realizes a representation with this property. Edges between connected components represent the equidistant positions between borders of connected components. Because the distance does not change by rotation, another important property of the area Voronoi diagram is that it is rotation invariant. This means that



**Fig. 5.8** Primitives for representing the background. (a) Foreground, (b) maximal empty rectangles, and (c) white tiles



**Fig. 5.9** Structural representations of background. (a) Original image and (b) area Voronoi diagram

even if an input image is rotated, the same representation can be obtained. Once the structural representation of background is constructed, the task here is to select from the graph appropriate edges that represent borders of page components.

For gray-level and color document images, separation of the foreground from the background is not trivial. Thus the primitives for the processing is first to solve this problem. Although the approach of superpixels may also be applied for this purpose, only few attempts have already been made. The most common way is to convert the analysis of gray-level and color documents into that of black-and-white documents. It can be done by applying color clustering; after the clustering, the image of each cluster can be viewed as a black-and-white image. If it is not possible to obtain reasonable results by the clustering, the appropriate primitive is “pixel.” For the analysis of overlapping layout, for example, it is normal to analyze images using this primitive.

### Strategy of Analysis

The final criterion is about the strategy of analysis. As shown in Fig. 5.1, page layout has a hierarchical structure. The strategy can be divided into two, with respect to from which side (from either the root or the leaves) the processing starts.

The strategy from the root, i.e., the whole page, is called “top-down,” while that from the leaves is called “bottom-up.” The top-down analysis is to split the whole page into the second level (text blocks, figures, tables) and continues this splitting until having the leaves such as characters or text-lines. On the other hand, the bottom-up analysis is to merge the primitives into the leaves and then continue to merge until obtaining the page components at the highest level. In the literature some methods take an intermediate strategy, starting from the middle (e.g., words); the analysis is to obtain larger components such as text blocks and smaller components such as characters.

The use of the terms top-down and bottom-up is similar to that of parsing in the field of language analysis. As is needed in the language parsing, those strategies generally require “backtracking” if a tentative result of analysis is found incorrect. For example, in the top-down analysis, the analysis may start by assuming that the page layout is two-column. Then after obtaining columns, it may be found that the result is incorrect due, for example, to different column widths. In order to pursue other possibilities of analysis such as three-column layout, the result of two-column should be cancelled by backtracking. In practice, however, the backtracking is rarely applied, simply because it is too time-consuming. Algorithms of page segmentation tend to be designed to work without backtracking. This is similar to the situation of parsing algorithms for programming languages. Note that although the parsing algorithms have a guarantee that it works properly without backtracking, there is no guarantee for the case of page segmentation.

---

## Analysis of Pages with Nonoverlapping Layout

In this section, we describe the details of page segmentation algorithms based upon the criteria introduced in the previous section. We first focus on methods for analyzing pages with nonoverlapping layout. We start from foreground analysis methods, followed by background analysis methods.

Table 5.1 lists representative methods for nonoverlapping layout. As we have already seen, methods are classified into two categories: foreground analysis and background analysis. The former is further divided into three subcategories: projection-based, smearing-based, and connected components-based methods. We describe each method in detail in this section.

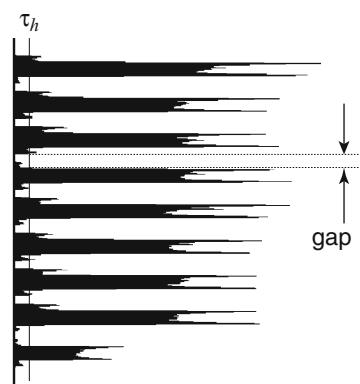
### Projection-Based Methods

We first introduce projection-based methods which utilize projection profiles of a page image. Figure 5.10 shows the horizontal projection profile of a page. If there is no skew of a page and all text-lines are parallel to the horizontal sides of a page, we can observe the periodical gaps as shown in this figure. The length of gaps is a clue to find a border between page components. Projection profile methods employ this clue to analyze pages. To be precise, gaps are identified as the ranges below the threshold  $\tau_h$  of the projection profile as shown in Fig. 5.10. We can split blocks vertically

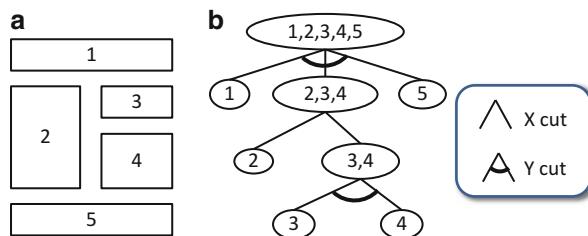
**Table 5.1** Representative methods for nonoverlapping layout

Method	Analysis	Primitive/representation	Strategy	Assumption
1. Projection-based methods				
Recursive XY cut (RXYC) [8]	Foreground	Projection profile	Top-down	No skew
Syntactic segmentation [9]	Foreground	Projection profile	Top-down	No skew
White streams [10]	Foreground	Projection profile	Top-down	
Hough transform [11, 12]	Foreground	Hough domain representation	Bottom-up	
2. Smearing-based methods				
Run-length smearing algorithm (RLSA) [1]	Foreground	Smeared pattern	Top-down	No skew
Morphology [13, 14]	Foreground	Pixel	Bottom-up	
3. Connected components-based methods				
Minimum spanning tree [2, 15, 16]	Foreground	Minimum spanning tree	Bottom-up	
Docstrum [3]	Foreground	<i>k</i> -NN	Bottom-up	
Delaunay triangulation [4]	Foreground	Delaunay triangulation	Bottom-up	
4. Background analysis methods				
Shape-directed covers [5, 17]	Background	Maximal empty rectangles	Top-down	No skew
White tiles [6]	Background	White tiles	Top-down	
Voronoi diagram [7, 18]	Background/foreground	Voronoi edges	Top-down	

processor simulator and a detailed memory simulator for the Dash prototype. Tango allows a parallel application to run on a uniprocessor and generates a parallel memory-reference stream. The detailed memory simulator is tightly coupled with Tango and provides feedback on the latency of individual memory operations.

**Fig. 5.10** Projection profiles

**Fig. 5.11** Recursive XY cut.  
 (a) Blocks and (b) XY tree



by finding wider gaps of horizontal projection profiles. This is called vertical cuts. Likewise, horizontal cuts can be found in vertical projection profiles.

A representative method of this category is called Recursive XY Cut (RXYC) [8]. In order to understand the algorithm, we first need to know the layout structure to be analyzed by this method. Figure 5.11 shows such a layout structure. Nodes of the tree in Fig. 5.11b represent regions. For example, (2,3,4) indicates the region consisting of blocks 2, 3, and 4. Each edge represents either  $X$  cuts or  $Y$  cuts. Edges of the tree also indicate a part of relation between nodes: regions of children of a node are included in the region of the node.  $X$  and  $Y$  cuts are applied alternately from top to bottom of the tree. Thus the processing is top-down. The leaves of the tree correspond to the smallest page components. The algorithm of RXYC works to construct the tree by analyzing projection profiles.

The above method employs a general assumption on layouts such as “inter-text-line spacing is narrower than inter-text-block spacing.” However, this is not the only way of top-down page segmentation. We can apply the top-down page segmentation which assumes a specific layout structure if we have already known it. The knowledge on a specific layout can be represented as a set of production rules [9]. This method analyzes a projection profile by rewriting it to extract page components. Such methods that employ document-specific knowledge of layout are sometimes called “model-based methods.” Note that in general we do not use a model-based method just only for page segmentation; it is used in combination with or in the process of logical layout analysis described in ►[Chap. 6](#) (Analysis of the Logical Layout of Documents).

Let us go back to methods without knowledge about specific pages. Many methods based on projection profiles assume that input page images are deskewed. In the case that the skew correction is incomplete, they generally fail to segment pages. This problem can be solved by applying projection locally [10]. In this method a page image is split into horizontal belts (blocks of scan lines) with a predetermined height. Then for each belt the projection profile is calculated for finding column gaps. Note that if the height of the belts is small enough, the influence of skew is limited so that column gaps can be found.

Another possible method for coping with skew is to calculate projection profiles by rotating the axis onto which black pixels are projected. It is known that this process is equivalent to the Hough transform [11]. For example, the Hough transform is applied to the centroids of CCs to extract text-lines from line

drawings [12]. However, most of the methods using the Hough transform are not for page segmentation but for skew correction, which are described in ►Chap. 4 (Imaging Techniques in Document Analysis Processes).

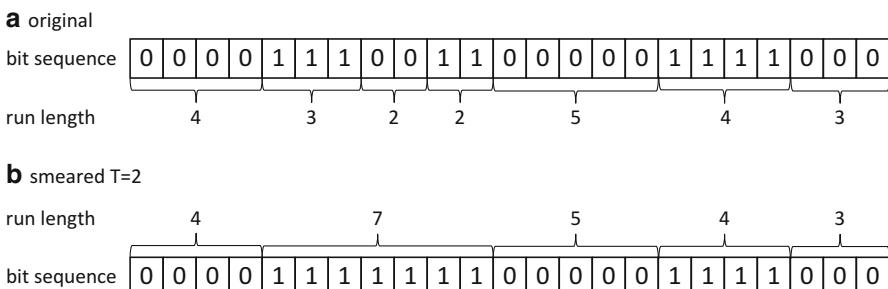
## Smeearing-Based Methods

Projection-based methods try to find white space (gaps) between page components. The totally reverse idea is to fill the space within each component to extract it. Smearing-based methods are methods based on this idea.

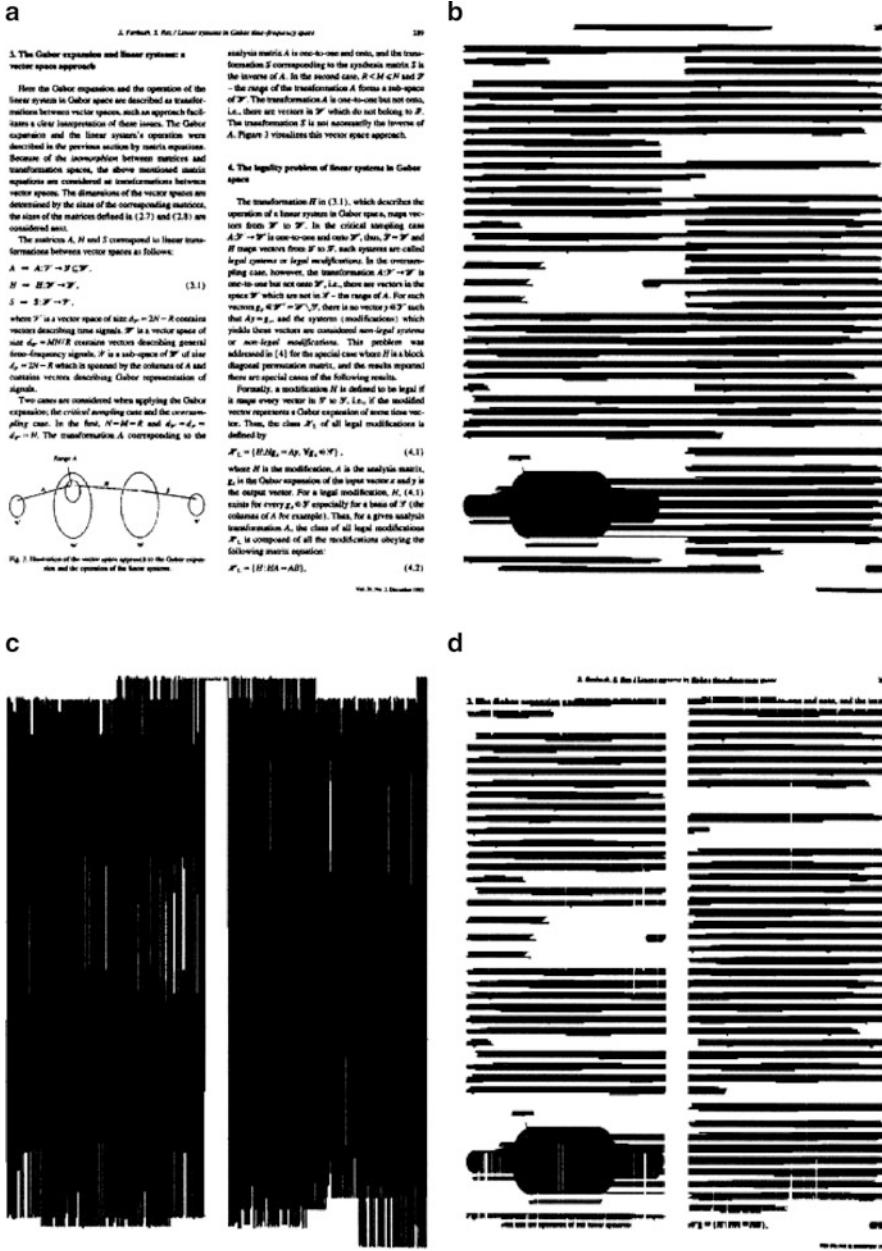
### Run-Length Smearing Algorithm (RLSA)

The earliest proposal was called run-length smearing algorithm (RLSA), which is sometimes called run-length smoothing algorithm. Suppose we deal with binary documents where 0s and 1s represent white and black pixels, respectively. The run-length means the number of continuous 0s or 1s along with the scan line. For example, 0000 and 111 are converted into 4 and 3, respectively. The run-length encoding of a binary image is to convert each scan line of the input image to its run-length representation. For example, a bit sequence shown in Fig. 5.12a is transformed into the one in Fig. 5.12b with the minimum length  $T = 2$  of white runs (white runs less than or equal to 2 are filled). Note that such a bit operation is extremely efficient, which is an advantage of this method.

In RLSA, the above run-length smearing is applied both horizontally and vertically. Figure 5.13 shows an example. The original image shown in Fig. 5.13a is transformed into horizontally smeared (b) and vertically smeared (c) images. Note that the appropriate threshold value  $T$  is different for horizontal and vertical smearing. In Fig. 5.13c, columns are correctly extracted. In Fig. 5.13b, text-lines and figures in different columns are merged. RLSA takes the logical AND of these horizontally and vertically merged images to generate Fig. 5.13d, where most text-lines and figures in each column have been successfully extracted. Similar to the case of the RXYC, RLSA also assumes that the skew has been corrected.



**Fig. 5.12** Run-length smearing



**Fig. 5.13** Run-length smearing algorithm (RLSA) [1] (Reprinted from Fig. 5.4 in Yin [19])

## Morphology-Based Method

The RLSA algorithm described above can be thought of as a special case of more general processing which we call here “morphology-based methods.” The fundamental operations of morphology-based methods are dilation and erosion from mathematical morphology:

$$\text{Dilation: } A \oplus B = \bigcup_{b \in B} A_b$$

$$\text{Erosion: } A \ominus B = \bigcap_{b \in B} A_{-b}$$

where  $A$  is an input image,  $B$  is a structural element, and  $A_b$  is the translation of  $A$  along the pixel vector  $b$ . As the names indicate, dilation and erosion are to dilate and erode the foreground of original image  $A$  by the amount the structural element  $B$  defines, respectively. Based on these two operations, opening and closing are defined as follows:

$$\text{Opening: } A \circ B = (A \ominus B) \oplus A$$

$$\text{Closing: } A \cdot B = (A \oplus B) \ominus A$$

Opening allows us to remove objects which are small with respect to the structural element, while closing fills gaps which are small with respect to it. If the structural element is isotropic, images with skew can also be handled. With the knowledge about layout structure such as the direction of text-lines, an anisotropic structural element allows us to find text-lines and blocks more effectively. The closing operation with the structural element  $1 \times T$  represents the horizontal smearing shown in Fig. 5.12.

In order to make the segmentation process efficient and reliable, a series of morphological operations is applied. Bloomberg has proposed multiresolution morphology to separate picture regions from text [13]. Its fundamental processing is as follows. Since the distance between pixels in halftones is smaller than that between characters, one can apply closing for filling the gaps in halftone pixels. This results in large blobs that represent parts of halftones. Then the opening is applied to erase characters. The remaining is the large blobs that become “seeds” to find pictures. This process requires the application of relatively large structural elements, which lowers the efficiency of processing. Actually, one of the major drawbacks of morphology is its computational burden. This problem has been solved by using multiresolution morphology called “threshold reduction.” In this method,  $2 \times 2$  pixels are transformed into one pixel by using a threshold for the sum of  $2 \times 2$  pixel values. This allows us to mimic opening and closing depending on the threshold. Applying the threshold reduction several times with different thresholds, it is possible to achieve a similar effect with a much lower processing cost.

## Connected Component-Based Methods

In this section, we introduce the analysis based on CCs. There are many methods of connected component analysis which can be characterized by the representation of structure among the CCs as we have already seen in section “[Primitives of Analysis](#).”

### Minimum Spanning Tree (MST)

The first method we describe here is based on the minimum spanning tree (MST). We define the distance between CCs as the Euclidean distance between centroids of CCs. Consider here a graph whose nodes are CCs and whose edges are between centroids and thus associated with the distance. Then the minimum spanning tree is defined as the tree structure that connects all nodes with the minimum sum of distance of edges [2].

The construction of MST is quite easy; greedy methods allow us to obtain the MST. Here we describe Kruskal’s. First, find the edge whose distance is the shortest among all possible edges between CCs and delete it from the set of all possible edges. Then select the next edge whose distance is the shortest among the remaining edges, on condition that the inclusion of the selected edge still keeps the tree structure. Otherwise, the edge is thrown and the next one is selected. When there is no more remaining edge, the resultant tree represents the MST.

Once the MST is constructed, it can be used to extract page components as subtrees of the MST [15]. This is based on the following assumption about the distance between connected components. In general, the distance between CCs in the same page component is smaller than that between CCs in different page components. If this assumption holds, the process of page segmentation is to select edges that lie between different components and delete them. Dias employs the distance associated with edges as well as information on CCs for the selection [16]. The threshold is calculated based on the distribution of distance.

### Docstrum

The problem of MST-based page segmentation is that the assumption does not always hold. In other words, some edges lying between different page components have smaller distance than those within the same page component. A way to solve this problem is to select edges for deletion not only based on the distance but also other features. Another possible way is to change the representation.

The document spectrum or *docstrum* proposed by O’Gorman [3] is a method that overcomes the above difficulty. The idea is simple; the problem can be rephrased as the lack of edges within page components. In order to have more chances not to lose such “within” component edges, edges for the representation are increased. For this purpose,  $k$ -nearest neighbor ( $k$ -NN) graph is employed in the docstrum, where nodes again represent CCs, and edges indicate  $k$ -NN from each CC. An example is shown in Fig. 5.14.

**Fig. 5.14** 5-NN of connected components



Taking  $k$ -NN of CCs is simple but powerful to obtain statistical information of the page. For each  $k$ -NN pair  $(ij)$  of CCs, we can observe  $(d, \phi)$  where  $d$  is the distance and  $\phi$  is the angle of each edge. The 2D plot of  $(d, \phi)$  is called docstrum. The value of  $k$  is typically set to 5. From upright page images, it is typical to have clear peaks at 0 and  $\pm 90^\circ$  which correspond to angles of edges in text-line and between text-lines, respectively. This shows that based on these statistics, the skew angle, within-line and between-line spacing, is estimated as follows: First, the highest peak of angle distribution indicates the skew angle. The within-line and between-line spacing are estimated from the distance distribution of edges close and perpendicular to the estimated skew angle, respectively.

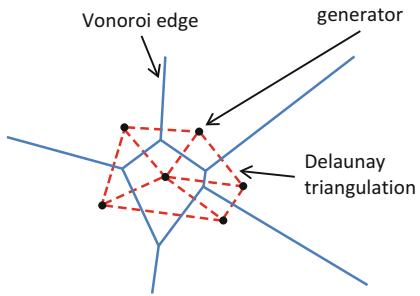
Then the above estimated parameters are employed for page segmentation. The processing is fully bottom-up. First, in order to extract text-lines, connected components are merged by using the estimated skew angle and within-line spacing. Next, text blocks are extracted by merging text-lines again by taking into consideration parameters such as angles of text-lines and distance between text-lines. Note also that there are several clear peaks in the distance distribution. The peak with the smallest distance shows the inter-character distance, and the peak with the largest distance indicates the inter-text-line distance. By using these values, it is possible to automatically adjust the parameters for merging CCs to obtain page components.

The above estimation of parameters is based upon the assumption that the values of these parameters are common in the page. In other words, the parameters are globally estimated. In the case that there are several page components with different values of parameters, the additional processing is necessary. The simplest one is to apply clustering in the parameter space so that each cluster represents possible values.

### Delaunay Triangulation

Although the docstrum is powerful and effective, a problem still remains in setting the parameter  $k$  since the appropriate value of  $k$  depends on the layout. If it is larger than the appropriate value, the  $k$ -NN graph includes edges spurious to estimate within-line and between-line spacing. For example, a larger  $k$  produces edges that connect not only neighboring CCs but also further CCs. For example, in the word “example” an edge from “x” to “m” can also be set in addition to “a.” On the other hand, the value of  $k$  smaller than the appropriate value prevents us from having enough edges between text-lines, which makes the estimation of between-line spacing unreliable. In order to solve this problem, we need a representation

**Fig. 5.15** Point Voronoi diagram and its corresponding Delaunay triangulation



that limits edges to the ones between the “neighboring” connected components. Note that the MST cannot give us a solution, since it tends to miss edges on the neighboring CCs.

This problem can be solved by the diagram called Delaunay triangulation. First, let us introduce some definitions by using Fig. 5.15 as an example. Let  $P = \{p_1, \dots, p_n\}$  be a set of points or *generators* and  $d(p, q)$  be the distance between points  $p$  and  $q$ . A Voronoi region  $V(p_i)$  of a point  $p_i$  is defined as

$$V(p_i) = \{p | d(p, p_i) \leq d(p, p_j), \forall j \neq i\}.$$

The Voronoi diagram  $V(P)$  generated from the point set  $P$  is defined as a set of Voronoi regions:

$$V(P) = \{V(p_1), \dots, V(p_n)\}.$$

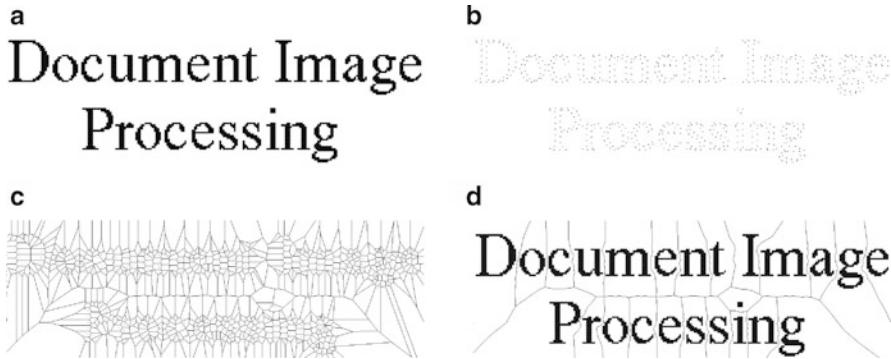
The boundaries of Voronoi regions are called Voronoi edges.

The dual graph of the Voronoi diagram is called the Delaunay triangulation. An example is also shown in Fig. 5.15 using dashed lines. Nodes of the Delaunay triangulation are generators, and edges exist between pairs of generators whose Voronoi regions share a Voronoi edge.

We can construct the Voronoi diagram and the Delaunay triangulation by taking centroids of CCs as generators. However, if the shape of CCs is far from the point and the size of CCs are not uniform, it is better to take their shape into account when we construct the Voronoi diagram and the Delaunay triangulation.

First, let us consider the case of the Voronoi diagram. The Voronoi diagram generated from figures of arbitrary shape is called the area Voronoi diagram. In order to distinguish it from the previously defined Voronoi diagram, we call the previous one the point Voronoi diagram. The definition of the area Voronoi diagram is as follows. Let  $G = \{g_1, \dots, g_n\}$  be a set of nonoverlapping figures (in our case CCs) and let  $d(p, g_i)$  be the distance between a point and a figure defined as

$$d(p, g_i) = \min_{q \in g_i} d(p, q).$$



**Fig. 5.16** Approximate construction of the area Voronoi diagram [7]. (a) Original image, (b) sampled points, (c) point Voronoi diagram, and (d) area Voronoi diagram

The Voronoi region  $V(g_i)$  and the Voronoi diagram  $V(G)$  are defined as

$$\begin{aligned} V(g_i) &= \{p | d(p, g_i) \leq d(p, g_j), \forall j \neq i\}, \\ V(G) &= \{V(g_1), \dots, V(g_n)\}. \end{aligned}$$

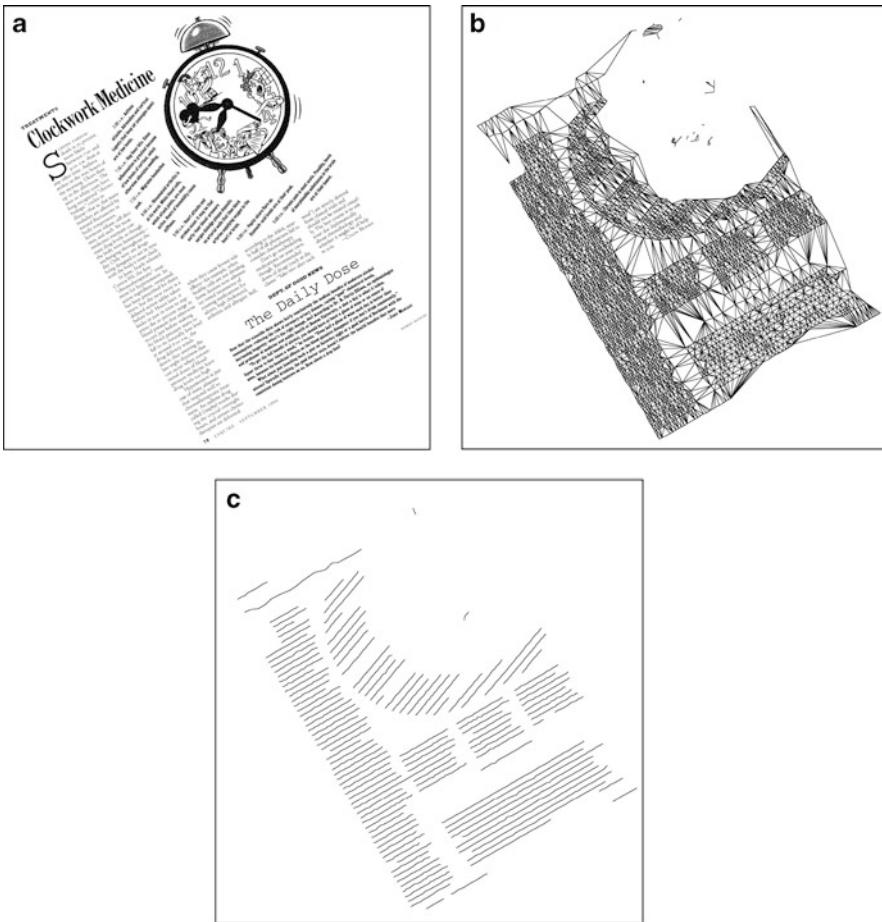
The area Voronoi diagram can easily be constructed in an approximate manner by sampling points on borders of CCs as generators and delete Voronoi edges lying on the same CC. Figure 5.16 shows an example. From the points (Fig. 5.16b) of CCs (Fig. 5.16a), the point Voronoi diagram (Fig. 5.16c) is generated. By deleting edges lying between the same connected components, the area Voronoi diagram (Fig. 5.16d) is obtained.

As the dual graph of the area Voronoi diagram, we can also define the Delaunay triangulation that connects figures of arbitrary shape sharing the Voronoi edge as shown in Fig. 5.7d. In this Delaunay triangulation, the distance associated with edges is defined as

$$d(g_i, g_j) = \min_{p_i \in g_i, p_j \in g_j} d(p_i, p_j).$$

In the approximate construction,  $p_i$  and  $p_j$  are sampling points of  $g_i$  and  $g_j$ , respectively.

Based on the Delaunay triangulation of a page, the task of page segmentation is to delete edges that connect different page components. In [4], a bottom-up method for the selection has been proposed. In this method, the area ratio of CCs connected by an edge as well as the angle of an edge is employed for the selection. Starting from the Delaunay triangulation shown in Fig. 5.17b, the method first selects seeds, which are reliable parts of text-lines, and then extends them to form full text-lines as in Fig. 5.17c. By merging text-lines taking into account their orientation, length, and distances, we can obtain text blocks.



**Fig. 5.17** Delaunay triangulation and text-lines as its subgraphs. (a) Original image, (b) Delaunay triangulation, and (c) text-lines

## Background Analysis Methods

We have seen methods that work directly on the foreground to extract page components by merging primitives. As described earlier, there exist different methods that deal with the background to segment the whole page into page components. In this section, some representative methods of this category are described.

### Shape-Directed Covers

The method “shape-directed covers” proposed by Baird [5] is the first well-known method of this category. Figure 5.18b shows blank regions around page components. As you can see, these regions can be represented by white rectangles each of which



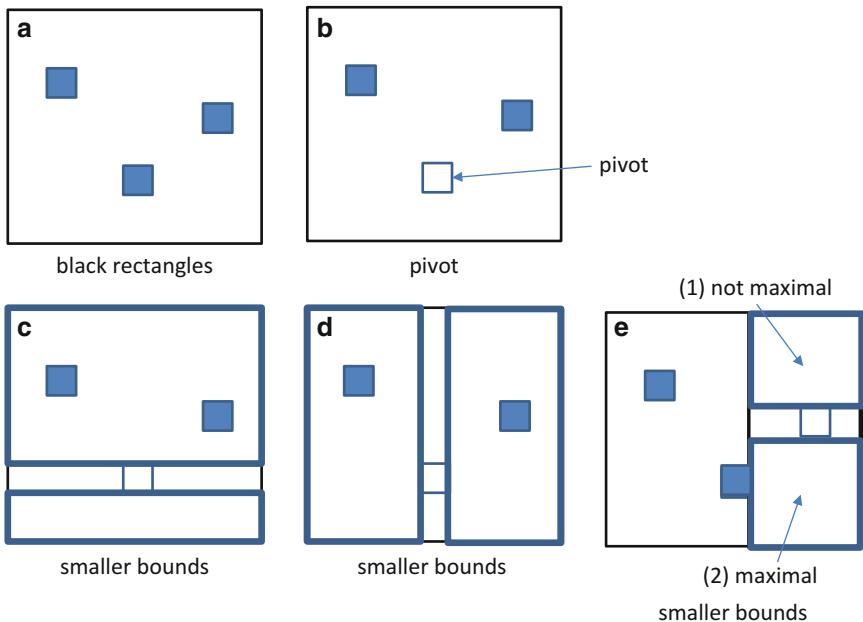
**Fig. 5.18** Examples of (a) black and (b) white rectangles (Reprinted from Fig. 5.4 in “Page Segmentation Based on Effective Area of Background,” Trans. IEE of Japan, Vol. 116, No. 9, p. 1038, 1996)

maximally covers a part of the blank regions. The first question here is how to extract such white rectangles in an efficient way.

Although there are several methods proposed for this purpose, it has been pointed out that they are not always easy to implement. In this section we describe an algorithm inspired by Breuel [17] which is to extract “maximum”(largest) white rectangle based on the branch and bound.

The input to the algorithm is a set of black rectangles  $B = \{b_1, \dots, b_n\}$  in the page bounding box  $w_p$ . As black rectangles  $b_i$ , we often utilize bounding boxes of CCs. Examples are shown in Fig. 5.18a. The task here is to find a set of empty (white) rectangles  $W = \{w_1, \dots, w_m\}$ , under the conditions that (1)  $w_i \subset w_p$  where  $\subset$  means the inclusion of rectangles, (2)  $\forall i, j w_i \cap b_j = \emptyset$  where  $\cap$  indicates their intersection, and (3)  $\forall i w_i$  is maximal, i.e., all sides of the white rectangle touch either on the side of  $w_p$  or that of a black rectangle  $b_i$ .

The key idea of the algorithm is shown in Fig. 5.19 where the outer rectangle in Fig. 5.19a is a bound and smaller rectangles in the bound are black rectangles.



**Fig. 5.19** A branch-and-bound algorithm for maximal empty rectangles. (a) Black rectangles, (b) pivot, (c) smaller bounds, (d) smaller bounds, and (e) smaller bounds

At the beginning, the bound is set to the page bounding box. The algorithm works as follows. If the bound is not empty, i.e., including black rectangles, the bound is divided. For each black rectangle, take it as a pivot as shown in Fig. 5.19b and generate four smaller bounds as in Fig. 5.18c, d. Then for each bound, select black rectangles that overlap with the bound and call recursively the procedure. On the other hand, if the bound is empty, it may be a maximal white rectangle. The maximality can be confirmed by checking whether all sides of the bound touch to either a black rectangle or a side of the page bounding box. The bound (1) in Fig. 5.19e is not maximal since the left side touches nothing. The bound (2), on the other hand, is a maximal white rectangle.

Once all maximal white rectangles are extracted, page segmentation can be done by selecting white rectangles appropriate as borders of page components. Baird et al. have proposed to use the shape score  $s = a \times r$  where  $a$  is the area of a white rectangle and  $r$  is a truncated aspect ratio (the ratio of the longer to the shorter side length, but if it is larger than a threshold, say 16, it is set to 0). White rectangles are sorted by the score and select as borders from top to a certain limit.

### White Tiles

The white rectangles are effective under the following assumptions: (1) the page is precisely deskewed and (2) the layout is Manhattan. These assumptions are removed by the method called the white tiles proposed by Antonacopoulos [6]. A white tile

is a vertically concatenated white runs with the following condition. Let  $(x_{si}, x_{ei})$  be the  $x$  coordinates of the start and the end of the  $i$ th white run. The white tile is a concatenated white runs from  $j$ th to  $k$ th ( $j \leq k$ ) on condition that

$$\sum_{i=j}^{k-1} (|x_{si+1} - x_{si} + 1| + |x_{ei+1} - x_{ei} + 1|) < T,$$

where  $T$  is a threshold. An example of white tiles is shown in Fig. 5.20c, which is constructed from the smeared image (Fig. 5.20b) of the original page image (Fig. 5.20a). Based on the extracted white tiles, the task of page segmentation is to connect them to form the borders of page components. In [6] contours of white tiles are traced to circumscribe a region that is recognized as a page component. An example of traced contours is shown in Fig. 5.20d.

### Voronoi Diagram

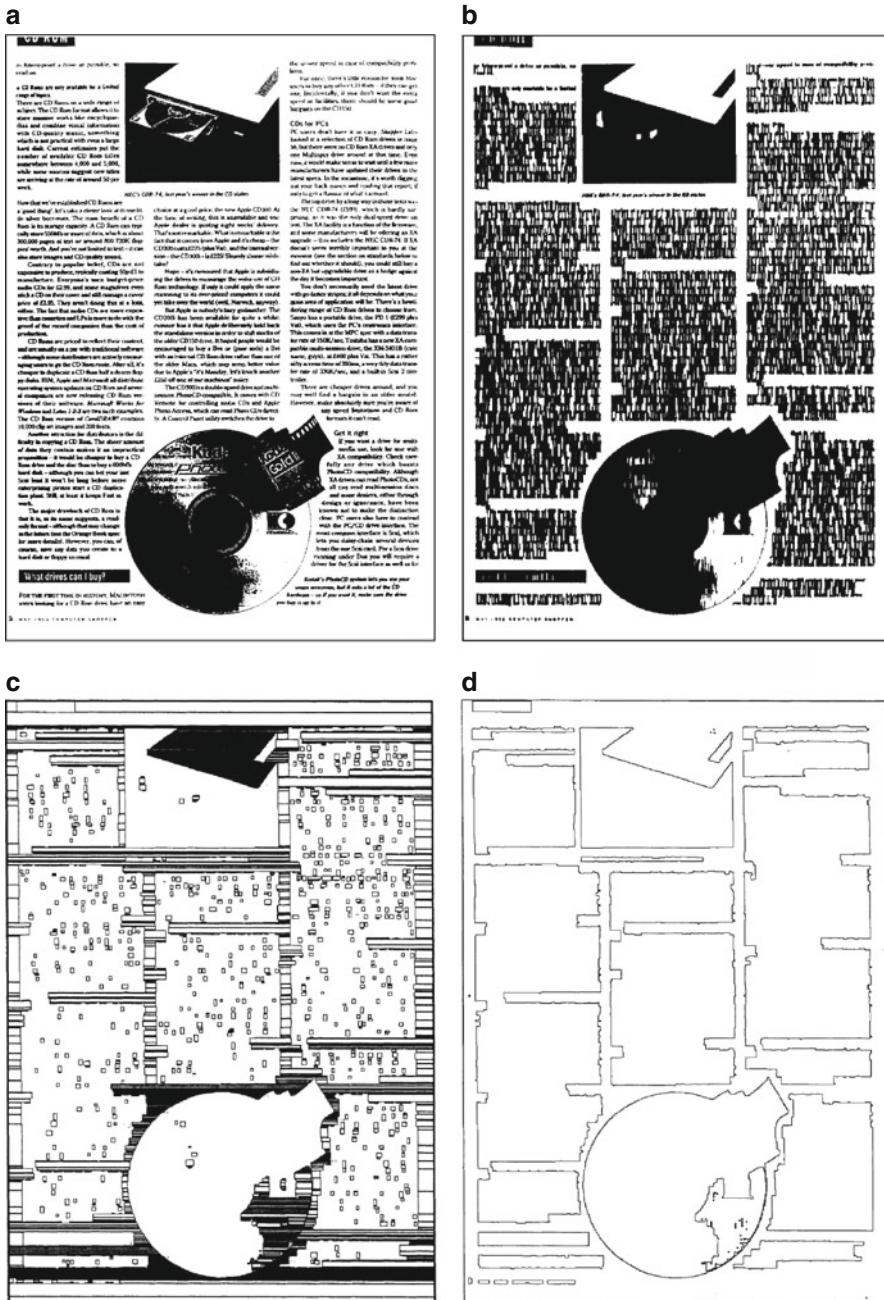
A disadvantage of the white tiles is that the representation is not rotation invariant. Thus the parameter for merging the while tiles should be determined based upon a possible range of skew angles. There is no problem if the skew can be limited within a narrow range such as in the case that documents are scanned by a flatbed scanner. If it does not hold, on the other hand, it is better to use a method with a rotation invariant representation.

Such a representation is realized by the area Voronoi diagram shown in Fig. 5.9. As shown in Fig. 5.21b, it represents the background structure of a page. Based on this representation, the task of page segmentation is to select Voronoi edges that form the borders of page components. This can be paraphrased as to remove unnecessary Voronoi edges. As the feature of deletion, Kise et al. employ simple features such as the minimum distance between CCs sharing an edge as their border, as well as the area ratio of the CCs [7]. The threshold for the distance is calculated based on statistics on distance distribution in the page. It is often the case that there is clear peaks that represent the inter-character and the inter-text-line gaps of the body text. This idea comes from the NN distance of docstrum. Note that in the case of the area Voronoi diagram, we do not have the parameter  $k$  of  $k$ -NN that affects the distribution.

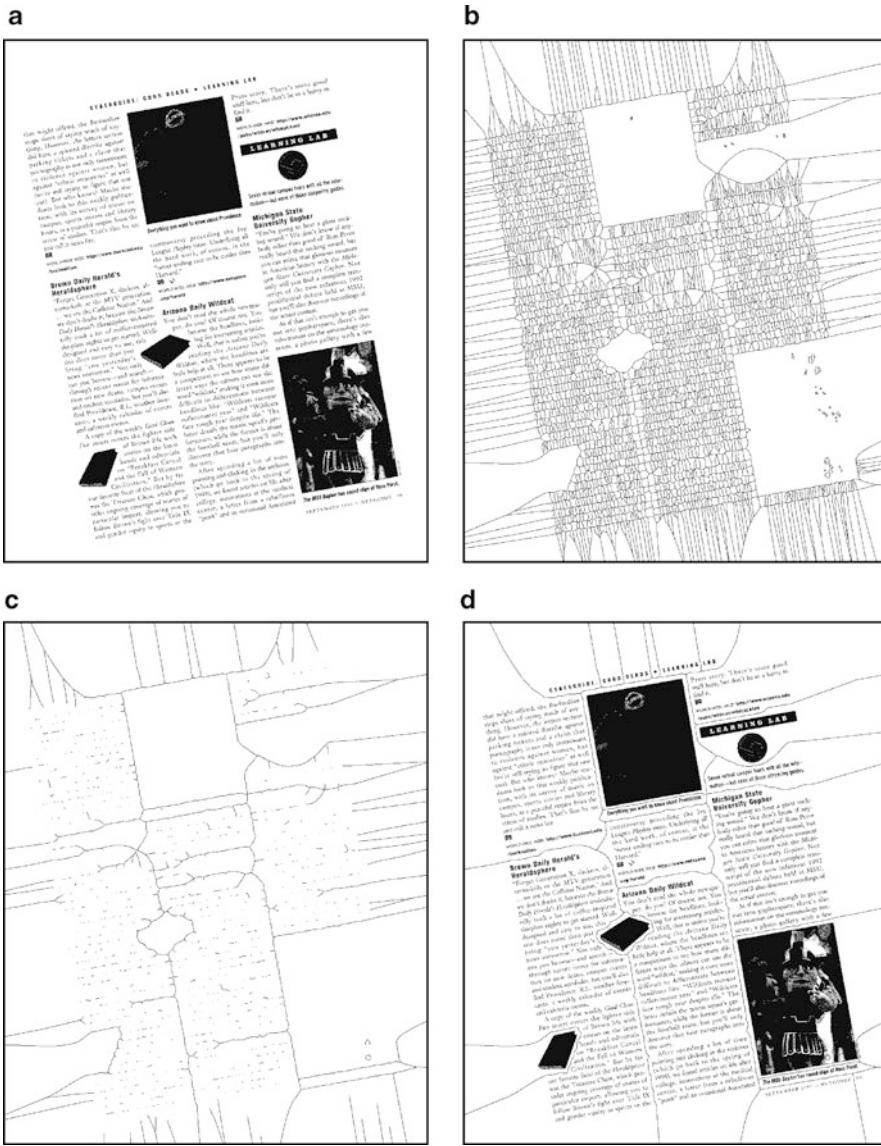
Using these values as thresholds, pages are segmented adaptively to the layout of pages. Figure 5.21c is the result of removing unnecessary edges. By further removing edges with open terminals, i.e., a terminal of an edge without connection to others, we can obtain the segmentation as shown in Fig. 5.21d.

### Comparison of Methods

We have described representative methods of page segmentation for binary images with nonoverlapping layout. For this category of layout, many more methods have



**Fig. 5.20** Page segmentation based on white tiles **(a)** Original image, **(b)** smeared image, **(c)** white tiles **(d)** segmentation result (Reprinted from Figs. 1, 7, 9, 13 of [6])



**Fig. 5.21** Area Voronoi diagram and its use for page segmentation (Reprinted from Fig. 4 of [7])

been proposed. Thus readers may be interested in finding which the best algorithm is among them.

In order to answer to this question, several research activities have been performed. A representative is the “page segmentation contest” starting as the newspaper page segmentation contest in 2001 and continued until 2009 as a

competition in biannual ICDAR conferences (e.g., [20]). The details can be found in ▶Chap. 29 (Datasets and Annotations for Document Analysis and Recognition). Another activity is the publication of benchmarking papers [21, 40]. An important conclusion of these papers is that the best algorithm must be determined based on the constraints on pages to be analyzed. For example, if the layout is rectangular and there is no skew, it is not always a good choice to use more general methods such as docstrum and the Voronoi in terms of the accuracy. Another important note is that among representative methods including docstrum and the Voronoi, there may be no statistically significant difference even when their accuracies differ [22]. Pages hard to segment by one method are also hard by other methods. In other words, the variation caused by pages is much larger than that by algorithms.

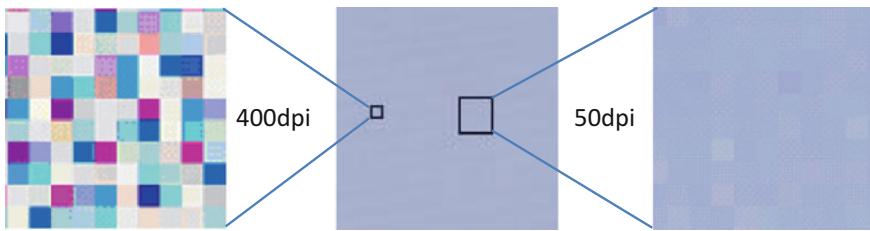
As for computation time and memory requirement, we should be careful to the following points. In methods that work directly on the image such as projection-based and smearing-based methods, the size of the image determines both the computation time and memory. Thus it is better to work not on the original image with high resolution but on its downsampled version. For methods based on larger primitives and representations such as connected component-based and background analysis methods, once the primitives and representations are extracted, computation time and memory consumption are independent from the resolution. Especially, for methods based on computational geometry such as MST and Voronoi, a lot of sophisticated algorithms are available for computing representations. Thus these methods have an advantage with respect to computation time and memory.

## Analysis of Gray-Level and Color Pages

It is sometimes inappropriate to capture documents as black-and-white images. There are two cases. One is the case that documents are inherently printed in gray-level or color. The other case is that even for documents printed in black and white, it is sometimes more appropriate to capture them as gray-level or color images due to the degradation. The latter is typical for historical documents.

Strategies for the analysis can be broadly divided into two categories: whether or not the foreground is separated from the background by preprocessing. If we can apply foreground separation, the problem of page segmentation can be transformed into that for black-and-white pages [23], and thus methods described above can be applied. Here we briefly describe the preprocessing for the purpose of page segmentation.

For gray-level images, a simple strategy is to apply a thresholding. For color images, the equivalent is clustering to obtain colors that represent foreground. Note that they are not just an application of simple thresholding or color clustering, because gray-level and color printing is based on fine dots or *dither*. If we scan such documents at high resolution, these dots are directly obtained as shown in Fig. 5.22 and make the processing difficult. Lower resolution images hardly have this problem as in Fig. 5.22. However, simple downsampling by averaging in local areas blurs



**Fig. 5.22** Effect of resolution change (Reprinted from Fig. 9 of [24])

edges of the foreground. To cope with this problem, Hase et al. proposed a method of *selective* averaging images in local areas [24]. The value of each pixel is determined by taking into account its surrounding local area whose center is the pixel. They assume that a local area contains at most two different regions. If the local area contains only a single region, the value of the current pixel at the center is set to the average of pixel values in the local area. Otherwise a simple discriminant analysis is applied to determine two regions, and the average of the region the current pixel belongs to is set to the current pixel.

If readers are interested in thresholding or color clustering, see ▶Chap. 4 (Imaging Techniques in Document Analysis Processes).

For the purpose of page segmentation, it is better not only to consider pixel values, but also features of page components. Perroud et al. have proposed a method that takes into account spatial information in addition to color information [25]. They employ a four-dimensional histogram, i.e., RGB and the Y-coordinate of a pixel, for the clustering. This enables us to take into account the vertical proximity to form a cluster of pixels.

In addition to the thresholding, we have another strategy for page segmentation of gray-level images. In gray-level images, text regions are characterized by strong edges from characters. Yuan and Tan have proposed a method that employs edge detection [26]. In their method text regions are characterized by those with horizontal edges. By grouping neighboring horizontal edges, text regions are extracted.

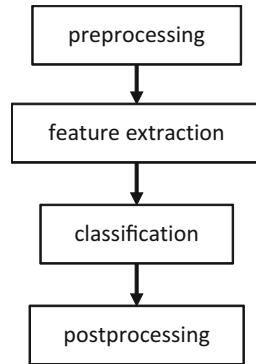
In case that the foreground-background separation is not easy to achieve by the above methods, such as the case of overlapping layout, it is necessary to solve this problem by taking into consideration features of page components at a signal level. Details are described in the next section.

---

## Analysis of Pages with Overlapping Layout

As we discussed in section “[Text-Block Level](#),” overlapping layout of type shown in Fig. 5.3d is far different from nonoverlapping layout in the sense that there is no clear distinction between foreground and background; as the case of text on a picture, one

**Fig. 5.23** Analysis of pages with overlapping layout



page component can be the background of another page component. Thus the simple separation between foreground and background is meaningless. In other words, page segmentation should be done with classification of page components. Note that some page classification methods assume that page components have already been extracted. Methods of page classification including this type are described in ▶Chap. 7 (Page Similarity and Classification). In this chapter, we focus on methods that segment page components simultaneously with their classification.

Most of the methods for this layout are based on signal properties of page components. To be precise, features and strategies based on texture analysis technologies enable us page segmentation of this class of layout. Figure 5.23 shows a general processing flow. After the preprocessing such as noise reduction, texture features are extracted from an input image, which is mostly given as gray-level or color. As a unit for feature extraction, it is common to consider each pixel of the image. A feature vector is associated with each pixel and employed to classify it. Methods of classification can be divided into two types – unsupervised and supervised. Classes of page components for classification depend on methods. The most basic classes are text and non-text. Some methods employ more detailed classes such as pictures, line drawings, and handwriting. In general classification results tend to be noisy due to limited amount of “local” information used at the classification. The task of postprocessing is, therefore, to make the results more reliable by using global constraints or contexts.

Table 5.2 shows representative methods of this category. Methods can be characterized by the items in the table: classes of page components to be extracted, features used for the classification, methods of classification, and postprocessing. In the following, details of each method are described.

Let us start with one of the pioneering work in this category, i.e., the method proposed by Jain and Bhattacharjee [27]. Their method is to classify each pixel of the input image into either text or non-text. As the feature for each pixel, they employ multichannel Gabor filters. By using filters of different scales and orientations (two scales and four orientations), the method can process images with various layouts and skew angles.

**Table 5.2** Representative methods for overlapping layout

Method	Classes	Feature	Classification method	Postprocessing
[27]	Text, non-text	Multichannel Gabor filters	<b>Unsupervised:</b> CLUSTER (3 clusters) <b>Supervised:</b> $k$ -NN ( $k = 7$ )	Heuristics (deletion of small and thin CCs, etc.)
[28]	Text + gra- phics Picture Background	Learned masks (16 masks of size $7 \times 7$ )	<b>Supervised:</b> Neural network	Morphology
[29]	Text Picture Graphics Background	Wavelet packets	<b>Supervised:</b> Local decision by neural network + decision integration (Coarse-to-fine)	Morphology (closing)
[30]	Text Graphics Background	$M$ -band wavelets	<b>Unsupervised:</b> $k$ -means	
[31]	Text Picture Background	Matched wavelets	<b>Supervised:</b> Fisher	MRF
[32]	Text Picture Background	Haar wavelet	<b>Supervised:</b> Multiscale Bayesian	
[33]	Machine- printed Handwriting Picture Background		<b>Supervised:</b> Iterated classification	Implemented as iterated classification

In their method, the feature vector  $e$  of a pixel at  $(x, y)$  is given by

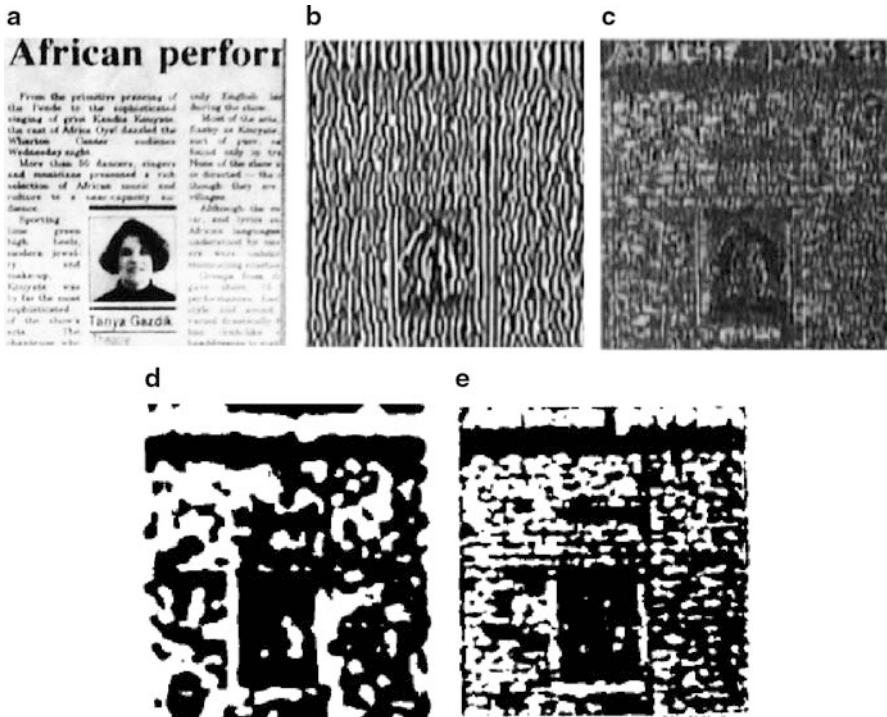
$$e(x, y) = (e_1, \dots, e_n)$$

$$e_k(x, y) = \frac{1}{M^2} \sum_{(a,b) \in W_{xy}} |\varphi(r_k(a, b))|, k = 1, \dots, n$$

where  $W_{xy}$  is a window of size  $M \times M$  centered at the pixel  $(x, y)$ ,  $r_k(x, y)$  is the  $k$ th filtered image as described below, and  $\varphi$  corresponds to a function for thresholding:

$$\varphi(t) = \tan h(\alpha t)$$

where  $\alpha$  is a parameter and typically set to 0.25. As filters to obtain filtered images, they propose to select appropriate ones from the following bank of Gabor filters with various radial frequencies with four orientations  $\theta = 0^\circ, 45^\circ, 90^\circ, 135^\circ$ . For the processing of newspaper images of size  $256 \times 512$ , they employed two radial



**Fig. 5.24** Examples of filtered images and features. (a) Original image, (b) a filtered image with the radial frequency  $32\sqrt{2}$  and orientation  $0^\circ$ , (c) radial frequency  $64\sqrt{2}$  and orientation  $0^\circ$ , (d) the feature obtained from (b), and (e) the feature from (c) (Reprinted from Figs. 2–4 of [27])

frequencies  $32\sqrt{2}, 64\sqrt{2}$ . Examples of filtered images and elements of the feature vector are shown in Fig. 5.24.

Then the feature vector is classified by two methods. One is unsupervised based on the CLUSTER algorithm. Note that the number of clusters is not two but three: text, non-text, and their border. The other method is supervised. To be precise, the  $k$ -NN with  $k = 7$  is employed for two-class classification with 4,000 training samples (pixels). As the postprocessing, they utilize some heuristics for cleaning up the results. Figure 5.25 shows an example of processing results.

The above method is based on general features that require a higher computational load. In other words, more efficient processing is possible if features are tuned for a specific purpose. To achieve this, Jain and Zhong have proposed a method using a learning capability [28]. Features are obtained by a feed-forward artificial neural network learned with the back-propagation algorithm. They first use three classes, text or graphics, picture, and background. Then after the classification, the class text



**Fig. 5.25** An example of segmentation results by the method. (a) Input image, (b) extracted text region shown in *gray color*, and (c) text segments after postprocessing (Reprinted from Fig. 7 of [27])

or graphics is further divided based on different features. As the postprocessing, they employ noise removal and morphological operators to enclose text regions.

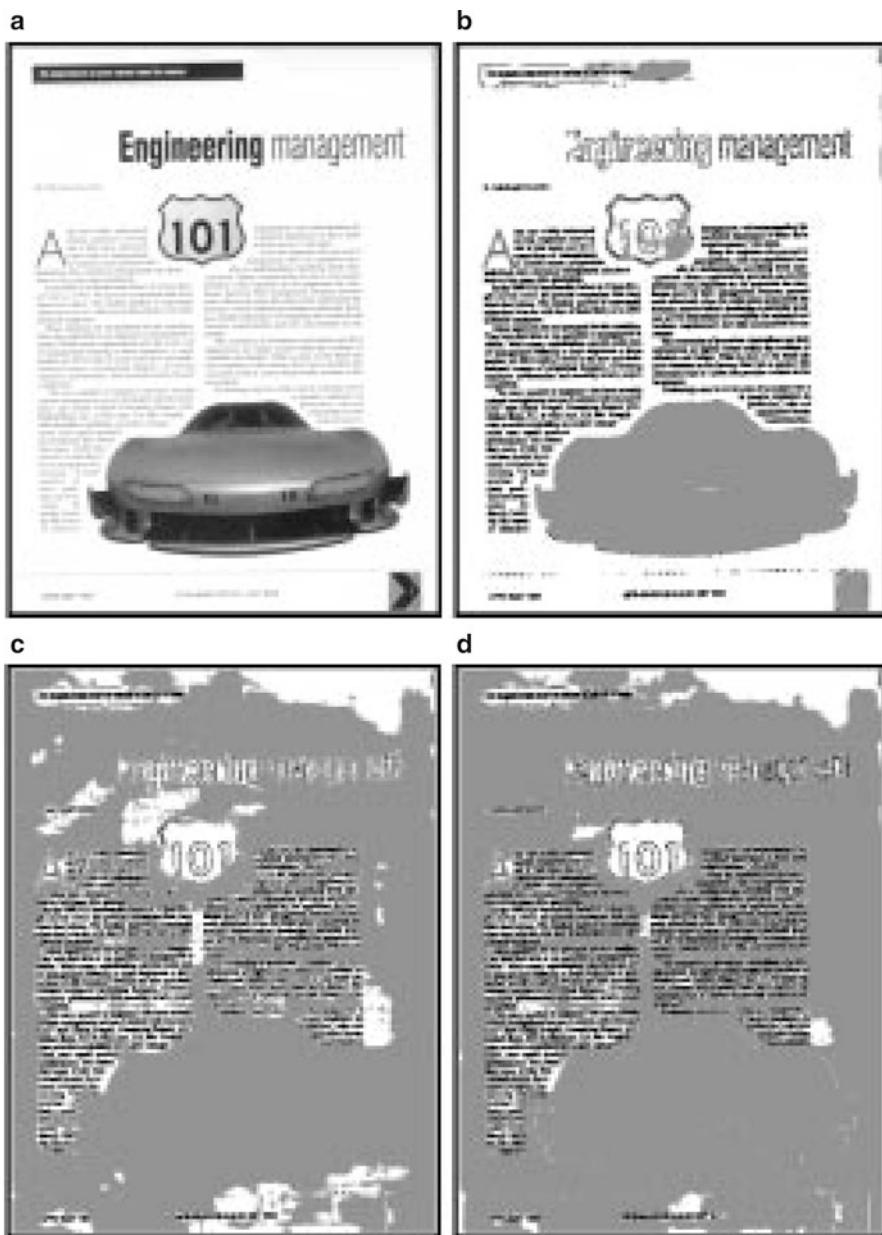
Etemad et al. have also proposed a method which employs “tuned” texture features [29]. Specifically, they used wavelet packets that can be adaptively designed based on learning data. One of the advantages of their method is that it is much more computationally efficient than Gabor filter-based methods, thanks to the fast algorithms for computing wavelet packets. Another advantage is in their decision process. Severe overlaps of page components prevent us from making hard decision based only on local information. This problem is solved by using “soft” (weighted) decision, followed by decision integration based on context information, i.e., classification results of neighboring pixels. As the postprocessing they also employ morphological operators to eliminate noise.

In addition to the above basic texture features (Gabor filters and wavelet packets), there are several approaches that employ more advanced features. Acharyya and Kundu have proposed to use  $M$ -band wavelet features ( $M > 2$ ) for better analysis of high-frequency signals as compared to the normal wavelet with  $M = 2$  [30]. Their method employs a simple  $k$ -means clustering for classification. Even with such a simple method, it has been experimentally shown that their method works better than previously proposed methods such as [27] and [29]. The method allows us results with less noise. A different way of adapting features is proposed by Kumar et al. [31]. Their method uses matched wavelets to implement a trainable segmentation together with the Fisher classifier for classification. A postprocessing method based on Markov Random Fields (MRF) is employed for improving the quality of segmentation results.

In general, most methods in this category including those mentioned above can be characterized by the following concepts: multiscale, learning, and context representation for postprocessing. Each method has their own way to deal with these concepts, but most of them lack a unified theoretical framework for these concepts. For example, some methods employ MRF for postprocessing, but it is independent of the classification and the multiscale feature representation. Needless to say, it is better to have a unified framework to achieve a better performance obtained by global optimization than a heuristic combination of these concepts.

Cheng and Bouman [32] have proposed a method to achieve this goal. Their method is based on a multiscale Bayesian framework in which a Markov chain in scale is utilized for modeling the texture features and the contextual dependencies for the image. The algorithm called trainable sequential maximum a posteriori (TSMAP), which is used to calculate the MAP estimates sequentially along with the multiscales, is capable of improving the performance with increasing the number of samples for learning as shown in Fig. 5.26.

There is another method that is worth being mentioned. All of the above methods, if they are supervised, require learning process which is in general computationally expensive. When the number of training samples becomes very large, the learning process is prohibitive due to the computational cost. This problem can be solved by using a simpler supervised method with less learning cost. For the classifier based on the nearest neighbor (NN) scheme, for example, the learning is just to record the samples with labels. The cost of matching can also be reduced by using approximate NN search. The method proposed by An et al. [33] embodies this concept. In their method, each pixel is characterized by a feature vector with 77 dimensions including a current result of classification for the pixel of interest and its surrounding pixels with a fixed radius (5–7 pixels). An interesting strategy of their method is that they do not have a single postprocessing step but with many steps of postprocessing: the method consists of cascaded classifiers, and each classifier is trained based on the result of a previous classifier. With such a multistage training and testing, error rates can be reduced even for a more complex task of distinguishing four classes: machine printed, handwriting, picture, and background.



**Fig. 5.26** Effect of training by the TSMAP algorithm. **(a)** Input image; **(b)** segmentation results when trained on 20 images where *black*, *gray*, and *white pixels* indicate text, pictures, and background, respectively; **(c)** results with 10 training images; and **(d)** results with 5 training images (Reprinted from Fig. 15 of [32])

## Analysis of Handwritten Pages

The final topic in this chapter is the analysis of handwritten pages. The task here is to distinguish handwriting from machine-printed part in a page image. As we have already seen that texture-based methods such as [33] offer a means for the distinction, many methods for this task are quite similar to those in the previous section. In this section, let us briefly describe an example, a method proposed by Zheng et al. [34].

In order to reduce the computational cost, the method employs connected components as units of processing. Thus the goal here is to classify connected components into either machine printed or handwriting. In order to achieve this goal for noisy documents, Zheng et al. define their task as classification into either machine printed, handwriting, or noise. Features used for classification are similar to those in texture-based page segmentation such as Gabor filters and bi-level co-occurrence. After the features are extracted from each connected component, the next step is classification. In the method, the Fisher classifier is applied after feature selection by principal component analysis. Since the classification is error prone due to the limited amount of information from each connected component, the postprocessing is necessary to clean the results by taking into account the contextual information. Zhang et al. employ a method based on Markov random field – another resemblance to methods in the previous section.

---

## Conclusion

In this chapter, we have described various methods of page segmentation focusing mainly on segmenting machine-printed page components such as text blocks, graphics, tables, and pictures. The research activities are in mature for nonoverlapping layout such as rectangular, Manhattan, and non-Manhattan. If the restriction on the layout is stronger, more stable methods with less computational costs are available. For the analysis of overlapping layout, methods are still under development. The difficulty of the analysis is similar to that of camera-based document analysis for unrestricted environments. We have seen that pixel-level classification by using texture features is a promising way to achieve the task.

The history of development of page segmentation methods is a series of challenges to remove the assumptions/limitations on page layout. A possible future trend in the research is to extend this challenge for adaptive and learnable page segmentation. Since most of the printed documents are prepared digitally, fully automated learning methods would be developed if appropriate degradation models of images are available.

---

## Cross-References

- ▶ [Analysis of the Logical Layout of Documents](#)
- ▶ [Datasets and Annotations for Document Analysis and Recognition](#)

► **Imaging Techniques in Document Analysis Processes**  
► **Page Similarity and Classification**

---

## References

1. Wong KY, Casey RG, Wahl FM (1982) Document analysis system. *IBM J Res Dev* 26(6): 647–656
2. Ittner DJ, Baird HS (1993) Language-free layout analysis. In: Proceedings of the second ICDAR, Tsukuba, pp 336–340
3. O’Gorman L (1993) The document spectrum for page layout analysis. *IEEE Trans PAMI* 15(11):1162–1172
4. Kise K, Iwata M, Matsumoto K, Dengel A (1998) A computational geometric approach to text-line extraction from binary document images. In: Proceedings of the 3rd DAS, Nagano, pp 346–355
5. Baird HS (1992) Anatomy of a versatile page reader. *Proc IEEE* 80(7):1059–1065
6. Antonacopoulos A (1998) Page segmentation using the description of the background. *Comput Vis Image Underst* 70(3):350–369
7. Kise K, Sato A, Iwata M (1998) Segmentation of page images using the area Voronoi diagram. *Comput Vis Image Underst* 70(3):370–382
8. Nagy G, Seth S (1984) Hierarchical representation of optically scanned documents. In: Proceedings of the 7th ICPR, Montreal, pp 347–349
9. Krishnamoorthy M, Nagy G, Seth S, Viswanathan M (1993) Syntactic segmentation and labeling of digitized pages from technical journals. *IEEE Trans PAMI* 15(7):737–747
10. Pavlidis T, Zhou J (1992) Page segmentation and classification. *CVGIP: Graph Models Image Process* 54(6):484–496
11. Srihari SN, Govindaraju V (1989) Analysis of textual images using the Hough transform. *Mach Vis Appl* 2:141–153
12. Fletcher LA, Kasturi R (1988) A robust algorithm for text string separation from mixed text/graphics images. *IEEE Trans PAMI* 10(6):910–918
13. Bloomberg DS (1996) Textured reduction for document image analysis. In: Proceedings of the IS&T/SPIE EI’96, conference 2660: document recognition III, San Jose
14. Bukhari SS, Shafait F, Breuel TM (2011) Improved document image segmentation algorithm using multiresolution morphology. In: SPIE document recognition and retrieval XVIII, DRR’11, San Jose
15. Dias AP (1996) Minimum spanning trees for text segmentation. In: Proceedings of the 5th annual symposium on document analysis and information retrieval, Las Vegas
16. Simon A, Pret J-C, Johnson AP (1997) A fast algorithm for bottom-up document layout analysis. *IEEE Trans PAMI* 19(3):273–277
17. Breuel TM (2002) Two geometric algorithms for layout analysis. In: Proceedings of the DAS2002, Princeton, pp 188–199
18. Agrawal M, Doermann D (2010) Context-aware and content-based dynamic Voronoi page segmentation. In: Proceedings of the 9th DAS, Boston, pp 73–80
19. Yin P-Y (2001) Skew detection and block classification of printed documents. *Image Vis Comput* 19(8):567–579
20. Antonacopoulos A, Pletschacher S, Bridson D, Papadopoulos C (2009) ICDAR 2009 page segmentation competition. In: Proceedings of the 10th ICDAR, Barcelona, pp 1370–1374
21. Shafait F, Keysers D, Breuel TM (2008) Performance evaluation and benchmarking of six-page segmentation algorithms. *IEEE Trans PAMI* 30(6):941–954
22. Mao S, Kanungo T (2001) Empirical performance evaluation methodology and its application to page segmentation algorithms. *IEEE Trans PAMI* 23(3):242–256
23. Strouthopoulos C, Papamarkos N, Atsalakis AE (2002) Text extraction in complex color documents. *Pattern Recognit* 35:1743–1758

24. Hase H, Yoneda M, Tokai S, Kato J, Suen CY (2004) Color segmentation for text extraction. IJDAR 6:271–284
25. Perroud T, Sobottka K, Bunke H, Hall L (2001) Text extraction from color documents – clustering approaches in three and four dimensions. In: Proceedings of the 6th ICDAR, Seattle, pp 937–941
26. Yuan Q, Tan CL (2001) Text extraction from gray scale document images using edge information. In: Proceedings of the 6th ICDAR, Seattle, pp 302–306
27. Jain AK, Bhattacharjee S (1992) Text segmentation using Gabor filters for automatic document processing. Mach Vis Appl 5:169–184
28. Jain AK, Zhong Y (1996) Page segmentation using texture analysis. Pattern Recognit 29(5):743–770
29. Etemad K, Doermann D, Chellappa R (1997) Multiscale segmentation of unstructured document pages using soft decision integration. IEEE Trans PAMI 19(1):92–97
30. Acharyya M, Kundu MK (2002) Document image segmentation using wavelet scale-space features. IEEE Trans Circuits Syst Video Technol 12(12):1117–1127
31. Kumar S, Gupta R, Khanna N, Chaudhury S, Joshi SD (2007) Text extraction and document image segmentation using matched wavelets and MRF model. IEEE Trans Image Process 16(8):2117–2128
32. Cheng H, Bouman CA (2001) Multiscale Bayesian segmentation using a trainable context model. IEEE Trans Image Process 10(4):511–525
33. An C, Baird HS, Xiu P (2007) Iterated document content classification. In: Proceedings of the 9th ICDAR, Curitiba, pp 252–256
34. Zheng Y, Li H, Doermann D (2004) Machine printed text and handwriting identification in noisy document images. IEEE Trans PAMI 26(3):337–353
35. O’Gorman L, Kasturi R (1995) Document image analysis. IEEE Computer Society, Los Alamitos
36. Dori D, Doermann D, Shin C, Haralick R, Phillips I, Buchman M, Ross D (1997) The representation of document structure: a generic object-process analysis. In: Bunke H, Wang PSP (eds) Handbook of character recognition and document image analysis. World Scientific, Singapore, pp 421–456
37. Jain AK, Yu B (1998) Document representation and its application to page decomposition. IEEE Trans PAMI 20(3):294–308
38. Okun O, Pietikäinen M (1999) A survey of texture-based methods for document layout analysis. In: Pietikäinen M (ed) Texture analysis in machine vision. Series in machine perception and artificial intelligence, vol 40. World Scientific, Singapore
39. Nagy G (2000) Twenty years of document image analysis in PAMI. IEEE Trans PAMI 22(1):38–62
40. Mao S, Rosenfeld A, Kanungo T (2003) Document structure analysis algorithms: a literature survey. In: Proceedings of the document recognition and retrieval X, Santa Clara, pp 197–207
41. Namboodiri AM, Jain A (2007) Document structure and layout analysis. In: Chaudhuri BB (ed) Digital document processing: major directions and recent advances. Springer, London, pp 29–48. ISBN:978-1-84628-501-1
42. Cattoni R, Coianz T, Messelodi S, Modena CM (1998) Geometric layout analysis techniques for document image understanding: a review. Technical report TR9703-09, ITC-irst
43. Normand N, Viard-Gaudina C (1995) A background based adaptive page segmentation algorithm. In: Proceedings of the 3rd ICDAR, Montreal, pp 138–141
44. Kise K, Yanagida O, Takamatsu S (1996) Page segmentation based on thinning of background. In: Proceedings of the 13th ICPR, Vienna, pp 788–792

## Further Reading

In addition to papers in conferences such as ICDAR and DAS, it is recommended to read some chapters in handbooks [35, 36] and survey papers [37–41] if readers are interested in finding more

methods of page segmentation. Although most of them are published before 2000, major activities of page segmentation are covered by them. The most comprehensive list of methods before 2000 can be found in [42].

The following are some additional comments we do not touch in the above.

In section “[Smearing Based Methods](#),” we introduced a smearing-based method using the mathematical morphology. A limitation of this method is that it can only deal with separation of text from halftones. A recent progress to solve this problem has been proposed by Bukhari et al. [14].

In section “[Background Analysis Methods](#),” we mainly focused on methods based on computational geometry, since they allow us efficient processing. However, there is another type of methods based on binary digital image processing. It is easy to understand that a rotation invariant representation of background can be obtained by using maximal empty “circles” [43]. This representation can be obtained by using the algorithm of skeletonization. A disadvantage of using skeletons is that they do not keep the information of connectivity among regions represented by empty circles. In the field of digital image processing, algorithms of thinning have been proposed to solve this problem. An example of using thinning for page segmentation can be found in [44].

Methods for improving the Voronoi-based segmentation have also been proposed. One of the serious problems of the Voronoi-based method is that the threshold for deleting unnecessary Voronoi edges is globally determined. Thus if a page contains page components with different sizes such as a bigger title with a smaller body text, the global threshold may cause the problem of over-segmentation for a nondominant text such as titles. This problem has been addressed by Agrawal and Doermann [18]. In this method, the deletion of Voronoi edges works adaptively depending on the “contexts,” i.e., the surrounding CCs, with the help of new features and decision rules.

---

# Analysis of the Logical Layout of Documents

6

Andreas Dengel and Faisal Shafait

## Contents

Introduction.....	178
History and Importance.....	180
Evolution of the Problem.....	181
Applications.....	183
Main Difficulties.....	184
Summary of the State of the Art.....	185
Components of a Document Understanding System.....	186
Document Structure Representation.....	186
Document Preprocessing.....	189
Geometric Layout Analysis.....	190
Document Categorization.....	192
Logical Labeling.....	193
Techniques for Logical Labeling.....	194
Rule-Based Approaches.....	194
Syntactic Methods.....	195
Perception-Based Methods.....	195
Learning-Based Methods.....	196
Knowledge-Based Systems.....	197
Case-Based Reasoning.....	197
Application Areas.....	198
Books.....	198
Invoices.....	203
Business Letters.....	204

---

A. Dengel (✉)

German Research Center for Artificial Intelligence (DFKI GmbH), Kaiserslautern, Germany  
e-mail: [dengel@dfki.de](mailto:dengel@dfki.de); [Andreas.Dengel@dfki.de](mailto:Andreas.Dengel@dfki.de)

F. Shafait

Multimedia Analysis and Data Mining Competence Center, German Research Center for Artificial Intelligence (DFKI GmbH), Kaiserslautern, Germany

School of Computer Science and Software Engineering, The University of Western Australia, Perth, Australia

e-mail: [shafait@dfki.de](mailto:shafait@dfki.de); [faisal.shafait@dfki.de](mailto:faisal.shafait@dfki.de)

Technical Journals, Magazines, and Newspapers.....	209
Other Application Areas.....	214
Experimental Validation.....	216
Performance Measures.....	216
Datasets.....	217
Recommendations.....	218
Conclusion.....	219
Cross-References.....	220
References.....	220
Further Reading.....	222

---

### Abstract

Automatic document understanding is one of the most important tasks when dealing with printed documents since all post-ordered systems require the captured but process-relevant data. Analysis of the logical layout of documents not only enables an automatic conversion into a semantically marked-up electronic representation but also reveals options for developing higher-level functionality like advanced search (e.g., limiting search to titles only), automatic routing of business letters, automatic processing of invoices, and developing link structures to facilitate navigation through books. Over the last three decades, a number of techniques have been proposed to address the challenges arising in logical layout analysis of documents originating from many different domains. This chapter provides a comprehensive review of the state of the art in the field of automated document understanding, highlights key methods developed for different target applications, and provides practical recommendations for designing a document understanding system for the problem at hand.

---

### Keywords

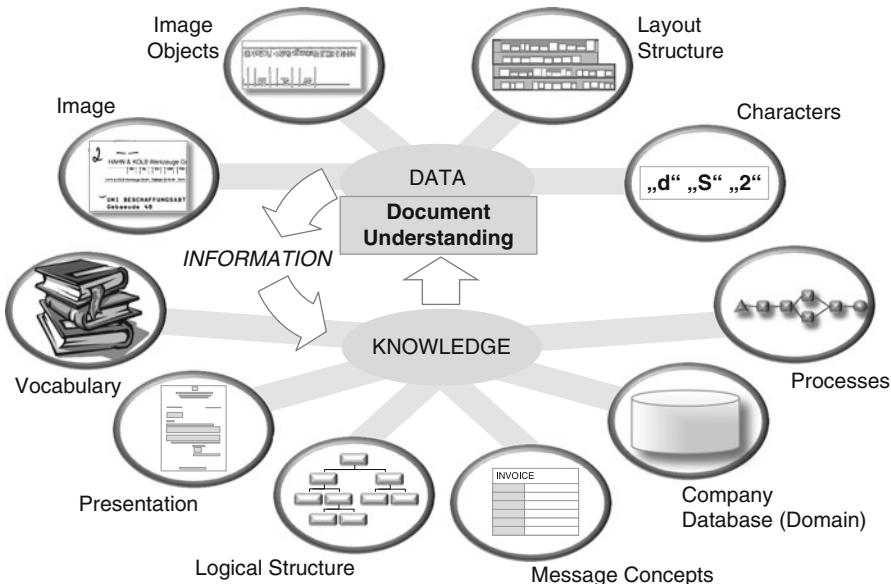
Bibliographic meta-data extraction • Document structure extraction • Document understanding • Information extraction • Invoice processing • Logical labeling • Logical layout analysis

---

---

## Introduction

Logical layout analysis or document understanding refers to the field that is concerned with logical and semantic analysis of documents to extract human understandable information and codify it into machine-readable form. This information comprises not only textual aspects such as event dates, names, or identification numbers but also logical objects which may be derived from document structure as well as from information, which is added from background databases because of content-driven relationships. In order to do so, document understanding systems provide technology to automatically transform meaningful information from a raster image into a formal representation. Hence, it is a form of reasoning in which the meaning of communication is deduced from the combination of the written text and



**Fig. 6.1** Document understanding as the process to transform data into information by applying knowledge

its presentation, the logical structure of a message, the context, knowledge of the language, as well as knowledge about the world (see Fig. 6.1).

The output of document understanding may be of different quality ranging from an editable description or a representation useful for document sorting, routing, or filing up to a structured message with attribute-value pairs describing all aspects of communication necessary to drive particular workflows. To achieve this quality of information, the level of data present in a document has to be enriched to the desired level. Before doing so, a paper document has to be converted into an electronic form which is suitable for the application of document understanding techniques.

- At a first glance a document is an object describing data of different types:
- Pictographic data where the whole image is represented as a sequence of orthogonal pixel runs
  - Segment data describing homogeneous color or texture image regions
  - Layout data describing the presentation of objects, i.e., their geometry, order, and nesting
  - Character data indicating that multiple glyph images have the same ASCII code

Initially, none of these different data types have information value. An analogous situation occurs when an ordinary European holds a letter in her/his hands written in Kanji or Hangul characters. For her/him, this letter is nothing more than just data: an image with segments arranged in a certain order and some of them look the same.

The conversion from paper to electronic data of the four qualities described above is the initial step of document understanding providing an electronic representation that is already sufficient for many applications, such as full text indexing, electronic editing, or reuse of document content.

## History and Importance

Documents are means of communication among individuals or institutions. They are created to announce conferences or new products, to summarize agreements, to comprise paragraphs of contracts, or even law, to publish innovative research results, or just to arouse, to promise, to inform, to question, to command, and to convince others. Documents are always present where information meets with human beings whose work gets done through documents. With the intention to assist these work processes, the question is how to make documents data useful. Document understanding plays a major role in extracting structured data from these documents such that it can readily be used effectively.

One of the stimulating factors that drove the initial progress in the field of document understanding was the advent of digital libraries. The objective of an electronic library is to create an electronic analog of a traditional library environment in each user's home or office. One of the first initiatives in this direction was the RightPages image-based electronic library for alerting and browsing technical articles, developed at AT&T Bell Laboratories [33]. The system provided users with online library services comprising stacks of journals. This was made possible by a complete document understanding pipeline including preprocessing, layout analysis, logical labeling, and text processing. A user interface was also developed that gave its user the look and feel of a conventional library. Another system, called Gobbledoc [23, 27], was developed by Nagy et al. to facilitate storage and browsing in digital libraries containing a large collection of technical journals. They developed an integrated segmentation and logical labeling approach for journal articles that became widely known as the recursive X-Y cut algorithm.

Desktop publishing also emerged as a new trend in the publishing industry in the mid-1980s. When reusing already printed material, a high demand for automating keyboard input data arose where large amounts of documentation had to be converted into a computer-readable form for data entry. Hence, text reader systems that could automatically convert a page into an electronic representation were required. Tsujimoto and Asada [35] developed a complete text reading system to fulfill these demands, contributing new approaches for document analysis, document understanding, and character segmentation/recognition. They defined document understanding as a transformation from a geometric structure into a logical one. A small number of rules were developed to carry out this transformation, based on the assumption that document layout is designed to seamlessly convey its logical components to humans.

Contrary to the previously mentioned complete systems for document analysis and processing on heterogeneous collections of documents, approaches for improved performance on a limited set of target documents also emerged at the same time. Bayer et al. [3, 29] developed a specific language called FRESCO for representing concepts of structured documents. FRESCO modeled the conceptual entities of structured documents from the very low level of connected components to high-level logical objects for a particular document class, like an IEEE journal.

Using syntactic analysis, desired logical objects (like title, authors, and page number of a technical article) were extracted.

Increasing use of personal computers in office environments also created the so-called media gap. Many companies desired to convert existing as well as incoming paper documents into an electronic representation for better information management including content-based retrieval and distribution. Hence a pressing need for document analysis systems that could be used as intelligent interfaces between paper and electronic media arose. Dengel [10, 12] presented a system called ANASTASIL to identify important conceptual parts (logical objects) within business letters, like sender, receiver, or company-specific printings. The system worked independently of text recognition and utilized only geometric information sources to assign logical labels to the segmented image regions, which provides the basis for an expectation-oriented text recognition, i.e., using controlled vocabularies.

The abovementioned early initiatives in the field of document understanding triggered a lot of research in this area in the later years. The next section outlines how the field evolved over the years while catering the demands of a rapidly changing market.

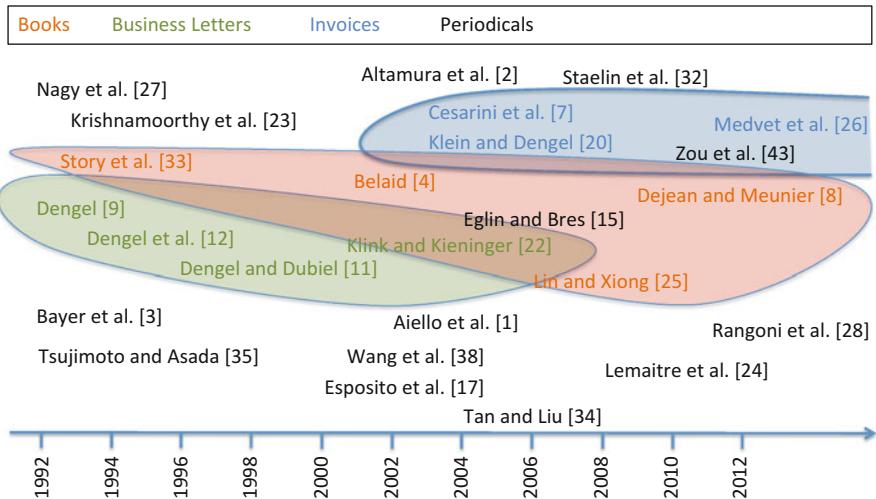
## Evolution of the Problem

Document understanding systems after over two decades of research have become not only more robust and accurate but also more versatile. Naturally, expectations from document understanding systems have also significantly increased. The capabilities of a document understanding system can be judged from various perspectives. Complexity and diversity of documents that can be handled by a particular system, besides performance (speed and accuracy) on target document collection, are often used to characterize it. Document complexity refers in this context to the layout as well as to the number of document objects on the document page. Diversity of a document collection refers to the number of domains (e.g., business letters, invoices, magazine articles) of documents present in that collection. From these perspectives, document understanding systems can be roughly classified into four classes of increasing capabilities [1]:

- Systems processing simple documents of one specific domain
- Systems able to process simple documents from multiple domains
- Systems processing complex documents of one specific domain
- Systems able to process complex documents from multiple domains

It should be noted that systems able to process complex documents are a superset of systems processing only simple documents. However, the systems able to process documents from multiple domains are not necessarily a superset of system processing one specific domain, as they might not achieve the same performance as that of specifically designed systems.

If one looks at the chronological development of the field, one can see that system presented in the early 1990s focused mainly on processing simple documents of one

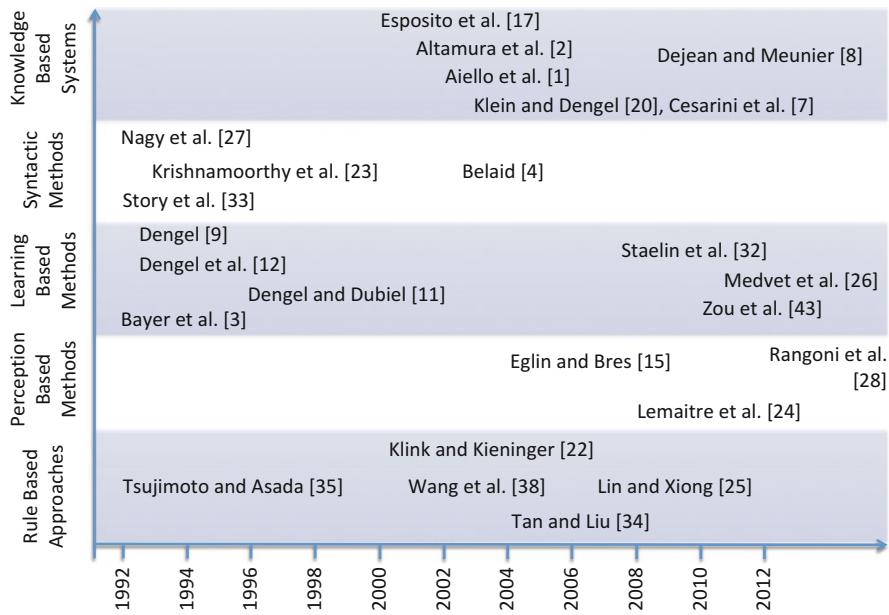


**Fig. 6.2** Evolution of the application domains of document understanding systems

specific domain. Some prominent examples of such systems are the RightPages system from AT&T Bell Laboratories [33], the FRESCO system from Bayer et al. [3], and the Gobbledoc system by Nagy et al. [27] for analyzing journal articles and the ANASTASIL system by Dengel [12] for logical labeling of business letters. One exception is the system by Tsujimoto and Asada [35], which can process simple documents from multiple domains including journal articles, letters, manual, newspaper, and magazines. An overview of the evolution of different application fields of document understanding systems is shown in Fig. 6.2.

Further progress in the document analysis field led also to development of more robust logical labeling systems, particularly with the capabilities of handling more complex documents. Towards the end of the last century, many approaches were already proposed to handle complex documents of specific domains. Examples of representative work in that direction are the DAVOS business letter analysis system by Dengel and Dubiel [11] and the INFORMys invoice reading system by Cesarin et al. [6]. Both of these systems are capable of learning and extracting logical structure of documents with diverse layouts.

The first decade of the twenty-first century has seen further evolution in document understanding systems based on ideas from a diversity of theoretical foundations (see Fig. 6.3). Not only more systems were presented to handle heterogeneous documents from one domain (e.g., [2, 7, 28]), but also systems capable of processing complex documents from different domains (e.g., [1, 15, 22, 26]) were proposed in the literature. The former category of methods uses knowledge about the domain of documents in a learning framework to handle a variety of complex documents of that domain. The latter category of methods, on the other hand, focuses on extracting knowledge about specific logical labels automatically



**Fig. 6.3** An overview of evolution of underlying theoretical foundations of document understanding systems

from the training data comprising documents from multiple domains. Using this knowledge, they are able to characterize different logical labels based on their meaning and are thus capable of identifying those logical labels even in complex documents from different domains. However, the generalization capabilities of current document understanding systems are still far away from human capabilities. Further evolution of the field towards generalized systems that deliver high performance on heterogeneous document collections from multiple domains is expected to be seen in near future.

## Applications

Understanding the contents of a document is crucial for many automated document analysis processes. Due to its vital role in making documents data useful, document understanding is a key component of document management systems employed in postal services, banks, incoming mail sorting of large organizations, and so on. Besides, digital library projects aimed at digitizing a particular collection of books, magazines, or newspapers also require to extract logical structure of the digitized material. Availability of logical structure facilitates navigation and advanced search inside the document as well as enables better presentation of the document in a possibly restructured format. Although document understanding is being used today

in a very wide spectrum of applications, the most prominent areas where it brings the highest value are outlined below. These application areas are described in more detail in section “[Application Areas](#).”

- **Creation of Digital Libraries:** In digital library projects, the aim is to digitize the books in the library so that they can be read online. The main role of logical labeling in this context is to identify table of contents pages in the books and to link individual entries in the table of content pages to the corresponding chapters/sections. Besides, if the document contents need to be reflowed, it is also desirable to identify page headers/footers, footnotes, and section headings. Similarly, when digitizing scholarly material like technical journals, it is often required to extract titles and authors of each article to facilitate advanced search within the digitized collection. Extraction of such logical entities is also a major application area of logical labeling.
- **Incoming Mail Sorting:** Many large organizations receive a large number of paper documents in incoming mail. To allow better information management and distribution, these incoming paper documents are converted into a structured electronic representation. A document understanding system is required at this stage to route the documents to the appropriate department/person in the company. This is achieved, for instance, by identifying key components representing certain concepts in a letter, like sender, receiver and subject, and then using business rules to appropriately forward the letter to the corresponding department.
- **Analysis of Invoices:** Another common application of document understanding is in automatic analysis of invoices. Invoices are also involved in daily workflows of many companies. Automatic analysis of invoices to extract header and position data and to make plausibility checks of the invoice items is the main contribution of document understanding in this domain.

## Main Difficulties

In order to extract all relevant information from a document image, the employment of knowledge is of vital significance. Knowledge is a term used very often with little meaning. However, for the task of document understanding, one may refer to knowledge as various intuitively comprehensible sources that one uses in daily business to capture the important bits of information driving different decisions and processes (cf. Fig. 6.1).

First of all, there is the terminology of communication consisting of the vocabulary of a language enriched by special designators and identifiers of a domain. A layout structure may be knowledge as well, especially if it is typical of a certain class of documents. Consider, for example, the rectangles in the document image shown in Fig. 6.4. Although they represent simple geometric data, it can be easily reasoned that they describe the typical layout of an invoice. Furthermore, it is easy to give hypotheses about where certain logical objects such as recipient or position data may be located. Neglecting the aspect of presentation, it is possible to describe,

**Fig. 6.4** Without having any knowledge about the textual contents of a document, in many cases it is possible to identify the category of that document just by looking at its layout. It is easy to identify that the document image in this figure is an invoice



for example, a business letter by just its defining logical constituents, such as sender, recipient, and subject. The structure can be further refined in some aspects. For example, a recipient is composed of a name and an address.

In addition to these complementary structural views, the various message concepts can be more restrictively defined in relevance to the context in which a document is appearing. For instance, in a corporate environment, the documents one receives correspond to one's role and tasks in that company, for example, "notification of claim" and "appraisal report" in an insurance transaction, or "invoice" and "delivery note" in a purchasing department. Furthermore, all of these sources are only valuable if the insured person, property, and value are known, or if knowledge about customers, suppliers, and products is accessible either from the company's databases or other devices.

If a human being is going to interpret the semantics behind a document message, he or she makes use of these kinds of knowledge sources either as tacit experience or by employing explicit expertise on the computer. It is one of the big challenges to combine these knowledge sources in a way that the relevant messages captured in the bulk of documents can be extracted automatically.

If this was possible, the extracted information might be added to the knowledge repository and used for later processing. For example, if a company receives an invoice, the existing information from the quote and from the order may be utilized as knowledge. Thus the process described in Fig. 6.1 also shows aspects of learning.

## Summary of the State of the Art

In transforming the model shown in Fig. 6.1 to a computer implementation, it is important to first consider the diversity of documents encountered in different application of document understanding systems. Based on variations in structural aspects with respect to layout and logical composition, documents can be categorized into three major classes:

1. Highly structured documents, which have a static form built on precisely defined page layout. They can be described by geometric templates including the regions of interest (ROIs) where the relevant information to be extracted is located. Approaches for analyzing highly structured documents, like form processing systems [14, 42], are described in ►Chap. 19 (Recognition of Tables and Forms) and hence are not covered in this chapter.
2. Semi-structured documents allowing a partial employment of templates. These are documents having only some regular geometric features to identify either the class of document or some of the ROIs, i.e., dynamic forms or business letters which have preprinted headers or footers and an empty page body for the message. Analysis of semi-structured documents is the major focus of this chapter.
3. Loosely structured documents which cannot be characterized by a geometric template. They contain typical logical objects hidden in an irregular page layout.  
For all these classes of documents, there have been various approaches published in the past, some of which are applied to a single domain of documents, while others are applied to a number of different domains. Most techniques combine geometric template matching and character recognition in order to extract keyword lists generated by post-OCR fuzzy match approaches for indexing and retrieving the contained text. However, there are also successful approaches extracting structured messages.

Figure 6.5 shows an attempt to categorize some of the important publications with respect to highly structured, semi-structured, and loosely structured documents. Although a large number of publications exist on logical layout analysis, the literature review here is limited largely to archival publications only (journal papers or book chapters).

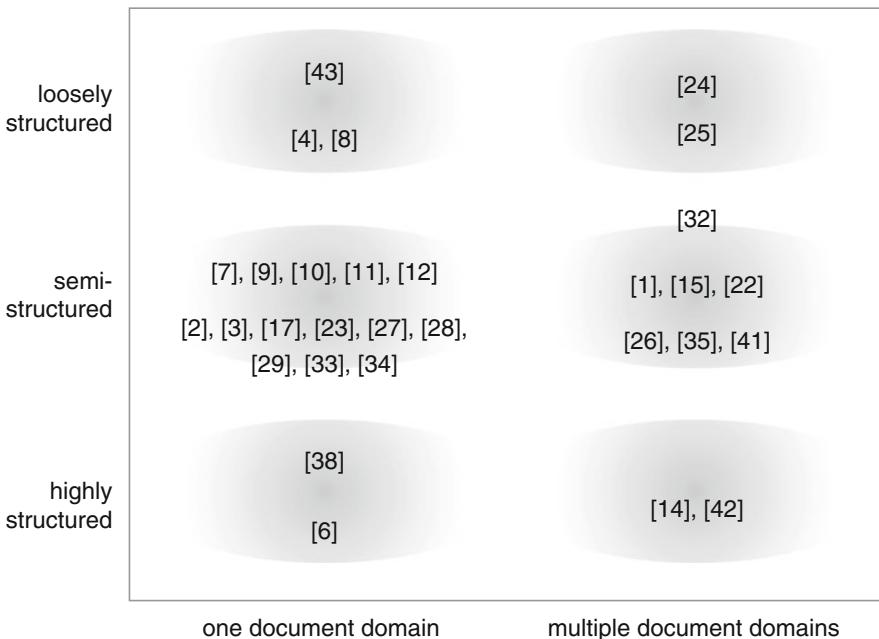
---

## Components of a Document Understanding System

Understanding a document image typically involves different processes (see Fig. 6.6). The exact order in which these processes are applied varies from one algorithm to another. Also, some algorithms might skip one or more of these processes, add some other processes, or apply them in a hybrid way. However, most of the document understanding systems use these processes in some form. Most of these processes have already been discussed in detail in previous chapters. However, for completeness, a brief outline of these processes and their role in a document understanding system is given here.

### Document Structure Representation

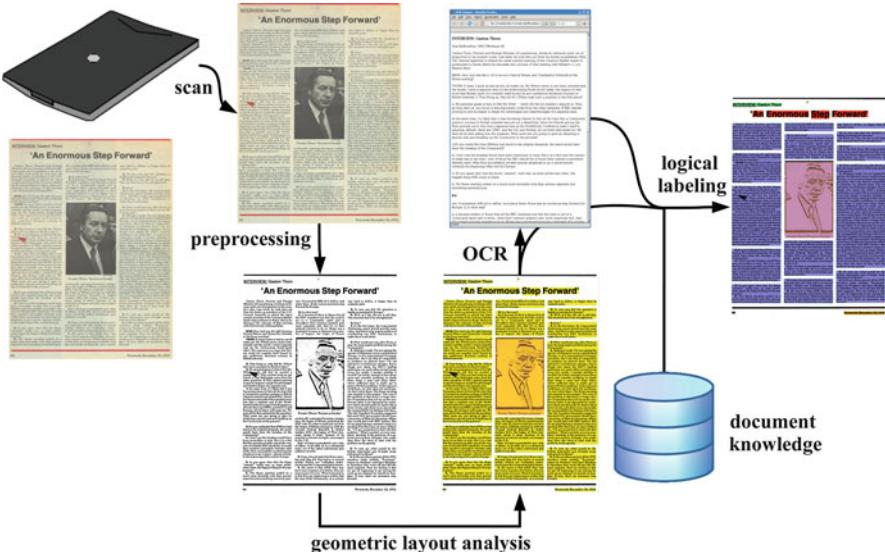
Models for document structure representation usually consist of two parts: the physical and the logical part. The physical part encodes the layout information



**Fig. 6.5** An overview of the state-of-the-art logical layout analysis approaches for documents of varying complexities and domains

of the document (e.g., a page consists of some blocks, a block consists of some text lines, a text line consists of some words), whereas the logical part represents how the content of the document is organized into related logical units (e.g., title, heading, page number). Due to hierarchical nature of documents, one of the earliest and most successful document models was a geometric or X-Y tree [12, 23, 27, 35]. The tree-based document models were later generalized to graph-based models [1, 26] to handle more complex layouts (for a more detailed discussion of document models, please refer to ▶Chap. 7 (Page Similarity and Classification) – section “[Page Representation](#)”).

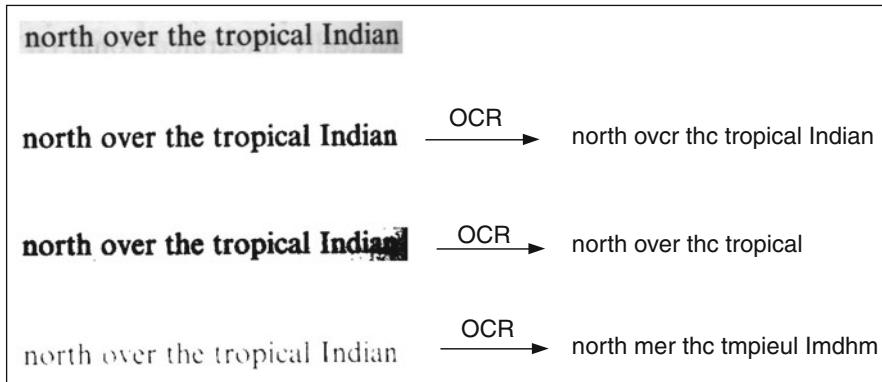
The tree-based models consider a document page as a rectangle, having a characteristic width and height. To describe its spatial structure, the page is divided into smaller rectangles by horizontal and vertical cuts. Model cuts are placed in white-space areas such that they do not intersect with textual or graphical areas. The sub-rectangles can be recursively divided in the same way, until the layout of the page is described in sufficient detail. To annotate the logical structure, different rectangles are assigned a label which describes their logical meaning. Advantages of such models include simplicity, a natural representation of page hierarchy, direct correspondence between physical and logical structure, and direct integration with algorithms of page decomposition and logical labeling. However, they limit the



**Fig. 6.6** An illustration of typical components of a document understanding system. The exact order in which these processes are applied (and which of the subcomponents are needed) varies from one approach to the other. For instance, many methods do not need OCR results for logical labeling

class of documents that can be handled to Manhattan layouts only. Besides, they only allow one layout structure per document page and one logical structure per page. However, complex document requires several views on both the layout and the logical information.

To adequately capture the structure of complex documents, generalized models represent a document as a set of layout or geometric structures [1, 26]. The set of layout structures is a collection of views such that each view represents a different layout interpretation of the document. Each layout structure itself is a set of geometric document objects and a set of geometric relations among them. Each type of geometric relation is represented as a graph. The vertices are document objects, and an edge represents a relation between the document objects. This graph can be a tree for a simple relation, but in general, it is a directed graph. The set of logical structure is represented in a similar way as a collection of views, where each view corresponds to a particular logical interpretation of the document. Due to this flexible representation, document objects are not required to have rectangular shapes, and one can define complex relations between document object. Besides, the possibility to have multiple views on the layout and logical structure benefits the labeling algorithms by allowing them to indicate multiple hypotheses as valid representations of the document.



**Fig. 6.7** The effect of choosing different binarization thresholds on the image and the resulting OCR output

## Document Preprocessing

### Binarization

Binarization is the process that converts a given input greyscale or color document image into a bi-level representation. The majority of document analysis systems have been developed to work on binary images [5]. The performance of subsequent steps in document analysis like page segmentation or optical character recognition (OCR) heavily depends on the result of binarization algorithm. Binarization with a high threshold results in merged components, which are difficult to recognize with an OCR system. On the other hand, binarization with a low threshold results in broken characters that are again a problem for OCR. An example of OCR results on differently binarized images of the same greyscale image is shown in Fig. 6.7.

Several approaches for binarizing a greyscale and colored documents have been proposed in the literature. Details of different binarization strategies are given in ▶Chap. 4 (Imaging Techniques in Document Analysis Processes) – section “Document Image Binarization.”

### Noise Removal

Different types of noise can be present in document images depending on the type of document degradation involved. Paper positioning variations usually result in marginal noise, making it hard to distinguish between the actual page contents and extraneous symbols from the neighboring page. Furthermore, non-textual noise (black borders, speckles etc.) may appear along the border of the page image as a result of binarization. Presence of border noise in document images might adversely affect the performance of page segmentation [30] or optical character recognition [31] modules in a document understanding system. Reliable removal of marginal noise under a wide diversity of conditions is still a challenging

problem. Details of different state-of-the-art noise removal algorithms can be found in ►Chap. 4 (Imaging Techniques in Document Analysis Processes) – section “[Document Image Enhancement](#).”

### **Skew and Orientation Detection**

In large-scale document digitization, skew and orientation detection plays an important role, especially in the scenario of digitizing incoming mail. The heavy use of automatic document feeding (ADF) scanners and moreover automatic processing of facsimiles results in many documents being scanned in the wrong orientation. These misoriented scans have to be corrected, as most subsequent processing steps assume the document to be scanned in the right orientation. In literature, no clear definition or distinction between page skew and orientation can be found. Skew detection methods are often presented with minimum and maximum rotation angles that can be detected. For orientation detection, some authors consider only upside up and upside down as possible orientations, whereas others also consider upside left and upside right as possible orientations. Although methods for skew and orientation detection are still being proposed [18, 37], these are generally considered as more or less solved problems for typical application scenarios (for a more details, please refer to ►Chap. 4 (Imaging Techniques in Document Analysis Processes) – section “[Document Image Normalization](#)”).

## **Geometric Layout Analysis**

### **Text/Non-text Classification**

Text/non-text classification is a key component of geometric layout analysis. Given a document image, the goal of page segmentation is to perform a decomposition of the document image into smaller zones or segments. The segments thus obtained are classified as containing text or non-text elements. The text segments or zones are then fed to a character recognition module to convert them into electronic format. If a page segmentation algorithm fails to correctly segment text from images, the character recognition module outputs a lot of garbage characters originating from the image parts. Figure 6.8 shows the case of an image merged with a text segment. When such a segment is fed to an OCR system, it outputs a large number of garbage characters in an attempt to classify the image portions as text. ►Chapter 7 (Page Similarity and Classification) – section “[Region Classification](#)” outlines several methods to classify a given block or region of a page into text and non-text elements.

### **Page Segmentation**

The task of page segmentation is to divide a document image into homogeneous zones, each consisting of only one physical layout structure (text, graphics, pictures, etc.). If the document contains more than one text column, the page segmentation algorithm should segment all text columns separately so that the text lines in different text columns are not merged together. Owing to the central role of page segmentation in OCR system, several page segmentation algorithms have

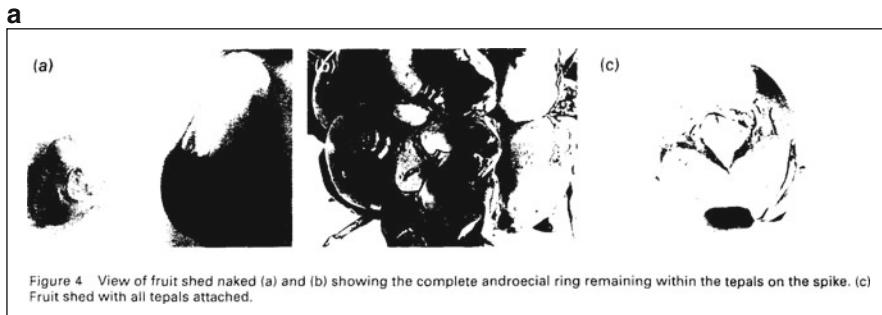


Figure 4 View of fruit shed naked (a) and (b) showing the complete androecial ring remaining within the tepals on the spike. (c) Fruit shed with all tepals attached.

Figure 4 View of fruit shed naked la) and lb) showing the complete androecial ring remaining within the tepals on the spike. (c) Fruit shed with all tepals attached.

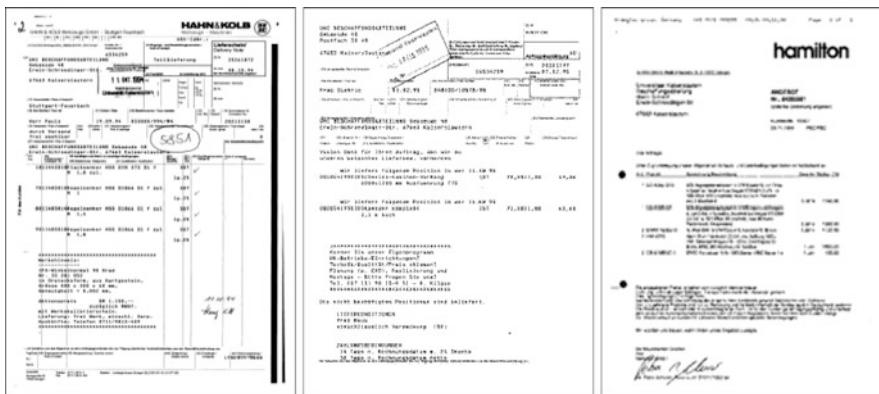
**Fig. 6.8** The OCR result of an in-correctly segmented zone containing both images and text. The OCR system generates many garbage symbols from the non-text parts of the input page segment  
**(a)** Input page segment. **(b)** OCR result

been reported in literature for last three decades (please see ►Chap. 5 (Page Segmentation Techniques in Document Analysis) Section “[Analysis of Pages with Nonoverlapping Layout](#)” for details). It should be noted that for logical layout analysis applications, it is also of vital significance that different text blocks corresponding to different semantic concepts (like author and title) are segmented into separate blocks in this step.

## Table Detection and Labeling

In many practical applications, the documents to be analyzed (e.g., bank statements or invoices) do contain important core information in tables. To extract tabular information, one has to take into account that tables have no fixed position but rather can be found more or less anywhere on a page. Furthermore, tables appear in a large variety of styles. Therefore, to locate and analyze tables in document images, two broad categories of table structure extraction approaches have been proposed in the literature, namely, model-based approaches and model-free approaches.

Model-based table recognition allow the definition of specific table models which describe textual or layout features [16]. Model-free approaches, on the other hand, attempt to locate tables based on their geometric structure [39]. Model-based approaches are not universally applicable and fail for a large number of tables. However, since they are domain specific, they generally work better than model-free



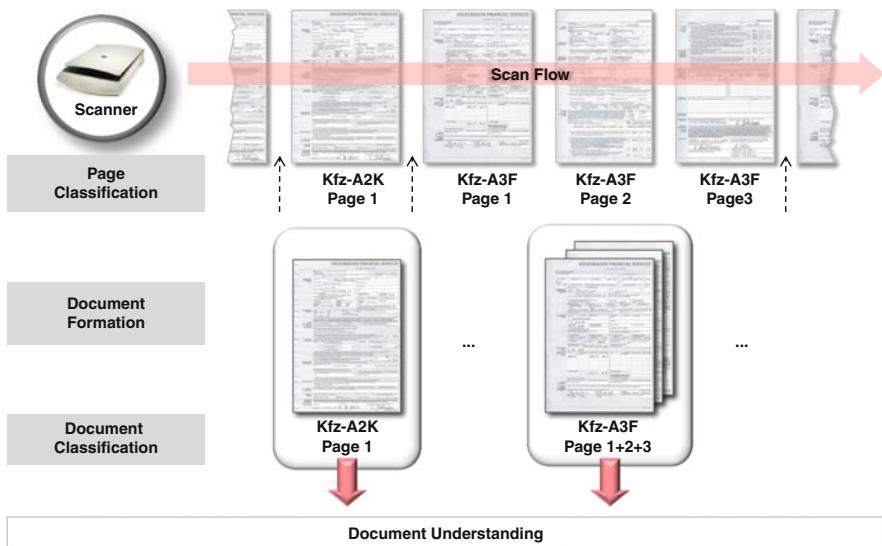
**Fig. 6.9** Typical examples of documents capturing the same message concepts but showing different degrees of structural complexity

approaches on the targeted domains. The main advantage of the model-based approaches is that they can rely on logical table models, which are defined by the column headers and corresponding content types specified via regular expressions. These models are then used to guide the analysis process. As a result, tables cannot only be located but also categorized or even semantically interpreted at the same time. Details of different table structure understanding approaches can be found in ▶Chap. 19 (Recognition of Tables and Forms) – section “[Consolidated Systems and Software](#).”

## Document Categorization

Document categorization is usually an important prerequisite for performing logical layout analysis, since the logical labels that one needs to extract from a document heavily depend on its category. For instance, in a scientific publication, one might be interested in the author, title, or abstract, whereas in a business letter, the semantic entities one would like to extract might be completely different (like sender, receiver, and date). In large-scale applications dealing with heterogeneous document collections, there is a need to categorize the documents before one can meaningfully extract logical components from a document.

One typical example is the bulk of documents at the mail entry point of a medium to large-sized organization. Such organizations receive thousands of documents daily, with widely varying structural complexity. The examples in Fig. 6.9, for instance, carry the same message concept “invoice” with the same relevant bits of information but show different levels of complexity. Furthermore, there may be single-page or multipage documents of different formats and paper quality. In many practical examples, documents having a different structural complexity as well as different formats are found in a single mail envelope. For example, in the domain of



**Fig. 6.10** A typical flow of a scanning process illustrating how page and document classification are employed before document understanding

health insurance, medical invoices are often sent with prescriptions or medical estimates. It is important to note here that a single document usually consists of multiple pages. Hence during the scanning process, one either needs to explicitly specify separator pages (e.g., placing a blank page after all pages of a document), or algorithms need to be developed to automatically cluster consecutively scanned pages into individual documents. A typical flow of the scanning process is shown in Fig. 6.10.

Consequently, there is a need to categorize many thousands of documents a day and further to determine how to proceed depending on the topic described. Document categorization addresses the task to automatically determine that a document belongs to a certain predefined class allowing routing or archiving of documents, the search for notes in online service systems, to respond to customer requests, to extract messages relevant to workflow, and many more practical applications. Categorization relies on methods that organize documents by structure or by content. Details of different methods for page or document classification can be found in ►Chap. 7 (Page Similarity and Classification) – section “[Page Classification](#).”

## Logical Labeling

The design of a logical labeling system requires carefully choosing a document structure representation that is able to capture the structure of most complex documents one expects the system to encounter and to develop a methodology using

that document model to extract relevant logical labels from a target document. The next section outlines key approaches for logical labeling grouped according to their theoretical foundations.

---

## Techniques for Logical Labeling

A large number of techniques have been presented in the literature for logical labeling as outlined in section “[Summary of the State of the Art](#).“ These techniques originate from many different concepts in computer science and artificial intelligence. In the following, an overview of different categories of document understanding techniques is given based on their underlying principles.

### Rule-Based Approaches

When a document is created, some of the logical information is encoded in the document using layout typesetting conventions, e.g., by using a specific font size and style to represent a heading. Therefore, the layout structure of a printed document carries a significant amount of information about its logical structure. Particularly for simple documents, a human reader can usually determine the logical structure from layout and typesetting information only. These formatting rules used to produce the page can also be employed in the reverse process of logical labeling. Hence, a large number of document understanding approaches in the literature (e.g., [[22](#), [25](#), [34](#), [35](#), [38](#)]) perform logical labeling based on document-specific rules and use many features derived from the document. These features may include the relative and absolute position of a block on a page, the relative and absolute position of a field within a block, general typesetting rules of spacing, and the size and font style of the text. However, the wide variation of formatting style across different documents usually does not allow a comprehensive coverage of rules for a large number of document classes. Hence, most of the rule-based approaches are restricted to particular domains.

A representative rule-based method for document structure understanding is presented by Klink and Kieninger [[22](#)]. Their system consists of several stages. First, the scanned document is segmented, and the characters and their fonts are recognized by a commercial OCR system. Then, common document structures such as header, footer, lists, and tables are recognized to refine the output of the OCR system. Finally, domain-dependent logical labeling is performed on the identified text blocks. A rule-based approach is used in the logical labeling step. For each label, exactly one rule is defined. A rule in their system is not a simple if-then rule but rather consists of a logical expression which contains the so-called rule units combined with the logical operations AND ( $\wedge$ ), OR ( $\vee$ ), NOT ( $\neg$ ), and parentheses. Hence, for assigning a particular label, logical expressions like  $(A \wedge B) \vee (C \wedge \neg D)$  can be evaluated. While evaluating the logical expression, each rule unit (e.g.,  $A$  or  $B$ ) is matched against the block actually investigated. A rule unit is divided into two parts defining self-related attributes and cross-related attributes.

The self-related attributes define only the features of the block actually inspected and consist of both layout (geometric) and textual features. These might include dimensions of the block, its position in the page, number and alignment of lines, font attributes such as font family, size and bold format as well as a list of strings contained in the block, e.g., “with best regards” or a regular expression. The cross-related attributes consist of geometric relations (e.g., relative position of two blocks), textual relations (e.g., common words among two blocks), and label relations (e.g., presence of a specific labeled block directly above the current block). Based on these rule units, logical expressions can be defined to represent a particular label within a domain. In the recognition phase, a fuzzy rule matching approach is used to assign logical labels to the input page segments.

## Syntactic Methods

Syntactic approaches for logical labeling attempt to draw an analogy between the structure of a document or page and the syntax of a language. The analogy is attractive not only due to the availability of mathematical linguistic tools but also due to the hierarchical nature of documents themselves. Syntactic logical labeling methods can be grouped into two broad categories. The first category of methods [23, 27, 33] is those that achieve logical labeling by building the parse tree for a page according to a specified grammar. The second category of methods [1, 4, 25] uses natural language processing on the output of an OCR system to guide the labeling process.

A representative method of the first category was presented by Nagy et al. [27]. They presented the X-Y tree data structure that transforms a 2D image analysis problem into a hierarchy of 1D string matching problems. Using conventional syntactic formulation, parsing a string effectively segments it into sub-strings that specify both the partitioning of the corresponding block on the page and the logical label of each partition. Hence, both segmentation and labeling of the page take place at the same time.

Abdel Belaïd [4] presented a logical labeling method based on part-of-speech (POS) tagging. The main idea behind this approach is that title and authors of an article could be identified using their specific linguistic characteristics (like author list contains several nouns corresponding to person names). The method parses the ASCII text output of an OCR system through a linguistic analysis tool to assign labels (like proper noun, common noun, adjective, article and number) to each word. Then, several rules are applied on the labeled strings to extract author and title sub-strings.

## Perception-Based Methods

Despite the large variety of layouts and formatting conventions, most humans can easily get the logical information embedded in a page. This has inspired many researchers to use concepts from human perception and cognitive psychology for

logical labeling [15, 24, 28]. The most recent work in this direction is by Rangoni et al. [28]. They employ a perceptive neural network – which has been developed to be an analogy to human perception – for logical labeling. The main idea of perceptive neural networks is to integrate knowledge as interpretable concept associated to each neuron resulting in an architecture with local representation. Both physical and logical structures are incorporated as concepts in the neurons. A training step allows learning the relationships between the two structures from samples. The recognition is not only a classic feed-forward propagation but also performs many perceptive cycles. A perceptive cycle consists of forwarding the physical features, getting the logical output, and, if an ambiguity occurs, correcting the input vector. Due to this perceptive cycle, the system can refine the recognition progressively. Additionally, they incorporated a time-delay neural network to take into account the results of the previous perceptive cycles.

## **Learning-Based Methods**

Learning-based methods make use of raw physical data to analyze the document. For that purpose, no knowledge or static rules are given. The underlying idea is to let the system learn the labeling function by itself and stop relying on rules and heuristics of an expert. A large number of learning-based approaches for logical labeling have been proposed in the literature [7, 9, 11, 12, 26, 32, 43]. While these approaches employ a broad spectrum of learning techniques and use different ways of using these learning methods for logical labeling, the underlying concept of data-driven methodology remains the same. The Biblio system by Staelin et al. [32] is a good example of learning-based methods. It uses example-based machine learning to adapt custom-defined document and metadata types. Like traditional machine learning systems, Biblio has two modes of operation: training and recognition. Both modes of operation use a cascading series of classifiers. During training, many example documents of a single type are fed to the system. The system uses both textual and layout features. Textual features are based on metadata dictionaries that are used to represent the words associated with a given metadata type. The function of these dictionaries is to give the probability that the given text stream contains text representing a particular type of metadata. This is achieved by training a support vector machine (SVM). The SVM engine uses a dictionary that contains a unique ID for every word it encounters. Each time a new word is encountered, a unique ID is generated, and the word is permanently stored in the dictionary. During training, the SVM engine builds input vectors using IDs from the dictionary to identify words in the document. This allows the SVM to create support vectors that identify the most probable words associated with a particular type of metadata. Other layout and textual features (like bounding boxes of page elements, font size, and ascender/descender ratios) are directly extracted from the document. These features along with the probabilities generated by the SVM for each meta-data type are used to train multiple neural networks. Each neural network outputs the probability for each metadata type directly. A majority voting is then used to obtain the final logical labels.

## Knowledge-Based Systems

Generic logical structure information of a document is encoded as a set of layout and typesetting conventions for that document class. These conventions can be regarded as document knowledge for that particular class of documents. Many researchers have taken knowledge-driven approaches [1, 2, 7–9, 17] to reversely engineer the document authoring process and obtain the desired logical structure. Usually a basic distinction of document knowledge into two classes is adopted: Class-Independent Domain Knowledge (CIDK) and Class-Dependent Domain Knowledge (CDDK). The CIDK describes similar characteristics of logical objects in different documents in one domain, regardless of the classes. A domain of documents is defined in this context as a group of documents that can be clustered in terms of the subject. For instance, technical journals, tax forms, business letters, and invoices represent different domains of documents. Hence, documents in one domain can be characterized by their logical similarities, and these similarities are encoded in the CIDK. The CDDK, on the other hand, describes the characteristics of the logical objects of a particular class of documents (like articles from a specific journal). A system will be more effective in document understanding, when it uses CDDK for the target document. However, the system will not be able to handle documents coming from different classes. Hence, robust systems able to process a broad class of documents first use CIDK and then refine the results further by applying CDDK.

Knowledge-based approaches for document understanding can be further divided into two groups: those that employ machine learning to use document knowledge in the recognition phase (like [1]) and those that extract rules from the document knowledge to guide the recognition process (e.g., [7]). Aiello et al. [1] extract a set of geometric and textual features of each document object and its relation to other document objects. These features come from commonsense reasoning and statistical methods and constitute the knowledge base. Based on these features, a decision tree learner is trained to assign one of the logical labels to a document object. Cesarin et al. [7], on the other hand, use a learning scheme to construct the knowledge base. However, application of the knowledge base for document understanding is done through a rule-based system.

## Case-Based Reasoning

Case-based reasoning (CBR) systems work by solving new problems based on the solutions of similar past problems. This concept was used by van Beusekom et al. [36] for logical labeling of title pages of journal articles. Their method takes a set of labeled document layouts and a single unlabeled document layout as input and finds the best matching layout in the set. Structural layout similarity and textural similarity on the block level are used to establish a similarity measure between layouts. Once a best matching layout is found, correspondence is established between the text block in the new document and those in the labeled document retrieved from the training set. Based on this correspondence, labels are simply transferred to the blocks in the new document.

## Application Areas

Logical layout analysis is a key component in document understanding solutions for a large number of domains. A summary of different target domains where logical layout analysis serves a major role in the digitization workflow and the corresponding state-of-the-art approaches in those domains is given in Table 6.1.

Demands from a logical layout analysis system vary a lot from one application domain to the other. This is not only because of different set of desired logical labels to be extracted from the different domains but also because the kind of features that can be used to extract logical labels varies across these domains. In this section, the perspective of a target domain will be taken. In doing so, typical expectations from a logical layout analysis system for that domain will be described, challenges faced in reaching those expectations will be highlighted, and some prominent approaches tackle those challenges will be summarized.

## Books

Digital libraries have become an important player in today's information age for bringing the library to the user. Besides the efforts of several individual libraries to bring their content online and worldwide accessible, last decade has seen a series of mass digitization projects, such as Google Book Search, Million Book Project, and the efforts of Open Content Alliance. The trigger for these activities was the initiative by Google Inc. in its ambitious launch of the Book Search project. They aimed at digitizing all the books in the world and making them accessible online.

The initial steps in most digital library projects involve efficient capture of book pages and converting the captured book pages into searchable text by using commercial OCR software. After that, the next step is to organize the logical units (e.g., chapters in a book or articles in a journal) of a book into a structured representation to facilitate further information retrieval. The goal is to improve the user experience in search and browsing through the book. To enable these tasks, the logical layout analysis module aims at two major tasks:

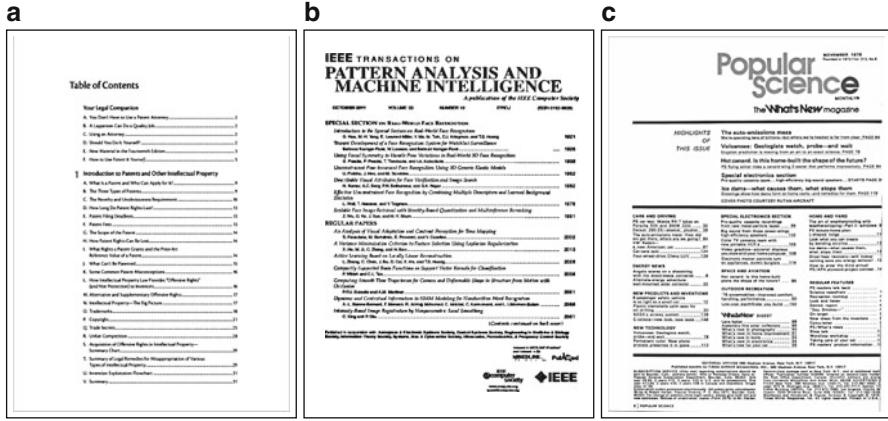
1. Identifying the start of each chapter and sections inside a chapter
2. Detecting table of contents (ToC) pages, recognizing individual entities in ToC pages, and linking them to the actual chapter/section headings

Several approaches have been presented in literature for solving these tasks [4, 8, 13, 25, 33]. In practice, the two problems are usually not solved independently but are tackled together since ToC provides rich information about possible chapter/section headings, which can be used to improve the accuracy of chapter identification step.

Although identifying a ToC page seems trivial at the first glance due to more or less regular structure, yet everyday documents show a variety of ToC pages (see Fig. 6.11). There are different ways to display the organization to the reader (font size, indentation, capitalization, dot leader, item marks, etc.), different numbering systems, different levels of information present, different order of the

**Table 6.1** A summary of state-of-the-art logical layout analysis approaches for different target domains

Application	Technique	Rules	Perception	Learning	Syntactic	Knowledge
Books	[25]				[4, 33]	[8]
Business letters	[22, 35]			[9, 11, 12]		
Invoices				[26]		[7, 20]
Periodicals	[22, 34, 35]	[15, 28]		[3, 32, 43]	[23, 27]	[1, 2, 17]
Other applications	[38]	[24]		[26]		



**Fig. 6.11** Different samples of table of contents (ToC) pages ranging from simple (book) to quite complex (magazine) layouts (a) Book. (b) Journal. (c) Magazine

main fields like author, title, or page number (every field can appear on the left, on the right, or in the middle), and so on. Furthermore, a ToC page can contain other miscellaneous information besides article references, like copyright notice or subscription information. However, since ToC pages serve a particular function (referring to other pages in the same document), they do exhibit some functional regularity. Déjean and Meunier [8] define a ToC as an element having the following properties:

1. Contiguity: A ToC consists of a series of contiguous references to some other parts of the same document.
2. Textual similarity: Each reference has a high textual similarity with the referred part.
3. Ordering: The references appear in the same order in the document as the referred parts.
4. Optional elements: A ToC may include elements whose role is not to refer to any other part of the document, e.g., decorative text or logical headlines grouping ToC elements themselves.

##### 5. No self-reference: All references in a ToC page refer outside the contiguous list of references forming the ToC.

It is important to note here that when resolving references from ToC entries to actual articles in a document, distinction should be made between logical page numbers and physical page numbers. Logical page numbers are the page numbers printed on the ToC pages, whereas physical page numbers are the result of the scanning process. For instance, if all the pages in a book are scanned, the front cover page will be the first physical page. Since the ToC entries refer to logical page numbers, one has to map these to physical page numbers to enable navigation to the correct target page.

A comparative overview of different ToC analysis methods is given in Table 6.2. Among pioneering works for ToC analysis was the RightPages image-based electronic library system [33] developed at AT&T Bell Labs. The system provided services to alert its users to the arrival of new journal articles matching their interest and to let them browse through the alerted article electronically. To enable navigation through different articles, ToC of the corresponding journal was analyzed, and its entries were linked to page numbers of the respective articles. This was done by specifying a grammar for each of the different ToC formats encountered in the indexed journals. The grammar used information about the relative positions of the blocks to determine their semantic structure. The specification was converted into a parse tree that was used to determine the structure of each new ToC page using a grammar specific to the particular journal title. For instance, association was made between the title of an article and its page number. This association was used to determine the page number automatically when a user clicked on the title of an article and then to retrieve and display the appropriate page.

Abdel Belaïd [4] presented a labeling approach for automatic recognition of ToC entries for a digital library system called Calliope. The main objective of Calliope is to index ToC pages from many periodicals and to allow scanning of the corresponding articles on demand. Hence, for automatic recognition of ToC pages, one needs to identify individual entries in a ToC page and further decompose these entries into the corresponding constituents (title, authors, and page number). The method developed by Belaïd is based on part-of-speech (POS) tagging of the OCR result of ToC pages. The tagging process consists of three stages: tokenization, morphological analysis, and syntactic grouping and disambiguation. The tokenizer isolates each textual term and separates numerical chains from alphabetical terms. The morphological analyzer contains a transducer lexicon that produces all legitimate tags (like adjective, article, preposition, common noun, proper noun) for words appearing in the lexicon. The syntactic grouping and disambiguation module employs another transducer which assigns a token to “author” or “title” by examining its context. As a result, each article reference is decomposed into its title, authors, and page number fields.

Lin and Xiong [25] presented TOCDAS (ToC Detection and Analysis System) to detect and analyze ToC pages in a wide range of documents without explicit modeling or training. The goal was to convert MIT Press’ 3,000 out-of-print books, journals, and magazines from paper to high-quality electronic version enabling new services such as online reading and print on demand. A commercial OCR system SDK was used to extract bounding boxes for words, text lines, and non-text regions

**Table 6.2** An overview of key approaches for book structure extraction

Reference	Input features	Target domain(s)	Logical labels	Key idea	Evaluation metric	Experimental dataset
Story et al. [33]	Page segments, OCR results	Books (journals)	Article segmentation and linking on ToC pages	A grammar for each type of ToC page is specified; logical structure is given by the parse tree of a page	Not specified	Private
Belaid [4]	OCR result	Book	Article segmentation and functional tagging in ToC pages	parts-of-speech tagging	Labeling accuracy	Private
Lin and Xiong [25]	Page segments, OCR results	Books	Table of content pages, article segmentation, linking, and functional tagging	Textual similarity and parts-of-speech tagging	Labeling accuracy	Private
Dejean and Meunier [8]	Page segments, OCR results	Books	Table of content pages, article segmentation, and linking	Knowledge-based approach exploiting link structure of ToC pages; logistic regression classifier for decision making	Precision, recall	Private

in addition to plain ASCII text output. The unique feature of their approach is its content association algorithm, through which ToC detection and a significant part of ToC analysis are conducted simultaneously, helping to verify each other. The contents on ToC pages are associated with those on the body pages using observations that text information about an article (like its title) and its start page number will be listed both on the article title page and the ToC page. Due to the distinct nature of these two kinds of information, two algorithms, general text mining and page number matching, are designed to handle them separately. A confidence score is then calculated for the candidate ToC page by adding together the general text mining score and page number matching score. A candidate page is marked as a real ToC page if its confidence score is above a given threshold. Since the identification of a ToC page is based on its association with the contents of other parts of the document, one automatically gets the references from different entries of ToC page to their corresponding articles.

Déjean and Meunier [8] present a method for automatically identifying ToC pages in a scanned book, recognizing different entries in a ToC page, and finally linking the recognized entries to the target chapters/sections. The method aims at developing a generic method to be applied on any document, without any specific knowledge about the document or the collection it may belong to. Besides, the same method can be used to detect other organizational tables (e.g., table of figures) in the document. The key idea of their approach is to first use functional knowledge to identify ToC pages and then to apply formal or layout knowledge to improve the accuracy of the method in a second step. The method computes pairwise textual similarity between all text blocks. Then, a functional definition of a ToC page based on contiguity, textual similarity, ordering, presence of optional elements, but ignore self-references, is used to generate and score ToC candidates. A ToC candidate is a set of contiguous text blocks, from which it is possible to select one link per block to provide an ascending order for the target text blocks. Finally, a Viterbi best-path algorithm is applied to select the best candidate ToC page. Moreover, its references are identified. The results of this step are further refined by training a binary classifier to determine for each link whether it belongs to the ToC or not. The classifier uses formal knowledge about the document itself by learning a feature vector consisting of layout and textual features. This results in a refined ToC that can be used to structure a document.

To evaluate and compare automatic techniques for deriving structure information from digitized books, a book structure extraction competition was organized at ICDAR 2009 [13]. The task that participants faced was to construct hyperlinked tables of contents for a collection of 1,000 digitized books selected from the INEX book corpus [19]. Two hundred out of those one thousand books did not contain a printed ToC. Four organizations participated in the contest, and their submissions were evaluated using commonly used precision, recall, and F-measure metrics. The results showed that the method from Microsoft Development Center Serbia achieved the best results with an F-measure of 41.5 %. Evaluation on a separate subset of books without an explicit ToC page showed that existing algorithms for this task are still premature, with the winning algorithm achieving an F-measure of 7 % only.

## Invoices

Processing of invoices, either automatically or manually, is a daily task in most of the companies. The volume of invoices to be processed largely depends on the type of business of a company. For instance, insurance companies receive a large number of reimbursement claims every day. These invoices contain similar information, but the information items are distributed according to a large number of different layout styles. The information items are usually categorized into two broad classes:

1. Header data: data about the invoicing party, invoice date, invoice number code, total invoice amount, due date for payment, etc.
2. Table or position data: details about the invoice elements line by line

Automatic extraction of these information items from a given invoice is the goal of invoice reading systems. Due to the presence of table data, table spotting and understanding play a major role in automatic invoice processing. This section focuses on techniques for extraction of head data. The readers are referred to section “[Table Detection and Labeling](#)” for a discussion about table structure extraction and analysis.

When processing invoices, several logical labels need to be extracted from the invoice head data. Since financial transactions are based on invoices directly, accurate extraction of these labels is very critical. Therefore, systems for invoice processing use as much knowledge as possible to verify the extracted information from the invoices. Medical invoices can be considered as a use case here, without loss of generality. A medical insurance company may receive thousands of invoices each day. When processing a particular invoice, the first logical label that is required is the identity of the patient (first name, last name, insurance number). Then, the date of the treatment needs to be extracted to verify whether the person was insured at the time of treatment. After that, the treatment details (table data) have to be extracted to do plausibility checks (are calculations correct, does the reference number look genuine, is treatment cost in line with similar treatments of other patients, does treatment suit the diagnosis, does the patient’s insurance policy cover this treatment, etc.). Finally, the total amount to be paid and payment target should be identified. If payment is to be made to a doctor/agency, the bank account details of the doctor/agency should also be extracted (and verified).

It is important to note here that all of the abovementioned information fields to be extracted from an invoice have a vital importance to ensure correct transaction to the recipient and to prevent insurance fraud. To reliably extract information, these fields are usually distinguished into different classes based on how they can be verified [21]:

1. Enumerable fields: These are the fields for which the extracted value can be judged as correctly recognized by matching it to a database having all possible values for that field. For instance, a person’s identification can be regarded as correctly extracted if the identified first name, last name, and date of birth match with the company database.
2. Labeled fields: These are the fields that are most often simply identified by a keyword present in the field, like “From:...,” “Total,” and “Account number.”

One of the earliest invoice processing systems was presented by Cesarini et al. [6] (for a comparative overview, please see Table 6.3). The system was designed to handle only form-like invoices and was tested on a dataset consisting of utility bills from Italian TELECOM and Italian electricity supplier company ENEL. Since the system was limited to form-like invoices, they used a three-step approach. In the first step, form registration was performed to align the target form with a reference form (prototype). Then, information fields were located by establishing correspondences between the marked information fields in the prototype with the fields in the target form. Finally, the identified information fields were recognized by using a connectionist-based model, using a multilayer perception as an autoassociator.

To analyze multi-class invoices, Cesarini et al. [7] developed a knowledge-based system. Two levels of knowledge for the invoice domain were elaborated. Logical similarities of invoices were captured by general knowledge about the domain (CIDK) as well as class-specific knowledge (CDDK). CIDK describes similar characteristics of logical objects in different invoices, regardless of the classes, whereas CDDK represents a collection of the characteristics of logical objects of particular classes of invoices. Both these knowledge sources are constructed by a learning procedure on a learning set consisting of a small number of invoices of each class. When an invoice has to be analyzed, both knowledge sources are used (for details, please see section “[Knowledge Based Systems](#)”).

A commercial system called smartFIX was presented in [20] for analyzing invoices from a variety of sources. The system implements a comprehensive document understanding pipeline, from scanning and document classification to information extraction and verification. Since the system is deployed in several health insurance companies in Germany, it has a thorough coverage of logical labels to support different workflows in the companies. A total of 107 labels are supported, although on average, about 20 appear on a single document.

Medvet et al. [26] present a probabilistic approach for logical labeling of invoices. Their approach is model based, where a model represents documents of the same class, i.e., invoices from the same firm. When an invoice from a new firm is received, the operator can make the corresponding model using a GUI. Given a new document from the same firm, the logical labels are identified as the sequence of blocks that are most probable given the model.

## **Business Letters**

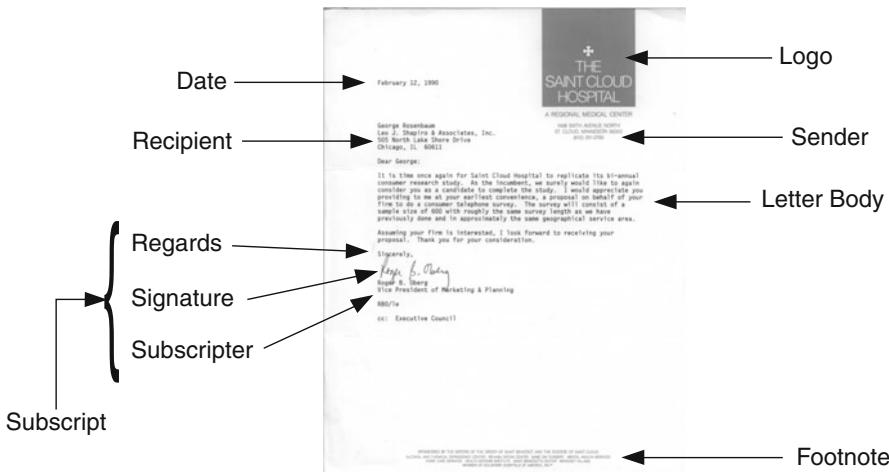
To cater the needs of modern offices, many companies convert incoming paper documents into a structured electronic representation that allows better information management and distribution. Business letters constitute a major part of these documents. Therefore, logical labeling of business letters has been a topic of research since early days of document analysis (see Table 6.4). Automatic analysis and understanding of business letters involve identification of key components representing certain concepts in the letter, like sender, receiver and date. Fortunately, business letters typically follow some specific layout structures, and hence the

**Table 6.3** An overview of key approaches for invoice analysis

Reference	Input features	Target domain(s)	Logical labels	Key idea	Evaluation metric	Experimental dataset
Cesarini et al. [6]	Raw image	Invoice	Data fields in form-like invoices	Registering new invoices to manual prototypes using attributed relational graphs	Labeling accuracy	INFORMys
Cesarini et al. [7]	Page segments, geometric layout tree, OCR result	Invoices	Invoice number, total amount	Knowledge-based approach using rules based on layout features and contents	Success rate, labeling, accuracy, label segmentation accuracy	Private
Klein and Dengel [20]	Page segments, OCR result	Invoices	107 labels appearing in medical invoices	Document management system specifically aimed at commercial-grade invoice processing	Classification accuracy	Private
Medvet et al. [26]	Text blocks and their OCR result	Invoices, patents, datasheets	Eight labels in invoices, 11 labels in patents, 8 labels in datasheets	Probabilistic modeling of known layouts, user interaction to model a new layout	Success rate	Ghega

**Table 6.4** An overview of key approaches for business letter understanding

Reference	Input features	Target domain(s)	Logical labels	Key idea	Evaluation metric	Experimental dataset
Dengel [9]	Page segments, geometric layout tree	Business letters	Sender, recipient, date, subject	Geometric trees representing different layouts are learned, labeling done using a decision tree	Not specified	Private
Dengel et al. [12]	Page segments, geometric layout tree	Business letters	Sender, recipient, date, subject	Logical object arrangements are learned using layout trees, labeling done using a probabilistic parsing of the tree	Not specified	Private
Tsujimoto and Asada [35]	Geometric layout tree	Business letters, technical journals, magazines, newspapers	Title, abstract, subtitle, paragraph, header, footer, page number, caption	Transform geometric layout tree to logical tree using a set of rules	Not specified	Private
Dengel and Dubiel [11]	Page segments, geometric layout tree	Business letters,	Sender, recipient, date, logo, subject, footer, body text	Unsupervised learning to build a geometric tree based on attribute of logical objects	Precision, recall	Private
Klink and Kieninger [22]	Page segments, OCR result	Business letters, technical journals	Business letters (header, addressee, subject, date, sender, closure) Journals (header, title, author, abstract, bibliography)	Human expert defines rules, which are then matched with a fuzzy combination to label new documents	Precision, recall	UW-I Private



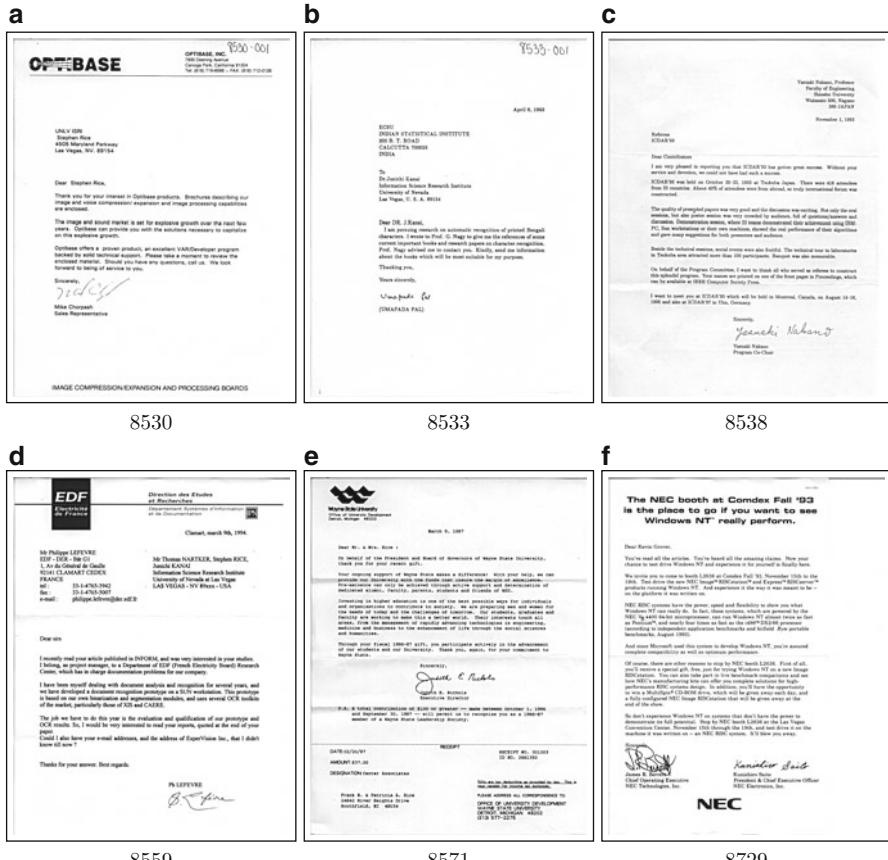
**Fig. 6.12** Typical logical components of a business letter

layout of a letter already provides rich information about the logical components of the letter. For instance, when one looks at the block arrangement in Fig. 6.4, it is easy to deduce where the recipient and date are located.

When specifying the logical components to be extracted from a business letter, one also should take care of the granularity with which these labels should be extracted. A typical business letter and its different logical components are shown in Fig. 6.12. It is important to note that the logical structure divides the contents of the letter into a hierarchy of logical objects. In the example in Fig. 6.12, only top-level logical objects are shown, except the *subscript* that is further divided into *subscriber*, *signature*, and *regards*. Other logical components could also be further subdivided, for instance, the *recipient* could be decomposed into *recipient name* and *recipient address*, whereas *recipient name* could further be split into *first name* and *last name*.

Although business letters have a lot of structural similarity, in practice one still encounters a wide variety of relative locations of different logical components. Furthermore, not all components might be present in a particular letter. Some sample letters highlighting the diversity in layouts of letters from the UNLV database are shown in Fig. 6.13.

One of the earliest works in the domain of business letter understanding was the ANASTASIL (Analysis System to Interpret Areas in Single-sided Letters) system [9, 10] by Dengel. The system identified important logical objects within business letters, like recipient, sender, or company-specific printings. The system utilized only geometric knowledge sources and worked completely independent of text recognition. The sources included global geometric knowledge about logical object arrangements, and local geometric knowledge about formal features of logical objects (e.g., extensions, typical font sizes). To model knowledge about more



**Fig. 6.13** Different samples from the UNLV database illustrating diversity in letter formats: **(a)** Date is missing. **(b)** No logo, header, or footer present. **(c)** No recipient name or address specified. **(d)** Sender details are on the *left side*, whereas recipient details are on the *right side*. **(e)** Sender and recipient details are provided at the *bottom* of the letter. **(f)** Multiple subscript blocks present

than one document layout and to obtain a compact knowledge representation, they used a geometric tree. The geometric tree described a global hierarchy of possible logical object arrangements. The tree was constructed using training data. When identifying logical labels of a given page image, the geometric tree is used as a decision tree.

While the ANASTASIL system focused on locating logical objects in a document image, its successor system ΠΟΔΑ [12] also extracted text from the identified logical objects. Since logical labeling provided the identification of document constituents, the subsequent text recognition and verification were performed in an expectation-driven manner. For controlled selection of vocabularies, syntactic knowledge was attached to some of the generic logical classes. This knowledge

determined the order in which subordinate logical objects (e.g., ZIP code, city, and street in an address block) can occur and, therefore, controlled the access to corresponding logical vocabularies.

Dengel and Dubiel [11] described a system (DAVOS) capable of both learning and extracting document logical structure. DAVOS learned document structure concepts by detecting distinct attribute values in document objects. The structural concepts were represented by relation patterns defined by a cut-and-label language. A geometric tree was used to represent the concept language. Unsupervised decision tree-based learning techniques were used to build the tree. Forty letters were used to train the system, and then the learned geometric trees were used to classify another set of 40 unknown letters.

Klink and Kieninger [22] presented a hybrid approach for document structure understanding. They make use of layout (geometric) as well as textual features of a given document. Based on these features, rules are formulated to indicate how one might observe a specific layout object in a document. Besides, the rules also express dependencies between different layout objects. For each block in a document image, it is checked which rules in the rule base are satisfied. This matching of rules is done on a fuzzy basis. Finally, different matched rules are combined into a single probability score for the logical label of that block.

## Technical Journals, Magazines, and Newspapers

Periodicals constitute a major portion of printed material, whether they be technical journals, newspapers, or magazines on a wide variety of subjects. Digitization of such periodicals has been a major driving force in document analysis research (see Table 6.5), since it is desirable to have centralized databases that index articles within a field. For instance, in the case of technical journals, several indexing projects exist to give a comprehensive overview of research in particular fields. A prominent example is MEDLINE, the flagship bibliographic citation database of the US National Library of Medicine that indexes more than 11 million articles from over 4,800 indexed titles in the fields of medicine and health sciences.

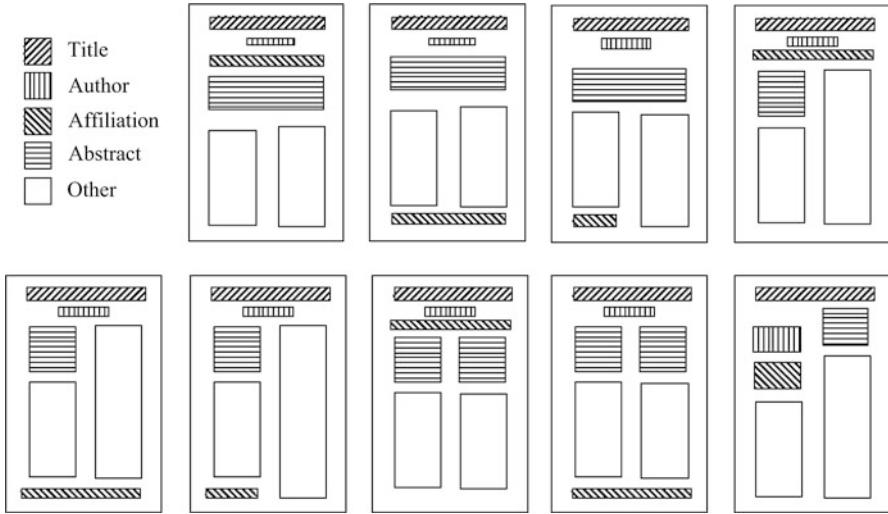
To facilitate automatic data entry of bibliographic information, one needs to identify titles and authors of different articles. Furthermore, automatic detection of headlines is desirable to allow navigation within an article. A major challenge in automatic bibliographic metadata extraction projects is to deal with a large variety of layouts that journals or magazines follow. Besides, articles within a particular journal may also show significant variations in position of different logical blocks depending on their length or presence of other optional logical elements. An example illustrating these interclass variations in layouts is shown in Fig. 6.14.

Pioneering work in the field of understanding technical journal articles were by Nagy et al. [23, 27] and Tsujimoto et al. [35]. The approach by Nagy et al. relied on page grammars to decompose a page image into its different logical components with a recursive X-Y cut method. Their method transforms a two-dimensional segmentation and labeling problem into a one-dimensional segmentation and labeling

**Table 6.5** An overview of key approaches for analysis of articles from technical journals, magazines, and newspapers

Reference	Input features	Target domain(s)	Logical labels	Key idea	Evaluation metric	Experimental dataset
Tsujimoto and Asada [35]	Geometric layout tree	Business letters, technical journals, magazines, newspapers	Title, abstract, subtitle, paragraph, header, footer, page number, caption	Transform geometric layout tree to logical tree using a set of rules	Not specified	Private
Nagy et al. [27]	Raw image	Technical journals	Title, abstract, page number, keywords, section titles, captions, etc.	Syntactic segmentation and labeling of pages using block grammars defined for each layout type	Success rate	Private
Altanura et al. [2]	Page segments, geometric layout tree	Technical journals	Title, author, abstract, body text, etc.	Machine learning to extract rules for logical labeling	Labeling accuracy	Private
Klink and Kieninger [22]	Page segments, OCR result	Business letters, technical journals	Business letters (addressee, subject, date, sender, closure) Journals (header, title, author, abstract, bibliography)	Human expert defines rules, which are then matched with a fuzzy combination to label new documents	Precision, recall	UW-I Private
Aiello et al. [1]	Geometric features, content features, neighborhood relationships	Technical journals, magazines, newspapers	Title, body, caption, page number	A knowledge base is created based on input features. A decision tree learner is used to assign logical labels	Precision, recall	UW-II MTDB

Tan and Liu [34]	Raw image	Newspapers	Headlines	Rules applied to smeared image	Precision, recall	Private
Eglin and Bres [15]	Page segments	Technical journals, magazines, newspapers	Title, sub-title, paragraph, footnote, caption	Distinguish labels of page segments based on their visual saliency	Labeling accuracy	MTDB
van Beusekom et al. [37]	Page segments	Technical journals	Title, author, abstract, affiliation	Case-based reasoning method: labeling based on correspondence of blocks with most similar layout	Labeling accuracy	UW-II MARG
Staelin et al. [32]	Page segments, OCR results	Journal articles, grant deeds	Journal articles (title, author, pages, date, etc.), Grant deeds (deed number, county, grantor, grantee, etc.)	A hierarchy of self-tuning classifiers (neural networks and SVMs)	Precision, recall	Private
Zou et al. [43]	OCR result, HTML tags	Technical journal	Bibliographic references locating and parsing	Machine learning to first identify and then parse references	Labeling accuracy	Private
Rangoni et al. [28]	Page segments, OCR results	Technical journals	21 logical labels (title, author, abstract, page number, etc.)	Time-delay neural network analogous to human perception	Labeling accuracy	MARG



**Fig. 6.14** Variations in layout of different journals with respect to position of various logical entities in the MARG dataset

problem without distinguishing between physical layout and logical structure. Tsujimoto et al. [35] represented document physical layout and logical structure as trees and tackled document understanding as the transformation of a physical tree into a logical one using a set of generic transformation rules.

Altamura et al. [2, 17] presented a structured document understanding system called WISDOM++. The presented system implements the complete processing pipeline including preprocessing, page segmentation, block classification, document classification, and logical labeling. The logical labeling module associates page layout components with basic logical components. The mapping is achieved by matching the document description against models of classes of documents and models of logical components interesting for that class. Both layout structures and models are described in a first-order language, where unary and binary function symbols, called attributes and relations, are used to describe properties of single layout components and their relative position. A first-order learning system is applied for the induction of rules used for the layout-based classification and understanding of documents. A free version of WISDOM++ software package is available at <http://www.di.uniba.it/~malerba/wisdom++/>.

Aiello et al. [1] presented a knowledge-based system for logical structure extraction from a broad class of documents. To keep the system generic enough, they considered extraction of only title, body, caption, and page number as logical entities. The contents of a document were described using textual and geometric features. The content features of a document object (block) consist of its aspect ratio, area ratio, font size ratio, font style, number of characters, and number of lines. The geometric features describe global arrangement of objects on the page as well as

spatial relation between objects. For global arrangement, neighborhood relations are used (two objects are considered neighbors if they share an edge in the Voronoi diagram of the centers of the document objects). Spatial relations between objects are described as thick boundary rectangle relations on both axes. These relations include precedes, meets, overlaps, starts, during, finishes, equals, and their inverse. A feature vector consisting of all of these features is used to train a C4.5 decision tree classifier to distinguish the desired logical objects (title, body, caption, page number).

Identification of title and author areas in a technical document image using a Delaunay triangulation-based method was presented in [41]. First, the positions and alignment of small text line regions are measured by different triangle groups. Then, character stroke widths are calculated from the constrained Delaunay triangulation. Finally, the rules defining spatial features and font attributes of the title and author region are applied to single line text regions to extract title and author regions.

A method for assigning functional labels to pre-segmented homogeneous page segments using a psycho-visual approach was presented by Eglin and Bres [15]. Their method classifies text blocks into headings, body paragraphs, and highlighted text according to their visual saliency and perceptual attraction power on the reader's eye. The main idea is to characterize text blocks using their complexity, visibility, and geometric characteristics. The complexity captures the frequency of transitions between text components, whereas the visibility estimates the density of these transitions. The geometric features correspond to the dimensions and location of each text block on the page. Finally, a decision tree is trained on these features to perform functional labeling.

A system for extracting headlines from newspaper microfilms was presented by Tan and Liu [34]. The goal was to provide automatic indexing of news articles by extracting headlines from digitized microfilm images to serve as news indices. Since the aim was to extract only prominent headlines, a method based on the run-length smearing algorithm (RLSA) [40] was proposed to extract headlines without the need for detailed layout analysis. The method was based on computing statistics of blocks returned by RLSA algorithm to first distinguish text blocks from non-text blocks and then to differentiate a headline from other text blocks.

An example-based machine learning system to adapt to customer-defined document and metadata type, called Biblio, was presented in [32]. One key element of Biblio's design was that it did not require a skilled operator to tune the underlying machine learning algorithm. For training the system, many example documents of a single type are fed to the system. Each document type has its own set of neural networks and support vector machines for detailed processing. SVMs are used to create metadata dictionaries that represent the words associated with a given metadata type. These dictionaries are used by neural networks while looking for evidence of data specific to a document type. Multiple neural networks are trained and combined in the form of committees to improve the overall accuracy of the system.

Rangoni et al. [28] have presented a dynamic perceptive neural network model for labeling logical structures in technical documents. Their system combines

capabilities of model-driven and data-driven approaches. The main idea is to integrate knowledge as interpretable concepts in a perceptive (also called transparent) neural network. The system performs several recognition cycles and corrections while keeping track of and reusing the previous outputs. This dynamic behavior is achieved by using a time-delay neural network. Using several cycles, their system is able to make corrections to the input when ambiguous inputs are encountered.

## Other Application Areas

A large diversity of documents exists around us that is not limited to one of the abovementioned categories (see Table 6.6). Different researchers have developed specialized techniques for understanding non-stereotypical documents. In the following, a few prominent contributions for different categories of documents are outlined.

Lemaitre et al. [24] presented a method inspired by perceptive vision phenomena for naturalization decree register treatment. The analysis of naturalization decree registers aimed at extracting register numbers and surnames for each act contained in the document. The method was applied on more than 15,000 registers from the French national archives, dated between 1883 and 1930, representing about 85,000 pages. To be robust against noise and poor quality of the documents, the authors used a multi-resolution approach. Different resolutions were built successively by low-pass filtering the initial image. Then, for each chosen resolution, suitable features were extracted. Since the low-resolution version contains only a global perspective of the document, initial detection of surname and register number regions is done at that level. This detection is then further refined at higher resolution to detect the actual locations of the desired logical objects.

Wang et al. [38] proposed a specialized method to analyze tender documents. In a tender, there is no indexing information such as a table of contents (ToC) to indicate different logical units of the document. Its hierarchical logical structure is embodied in citing relationships between its different pages. A tender document can be divided into several logical parts, called bids. Every bid can be divided into either sub-bids or basic pages, and every sub-bid is also composed of basic pages. The main part of every page is a form containing several items. An item is the basic unit that can be priced in a tender document. At the end of every part, such as a sub-bid, bid, and the whole document, there exist several pages containing the summary information of this part. The logical structure of the tender includes not only the structure within a single page but also the relationship between different pages (e.g., a bill page may refer to several sub-bill pages). Hence, instead of only a sequential order, a citing order also exists between pages. This means that one page is not only before or after another page, but it also may be cited by or cite other pages. Since the citing information is essential for generating the hierarchy of the tender, the extraction of the logical structure of the tender is therefore more difficult than that of a common document. The system developed by Wang et al. called VHTender consists of three steps. First, preprocessing is done to correct skew angle of the

**Table 6.6** An overview of document understanding methods for certain niche domains

Reference	Input features	Target domain(s)	Logical labels	Key idea	Evaluation metric	Experimental dataset
Wang et al. [38]	Page segments, geometric layout tree, OCR result	Tender documents	Extraction of citing relationships between pages to reconstruct hierarchy of a tender	A knowledge base encodes domain-specific rules. A rule learning method is applied to learn and then extract logical labels	Labelling accuracy (percentage of correctly assigned labels)	Private
Staelin et al. [32]	Page segments, OCR results	Journal articles, grant deeds	Journal articles (title, author, pages, date, etc.) Grant deeds (document number, county, grantor, grantee, etc.)	A hierarchy of self-tuning classifiers (neural networks and SVMs)	Precision, recall	Private
Lemaître et al. [24]	Raw image	Naturalization decree	Serial number, candidate name	Multi-resolution image analysis mimicking human perception	Precision, recall	Private
Medvet et al. [26]	Text blocks and their OCR result	Invoices, patents, datasheets	Eight labels in invoices, 11 labels in patents, 8 labels in datasheets	Probabilistic modeling of known layouts, user interaction to model a new layout	Success rate (percentage of documents with no labeling error)	Ghega

scanned pages. Then, the page analyzer extracts the geometric layout of the page. In the last step, the page grouper module synthesizes the layout information of every page into a physical structure tree of the tender and analyzes the citing information to deduce and create the logical structure tree. A knowledge base storing domain-related rules is used to aid both page analyzer and page grouper modules.

Logical labeling of patent applications was investigated by Medvet et al. [26]. Their probabilistic modeling approach was designed for application scenarios in which the set of possible document classes may evolve over time. Patent applications make a suitable test case for this approach since different patent sources have different layouts. The labels that they automatically extracted from patent applications were title, applicant, inventor, representative, filing date, publication date, application number, publication number, priority, classification, and first line of abstract. When a patent application from a new source is to be indexed, the operator can make the corresponding model using a GUI. Later, when a new application from the same source is encountered, the logical labels are identified as the sequence of blocks that are most probable given the model.

---

## Experimental Validation

### Performance Measures

To evaluate a logical labeling algorithm, one needs to take into account the perspective and scope of the target application. When logical labeling is done individually as a module of a rather complex document understanding system, the target of evaluation is to find out the percentage of correctly identified labels. From the perspective of an algorithm/system developer, one is further interested in identifying different classes of errors quantitatively, whereas end users are typically only interested in overall performance of the algorithm (irrespective of the types and number of different kinds of errors). Similarly, in many cases, the results of other modules like page classification or OCR are combined to give an overall error rate of the system.

Consider an automatic system for processing business letters as a use case. When evaluating such a system, one is interested in finding out what percentage of *actual* labels, e.g., insurance number, has been correctly recognized. However, the incoming mail sorting system might not identify a letter correctly and label it as a form. That would increase the error rate of the system despite the fact that logical labeling itself had no chance of fixing this error. Similarly, the presence of OCR errors would also reduce the accuracy of the system. Hence, what is actually being measured, even when using the same metrics, might vary from one publication to the other.

The most general and widely used measures in different domains of logical labeling are precision and recall. These measures are well understood in information retrieval community and applying them to a logical labeling problem is straightforward. The precision and recall are defined for every logical label  $l$  as

$$\text{Precision}_l = \frac{|D_l \cap G_l|}{|D_l|} \quad \text{Recall}_l = \frac{|D_l \cap G_l|}{|G_l|}$$

where  $D_l$  represents the set of document objects classified by the system as logical objects with label  $l$  and  $G_l$  represents the set of document objects with the label  $l$  in the ground truth. The overall precision and recall of the system can be computed as the average precision and recall for all the labels. For a more detailed discussion of these measures, please refer to ►Chap. 30 (Tools and Metrics for Document Analysis Systems Evaluation).

## Datasets

Public ground-truth datasets are crucial for progressing the state of research in many fields of computer science by providing common grounds for comparing different algorithms. The need for such datasets was identified early in the document analysis community, and several ground-truth datasets emerged for different target applications. For logical labeling, there are also several ground-truth datasets available. A brief overview of these datasets is given here. The reader is referred to ►Chap. 29 (Datasets and Annotations for Document Analysis and Recognition) for details.

### **University of Washington Datasets (UW-I/II/III)**

The UW datasets are probably the most widely used datasets in the document analysis community. The UW-III dataset, which is the latest one in the series, consists of 1,600 scanned pages from technical journal articles with an extensive ground truth for each page. These datasets have been used for validation of logical labeling approaches in [1, 22].

### **Oulu MediaTeam Documents Datasets (MTDB)**

The MTDB is a collection of scanned documents and the corresponding ground-truth data about the physical and logical structure of the documents. It consists of 500 document images from a wide diversity of sources (journal articles, newspaper articles, advertisement, correspondences, etc.). Use of this dataset for evaluating logical labeling methods can be found in [1, 15].

### **Medical Articles Records Ground-Truth Datasets (MARG)**

The MARG dataset contains title pages of medical journal articles from various publishers. The title pages have varying layouts. For each title page, ground-truth boxes containing the title, authors, affiliation, and abstract are provided. This dataset was used in [28] for evaluating their methods.

### **INFORMys Invoice Datasets**

This dataset consists of 100 invoices from Italian TELECOM and 182 invoices from Italian electricity company ENEL. It was used in [6] to validate their invoice reading

system and is still available at <http://www.dsi.unifi.it/~simone/Dante/informys/demo/node24.html>.

### Ghega Datasheets and Patents Datasets

This dataset is composed of 110 datasheets of electronic components and 136 patent documents each divided into 10 classes. Datasheets of the same class share the producer and the component type. Multiple logical labels (model, type, case, power dissipation, etc.) are identified for each datasheet. For patent documents, each class represents one patent source. Several logical labels are identified (title, applicant, filing date, publication date, etc.) on each patent document. The dataset was used to demonstrate application of the document understanding approach in [26] on documents from different domains. The dataset is publicly available at <https://sites.google.com/site/ericmedvet/research/ghega-dataset>.

---

### Recommendations

Choosing a logical labeling approach that best fits the needs of a project is not an easy task. One not only has to choose among a large variety of algorithms, but also the basic assumptions underlying different approaches need to be carefully investigated. Besides, choosing the right balance between specificity and generality is also crucial. While it is natural to desire having a general-purpose system, the cost of developing such a system might not be justified in the context of the project. Hence, a very important step is to do a thorough requirement analysis for the logical labeling module of the project. Questions one may ask at this stage are:

- What is the domain of documents that need to be processed? Is the project targeting documents from multiple domains? Is the domain of each document to be processed known *a priori*? Is the domain of documents to be processed static or dynamic?
- How variable is the layout of documents from a specific class? Do the documents always capture all logical components or are some of the elements optional?
- What is the intended role of the logical labeling module? Which set of logical labels are absolutely necessary for executing the project? Which labels can be considered optional?
- What is the scope of the project? Is it a one-time conversion project, or will the system remain operational to handle new documents every day?
- Who are the intended users of the system? Will the system run autonomously without a human operator? Will end users have the skills to adapt the system to new document classes?
- What are the requirements on the operational speed of the system? In other words, what is the volume of the documents to be processed each day by the system?
- How stringent are the accuracy requirements? What are the costs of different kinds of errors made by the system?

If the domain of each document to be processed is known, one can limit the search to specifically tailored methods for that domain (cf. Table 6.1). Otherwise, one of the generic methods (e.g., [1, 22, 26]) could be used. It is interesting to note that each of these methods has their particular strengths. The method by Klink and Kieninger [22] is a rule-based system in which an expert can define rule expressions (one rule for each desired logical label). Such rules could be defined by some domain expert who might not be from a computer science background. Hence, in extreme cases, one could develop such a system based only on the domain knowledge of the expert, without having any training data at hand. Application of these rules using fuzzy logic is also straightforward, and hence the system overall is easy to implement. The approach by Aiello et al. [1] is a knowledge-based approach that uses machine learning to draw inference based on the knowledge base. This approach is quite suitable if one has labeled training data available. The learning algorithm (a decision tree in this case) learns relations from data and automatically builds a model (set of rules) to assign logical labels based on the observed features of the document objects. The method presented by Medvet et al. [26] is designed for scenarios in which the set of target document domains is dynamic and may evolve over time. They provide a graphical user interface (GUI) with which an unskilled operator can train a new class/domain of documents by merely providing a few samples and defining regions of interest with their labels. A probabilistic approach is then used to automatically estimate the parameters of the newly defined class, which can then be used to label new documents of that class.

As a general recommendation, it should be noted that the amount of effort required for setting up a logical labeling system and the achievable error rates are both often underestimated. This is probably due to the ease with which humans can readily extract logical information from a given document. Hence, when designing a logical labeling system, one should not try to solve the problem in a more general way than required. This will not only save development time but also result in more accurate system for the target application.

---

## Conclusion

Logical labeling is an integral part of most of the document analysis systems employed in digitization workflows. Owing to the central role of logical labeling in extracting semantic information from documents, a major effort in the document analysis community has been spent on it for over two decades. Hence, a large variety of methods exist for logical labeling using diverse concepts from artificial intelligence and computer science – not only present in the cited work, but manyfold proposed in the respective conference proceedings of this field. Initial efforts in logical labeling focused on processing documents from one particular domain with a limited variety of layouts. As the field matured, methods capable of handling documents from multiple domains emerged. In the meantime, several publicly available ground-truth datasets were developed. This allowed researcher not only to better perform comparative evaluations but also to benchmark on representative

samples of the target domain. Recent systems for logical labeling [28] report about 98 % accuracy on large public datasets containing documents with diverse layouts.

Despite the impressive advances in the functionality of the state-of-the-art logical labeling methods, their capabilities still remain far behind human skills. The ease and efficiency with which humans can just skim a document (even if the document is in a foreign language) to get its logical structure are yet to be matched by machines. Hence, much room for improvement remains in existing methods, and it is expected that more generic as well as accurate systems will be developed in the near future.

---

## Cross-References

- ▶ [Datasets and Annotations for Document Analysis and Recognition](#)
  - ▶ [Imaging Techniques in Document Analysis Processes](#)
  - ▶ [Page Segmentation Techniques in Document Analysis](#)
  - ▶ [Page Similarity and Classification](#)
  - ▶ [Recognition of Tables and Forms](#)
  - ▶ [Tools and Metrics for Document Analysis Systems Evaluation](#)
- 

## References

1. Aiello M, Monz C, Todoran L, Worring M (2002) Document understanding for a broad class of documents. *Int J Doc Anal Recognit* 5(1):1–16
2. Altamura O, Esposito F, Malerba D (2001) Transforming paper documents into XML format with WISDOM++. *Int J Doc Anal Recognit* 4(1):2–17
3. Bayer T, Franke J, Kressel U, Mandl E, Oberländer M, Schürmann J (1992) Towards the understanding of printed documents. In: Baird H, Bunke H, Yamamoto K (eds) Structured document image analysis. Springer, Berlin/New York, pp 3–35
4. Belaid A (2001) Recognition of table of contents for electronic library consulting. *Int J Doc Anal Recognit* 4(1):35–45
5. Cattoni R, Coianiz T, Messelodi S, Modena CM (1998) Geometric layout analysis techniques for document image understanding: a review. Technical report 9703-09, IRST, Trento, Italy
6. Cesarini F, Gori M, Marinai S, Soda G (1998) INFORMys: a flexible invoice-like form-reader system. *IEEE Trans Pattern Anal Mach Intell* 20(7):730–746
7. Cesarini F, Francesconi E, Gori M, Soda G (2003) Analysis and understanding of multi-class invoices. *Int J Doc Anal Recognit* 6(2):102–114
8. Déjean H, Meunier J (2009) On tables of contents and how to recognize them. *Int J Doc Anal Recognit* 12(1):1–20
9. Dengel A (1992) ANASTASIL: a system for low-level and high-level geometric analysis of printed documents. In: Baird H, Bunke H, Yamamoto K (eds) Structured document image analysis. Springer, Berlin/New York, pp 70–98
10. Dengel A, Barth G (1988) High level document analysis guided by geometric aspects. *Int J Pattern Recognit Artif Intell* 2(4):641–655
11. Dengel A, Dubiel F (1996) Computer understanding of document structure. *Int J Imaging Syst Technol* 7:271–278
12. Dengel A, Bleisinger R, Hoch R, Fein F, Hönes F (1992) From paper to office document standard representation. *IEEE Comput* 25(7):63–67

13. Doucet A, Kazai G, Dresovic B, Uzelac A, Radakovic B, Todic N (2011) Setting up a competition framework for the evaluation of structure extraction from OCR-ed books. *Int J Doc Anal Recognit* 14(1):45–52
14. Duygulu P, Atalay V (2002) A hierarchical representation of form documents for identification and retrieval. *Int J Doc Anal Recognit* 5(1):17–27
15. Egin V, Bres S (2004) Analysis and interpretation of visual saliency for document functional labeling. *Int J Doc Anal Recognit* 7(1):28–43
16. e Silva AC, Jorge AM, Torgo L (2006) Design of an end-to-end method to extract information from tables. *Int J Doc Anal Recognit* 8(2–3):144–171
17. Esposito F, Malerba D, Lisi F (2000) Machine learning for intelligent processing of printed documents. *J Intell Inf Syst* 14(2–3):175–198
18. Fan H, Zhu L, Tang Y (2010) Skew detection in document images based on rectangular active contour. *Int J Doc Anal Recognit* 13(4):261–269
19. Kazai G, Doucet A (2008) Overview of the INEX 2007 book search track (BookSearch'07). *SIGIR Forum* 42(1):2–15
20. Klein B, Dengel A (2003) Problem-adaptable document analysis and understanding for high-volume applications. *Int J Doc Anal Recognit* 6(3):167–180
21. Klein B, Agne S, Dengel A (2006) On benchmarking of invoice analysis systems. In: Proceedings of the international workshop on document analysis systems, Nelson, pp 312–323
22. Klink S, Kieninger T (2001) Rule-based document structure understanding with a fuzzy combination of layout and textual features. *Int J Doc Anal Recognit* 4(1):18–26
23. Krishnamoorthy M, Nagy G, Seth S, Viswanathan M (1993) Syntactic segmentation and labeling of digitized pages from technical journals. *IEEE Trans Pattern Anal Mach Intell* 15(7):737–747
24. Lemaitre A, Camillerapp J, Coüasnon B (2008) Multiresolution cooperation makes easier document structure recognition. *Int J Doc Anal Recognit* 11(2):97–109
25. Lin X, Xiong Y (2006) Detection and analysis of table of contents based on content association. *Int J Doc Anal Recognit* 8(2–3):132–143
26. Medvet E, Bartoli A, Davanzo G (2011) A probabilistic approach to printed document understanding. *Int J Doc Anal Recognit* 14(4):335–347
27. Nagy G, Seth S, Viswanathan M (1992) A prototype document image analysis system for technical journals. *IEEE Comput* 7(25):10–22
28. Rangoni Y, Belaïd A, Vajda S (2012) Labelling logical structures of document images using a dynamic perceptive neural network. *Int J Doc Anal Recognit* 15(2):45–55
29. Schürmann J, Bartneck N, Bayer T, Franke J, Mandler E, Oberländer M (1992) Document analysis – from pixels to contents. *Proc IEEE* 80(7):1101–1119
30. Shafait F, Breuel TM (2011) The effect of border noise on the performance of projection based page segmentation methods. *IEEE Trans Pattern Anal Mach Intell* 33(4):846–851
31. Shafait F, van Beusekom J, Keysers D, Breuel TM (2008) Document cleanup using page frame detection. *Int J Doc Anal Recognit* 11(2):81–96
32. Staelin C, Elad M, Greig D, Shmueli O, Vans M (2007) Biblio: automatic meta-data extraction. *Int J Doc Anal Recognit* 10(2):113–126
33. Story GA, O’Gorman L, Fox D, Schaper LL, Jagadish HV (1992) The rightpages image-based electronic library for alerting and browsing. *IEEE Comput* 25:17–26
34. Tan CL, Liu QH (2004) Extraction of newspaper headlines from microfilm for automatic indexing. *Int J Doc Anal Recognit* 6(3):201–210
35. Tsujimoto S, Asada H (1992) Major components of a complete text reading system. *Proc IEEE* 80(7):1133–1149
36. van Beusekom J, Keysers D, Shafait F, Breuel TM (2007) Example-based logical labeling of document title page images. In: Proceedings of the international conference on document analysis and recognition, Curitiba, pp 919–923
37. van Beusekom J, Shafait F, Breuel TM (2010) Combined orientation and skew detection using geometric text-line modeling. *Int J Doc Anal Recognit* 13(2):79–92

38. Wang S, Cao Y, Cai S (2001) Using citing information to understand the logical structure of document images. *Int J Doc Anal Recognit* 4(1):27–34
39. Wang Y, Phillips I, Haralick R (2004) Table structure understanding and its performance evaluation. *Pattern Recognit* 37(7):1479–1497
40. Wong KY, Casey RG, Wahl FM (1982) Document analysis system. *IBM J Res Dev* 26(6):647–656
41. Xiao Y, Yan H (2004) Location of title and author regions in document images based on the Delaunay triangulation. *Image Vis Comput* 22(4):319–329
42. Yu B, Jain AK (1996) A generic system for form dropout. *IEEE Trans Pattern Anal Mach Intell* 18(11):1127–1134
43. Zou J, Le D, Thoma G (2010) Locating and parsing bibliographic references in HTML medical articles. *Int J Doc Anal Recognit* 13(2):107–119

## Further Reading

- Aiello M, Monz C, Todoran L, Worring M (2002) Document understanding for a broad class of documents. *Int J Doc Anal Recognit* 5(1):1–16
- Medvet E, Bartoli A, Davanzo G (2011) A probabilistic approach to printed document understanding. *Int J Doc Anal Recognit* 14(4):335–347
- Rangoni Y, Belaid A, Vajda S (2012) Labelling logical structures of document images using a dynamic perceptive neural network. *Int J Doc Anal Recognit* 15(2):45–55

---

# Page Similarity and Classification

7

Simone Marinai

## Contents

Introduction.....	224
Evolution of the Problem .....	226
Applications.....	226
Main Difficulties.....	227
Chapter Overview.....	227
Page Representation.....	228
Global Page Representations.....	228
Representations Based on Lists of Blocks.....	231
Structural Representations.....	233
Page Comparison.....	236
Block Distance.....	236
Tree Distance.....	238
Region Classification.....	239
Page Classification.....	240
Page Retrieval.....	245
Page Clustering.....	250
Conclusions.....	250
Cross-References.....	251
References.....	251
Further Reading.....	253

---

### Abstract

Document analysis and recognition techniques address several types of documents ranging from small pieces of information such as forms to larger items such as maps. In most cases, humans are capable of discerning the type of document and therefore its function without *reading* the actual textual content. This is possible because the layout of one document often reflects its type.

---

S. Marinai

Dipartimento di Ingegneria dell'Informazione, Università degli studi di Firenze, Firenze, Italy  
e-mail: [simone.marinai@unifi.it](mailto:simone.marinai@unifi.it)

For instance, invoices are more visually similar to one another than they are to technical papers and vice versa. Two related tasks, page classification and page retrieval, are based on the analysis of the visual similarity between documents and are addressed in this chapter. These tasks are analyzed in this chapter in a unified perspective because they share several technical features and are sometimes adopted in common applications.

---

**Keywords**

Block distance • Global features • Graph • Page classification • Page clustering • Page representation • Page retrieval • Region classification tree • Tree distance

---

## Introduction

The first applications of document classification have been considered in the context of office automation, in particular in form processing applications. In this context, form classification methods are aimed at selecting one appropriate interpretation strategy for each incoming form [29, 47]. Other approaches addressed the problem of grouping together similar documents in business environments, for instance, separating business letters from technical papers [18]. One early approach for business document classification was presented in [48] where documents belonging to different classes are identified on the basis of the presence of class-specific landmarks. The rules implemented to identify the landmarks are context-based and rely also on the text recognized by an OCR engine. Since no positional information is modeled, the classification performed in this case is of little interest for layout-based page classification. More recently the classification of pages in journals and books received more attention [25, 45], in particular in applications related to Digital Libraries (DL).

When considering the visual appearance of documents, we can notice that in different application scenarios, the pages can be classified taking into account different features of the pages. In some cases, for instance, when dealing with preprinted forms, the categories have a fixed layout and two documents could be split apart just because one ruling line is placed in a different position in the two cases. Some documents, for instance, papers published in one specific journal, are more variable but are still constrained to obey to some explicit (or implicit) rules. In other cases the documents can be distinguished only considering their textual content and the layout similarity is of little interest. As a consequence, also the visual similarity between documents can vary by application. Some examples of different classes considered in the context of Digital Libraries are shown in Fig. 7.1.

One task related to page classification is Web page classification, or Web page categorization, where techniques related to text categorization [43] can be integrated with approaches that analyze the Web page layout [46] and the structure inferred from hyperlinks [42].



**Fig. 7.1** Examples of the five classes considered in [10]. From left to right: advertisement, first page, index, receipts, regular

## Evolution of the Problem

In the late 1990s, thanks to several Digital Library projects, large collections of digitized documents have been made available on Internet or in off-line collections. The first attempts to extend page classification techniques to digitized document in DLs such as books and journals needed to face the intrinsic ambiguity in the definition of classes that may change in different application domains. As a consequence, in some areas page classification techniques evolved in layout-based document image retrieval where pages are sorted on the basis of the similarity with a query page, rather than being explicitly labeled with one class.

## Applications

There are various applications in Document Image Analysis and Recognition (DIAR) where the comparison of documents by visual appearance is important. Examples include document genre classification, duplicate document detection, and document image retrieval.

A document genre [6, 13] is a category of documents characterized by similarity of expression, style, form, or content. In this chapter we are more interested on the visual genre that dictates the overall appearance of a document. Genre classification can be used to group together documents that should be processed with document analysis algorithms tuned on specific types of documents so as to improve the overall performance.

Other applications of document classification include the automatic routing of office documents in different work flows (e.g., identifying the type of one filled form) and the automatic archiving of documents, especially in the field of Digital Libraries.

One important issue in page classification is the a priori definition of an exhaustive set of classes that should not change later. This issue is particularly important in DL applications where the set of layout classes may be modified when switching from one collection to another. Moreover, different user needs could require different labels assigned to the same pages. For instance, general public may be more interested on macroscopic page differences, e.g., looking for pages with illustrations. On the opposite, scholars of typography may look for fine-grained differences, e.g., looking for pages with illustrations in specific zones of the page. To address these ambiguities in the definition of classes, recent applications have considered layout comparison techniques in document image retrieval frameworks often applied in the context of Digital Libraries [35]. In DLs the document image retrieval based on layout similarity offers to users a new retrieval strategy that was possible before only by manually browsing documents (by either interacting with physical books-journals or dealing with online images on DLs).

Page classification and retrieval can be considered both for scanned documents and for born-digital documents (see ▶Chap. 23 (Analysis of Documents Born Digital)). One typical application of interest in the case of born-digital documents

is the identification of pages containing the table of contents of books that can be useful to allow an easy navigation of book content [37].

## Main Difficulties

One significant difficulty of page classification and retrieval is the intrinsic ambiguity of the page similarity that arises when the same set of documents is used in different applications. For instance, pages in a collection of technical papers can be grouped considering the role of the page in each paper (e.g., title page vs. bibliography page) or according to the typographical rules of a given publisher (e.g., IEEE vs. Springer). This is in contrast, for instance, with character recognition where the class of a given symbol is in most cases not ambiguous and independent of the application.

As discussed by Bagdanov and Worring in [6], page classification can be addressed at two levels of detail. Coarse-grained classification is used to distinguish documents with a significant difference of characteristics, for instance, separating technical articles from book pages. At a lower resolution, a fine-grained classification is adopted to identify subclasses of documents that would otherwise be grouped together from a coarse-grained point of view. For instance, we can in this case distinguish pages of articles typeset by IEEE or by Springer.

## Chapter Overview

This chapter is organized taking into account the features shared by page classification and page retrieval. In particular, section “[Page Representation](#)” addresses the main approaches that have been considered to represent the page layout for either page classification or retrieval. In some cases, the same representation can be used to solve both problems. Several levels can be considered in the page representation. Each representation implicitly defines the possible approaches that can be adopted to compute the distance (or similarity) between pages. Alternative approaches to compare page layouts are described in section “[Page Comparison](#).“ Section “[Region Classification](#)” briefly summarizes some approaches that have been proposed to perform region classification. The latter task aims at assigning a suitable label to zones in the pages that have been previously identified as containing a uniform content. Region classification is described in this chapter, even if it is closely related to more general layout analysis techniques (►[Chap. 5](#) (Page Segmentation Techniques in Document Analysis)), for two main reasons. First, some approaches for region classification are similar to page classification techniques. Second, page classification and retrieval often takes into account labeled regions that can be obtained as described in section “[Region Classification](#).“

The next three sections analyze how page representation and page similarity can be put together in order to implement sections “[Page Classification](#),“ “[Page Retrieval](#),“ and “[Page Clustering](#).“ Some representative systems are summarized in

these sections according to the various page representations considered. Due to the large number of different application domains that have been addressed in recent years, it is not possible to provide an effective comparison of the performance achieved by the systems. For page classification we report the classification accuracy of some methods that are discussed in the paper. We also highlight the main techniques used for performance evaluation in page retrieval systems. The section “[Conclusions](#)” closes this chapter.

---

## Page Representation

In many application domains, users are generally capable of discerning the type of document and therefore its function without *reading* the actual text content since the layout structure of a document often reflects its type.

The visual appearance of a document is completely determined by the whole set of pixels in its image representation. Even if this description is complete, it is in general too complex to directly process the pixels acquired by the document scanner. Besides computational problems, higher-level representations are needed in order to extract a more abstract page representation that is required to achieve generalization in the classification and retrieval processes. Otherwise, over-fitting can occur, giving rise to the identification of similar pages only when these are near duplicate one of the other. According to Hu et al. [26] low-level page representations based, for instance, on bit maps are easy to compute and to compare, but do not allow for a structural comparison of documents. High-level representations are, however, more sensitive to segmentation errors.

Therefore, one essential aspect of page classification and retrieval systems is the kind of features that are extracted and used to represent the page. Sub-symbolic features, like the density of black pixels in a region, are usually computed directly from the image. Symbolic features, for instance, related to horizontal and vertical ruling lines, are extracted starting from one preliminary segmentation of the page. Likewise, structural features, such as relationships between objects in the page, are computed from a suitable page segmentation. Some techniques take into account textual features, such as the presence of keywords, that can be identified considering the text in the image recognized by an OCR (Optical Character Recognition) engine. The latter techniques are less relevant for this chapter that mostly deals with layout-based page classification and retrieval.

In the rest of this section, we analyze the principal approaches that can be used to describe the page layout. These approaches are also summarized in Table 7.1. The page comparison techniques that are used either in page classification or retrieval are described in the next section.

### Global Page Representations

The most straightforward adaptation of statistical pattern classification techniques to page classification and retrieval leads to the development of approaches that

**Table 7.1** Main approaches used in page representation for classification and retrieval

Representation		Description	Ref.
Global features	Page features	Image features computed at the page level	[6, 15]
	<b>Zoning:</b> image	Image features computed for each grid zone	[16, 26, 49]
	<b>Zoning:</b> character/symbol	Features computed from characters and accumulated in each grid zone	[16, 45]
	<b>Zoning:</b> region	Features computed from each region and accumulated in each grid zone	[32, 40]
List of blocks	<b>Lists of text lines</b>	Features computed for each text line. Page represented by the set of textlines	[27]
	<b>Discriminative blocks</b>	Sub-images discriminate one class from the others	[3]
	<b>Layout blocks</b>	Layout regions are stored in the list	[39]
	<b>Lists of lines</b>	Ruling lines are used to describe forms	[30]
Structural representation	<b>Graph</b>	Graph nodes correspond to layout regions; edges describe relationships between nodes	[4, 23]
	<b>Tree</b>	Hierarchical page decompositions (e.g., X-Y trees)	[2, 17, 21]

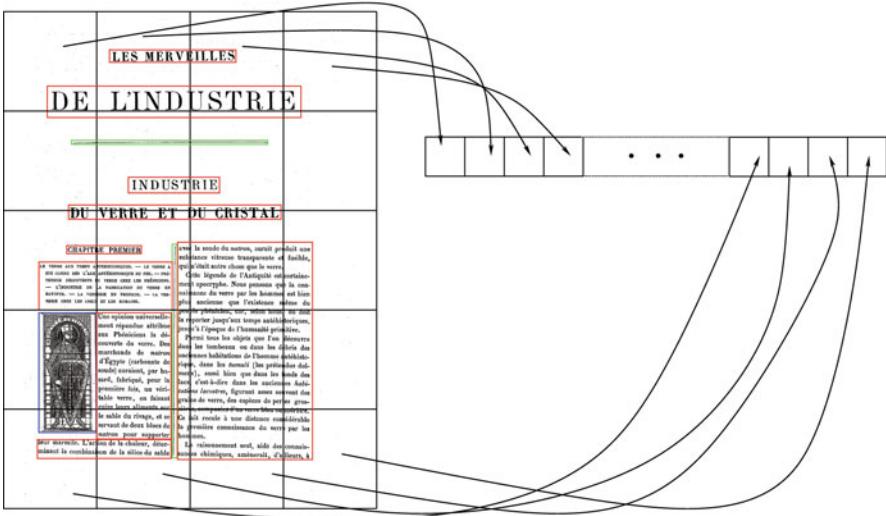
represent the pages with global image characteristics. These features are in most cases arranged into fixed-size feature vectors that are subsequently used as inputs to classifiers.

Documents containing a large amount of text can be described with features computed from the connected components. On the opposite, form documents (whose processing and classification are also described in ▶Chap. 19 (Recognition of Tables and Forms)) can be represented on the basis of ruling lines that characterize the page layout. In the latter case, due to the fixed page structure, the lines are organized in a structural representation (section “[Structural Representations](#)”).

When dealing with heterogeneous documents, the features are in most cases extracted with low-level image processing techniques. For instance, morphological operations and texture-based features are often considered.

In [15] a histogram for each of five features extracted from the page to be described is computed. The features are the word height, the character width, the horizontal word spacing, the line spacing, and the line indentation. Each histogram is smoothed with a standard kernel function.

Global page features computed by applying morphological operations on the binarized page image are extracted at multiple resolutions in [6]. The input image is morphologically opened with horizontal ( $x$ ) and vertical ( $y$ ) structuring elements of variable sizes. The page is subsequently mapped to one two-dimensional function that considers the area of the black zones in the transformed image obtained



**Fig. 7.2** One page with the grid used to compute the zoning vectorial representation

with a combination of  $x$  and  $y$  values. This distribution is then sampled in the  $x$  and  $y$  directions obtaining a rectangular size distribution. The size of this high-dimensional feature vector is reduced with Principal Components Analysis (PCA) so as to perform a feature extraction and allow an easy comparison of pages in the reduced space.

One widely used approach to represent the page with a fixed-size feature vector is based on the computation of the desired features in the zones identified by a grid (with sides parallel to the page border) superimposed to the page [26, 49]. This approach is often referred to as zoning. One example is shown in Fig. 7.2 where one  $4 \times 5$  grid is overlapped to page on the left. The features computed for each zone are concatenated row by row in the fixed-size feature vector represented on the right part of the illustration.

One of the first systems proposed to perform page retrieval [16] considers global page features stored in an 80-dimensional feature vector. Some of the features are related to the distribution of *interest points* (computed from high curvature points in the characters' contour), while additional information is obtained by calculating the density of connected components in each cell of one grid overlapping the page.

A zoning-based approach is described in [45] where the features are extracted from one page considering a grid overlapped to the image. Zones in the grid are called windows and the minimum size of windows is defined to assure that each window contains enough objects. Four types of windows are extracted on the basis of the zones identified by the grid: *basic* window (the smallest area defined by the grid), *horizontal* and *vertical strip* windows (a set of basic windows aligned horizontally or vertically), and the *page* window that covers the whole page. Several features are then computed from each window. Most features are extracted from

the connected components in the zone (e.g., the median connected component width) or directly from the image (e.g., the foreground-to-background pixel ratio). Specific features are computed for composite window types such as the column and row gaps. The overall description of a page is obtained from the concatenation of the features extracted from each zone, and therefore, standard statistical classifiers (such as decision trees) can be used to identify the page classes.

Another global representation whose features are related to zoning is described in [40] where blocks with uniform content are first extracted from each page. One new abstracted image containing only the block edges is then used to compute the global features. In particular, the horizontal and vertical projection profiles of the edges image are computed and concatenated, obtaining one feature vector of size  $w + h$  where  $w$  and  $h$  are the page width and height, respectively.

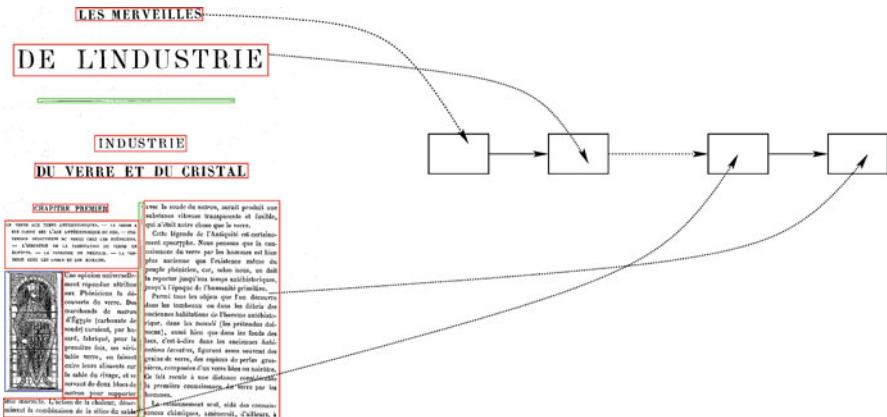
To balance low-level and high-level page representations, Hu et al. propose in [26] the *interval encoding* that allows to encode region layout information (i.e., intrinsically structural) into fixed-size vectors that can be easily compared each other using standard vectorial distances such as the Euclidean one. Pages are sampled with a grid of  $m$  rows and  $n$  columns. Each item in the grid is called bin and is labeled as text or white space according to the prevailing content of the corresponding area.

One different way to obtain a fixed-size representation from a variable number of regions is described in [32] where each document is indexed by considering a set  $RB$  of representative blocks. The representative blocks are identified by clustering, with  $k$ -means, all the blocks in the collection. The distance between blocks used by the clustering algorithm is based on the overlapping similarity or Manhattan distance between blocks. The features of the representative blocks are obtained by computing the average position of the blocks in each cluster. Each document is afterwards represented by measuring the similarity of each block in the page to each Representative Block and then aggregating all the similarities of page blocks in a feature vector having size  $|RB|$ . Once this page representation is computed, it is possible to evaluate the page similarity by means of the cosine of the angle between the feature vector of the query page and the feature vector of each indexed page.

In [11] global and structural features (computed from an X-Y tree page decomposition) are merged in the page representation. The global features allow to obtain one representation that is invariant with respect to the book dimensions.

## Representations Based on Lists of Blocks

Some of the page representations described in the previous section consider as input the blocks identified in the page. In this section we analyze approaches where the page is explicitly represented with a list of objects (either blocks or ruling lines) rather than summarizing these objects with global numerical features. This type of representation is graphically depicted in Fig. 7.3 where one sample image is shown on the left with colored rectangles corresponding to homogeneous regions. The whole page is represented with one linked list of items that is shown on the right part of the illustration.



**Fig. 7.3** Page representation with a list of blocks

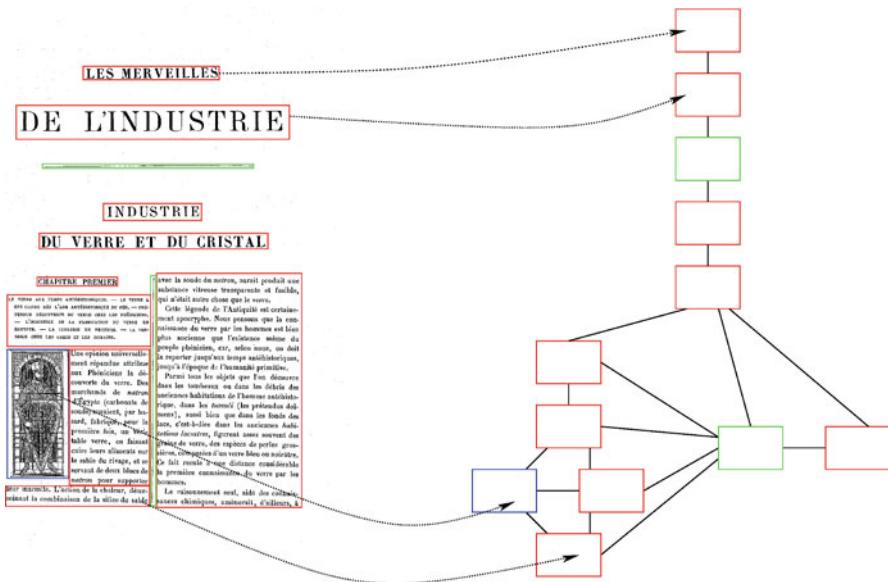
The lists of objects are in most cases matched with analogous lists extracted from reference pages to compute the page similarity. This page representation is more accurate with respect to global page representations, but it is more difficult to handle by comparison algorithms. One of the reasons is the variable size of representations of different but similar pages.

Many algorithms in layout analysis address Manhattan layouts where the page is assumed to be both physically and logically composed by upright rectangles containing uniform content (e.g., text and graphics). ► Chaps. 5 (Page Segmentation Techniques in Document Analysis) and ► 6 (Analysis of the Logical Layout of Documents) contain extensive descriptions of layout analysis algorithms.

Document images can be described by means of blocks that correspond to text lines. Huang et al. [27] exploit this representation by describing the page layout with the quadrilaterals generated by all pairs of lines. To speedup the retrieval, the quadrilaterals are clustered and cluster centers in the indexed pages are compared with the query one.

While most methods to identify blocks in the page are class independent, it is possible to identify discriminant areas in the page that can act as signatures for classes. In form classification, Arlandis et al. [3] describe a method based on the identification of discriminant landmarks (the  $\delta$ -landmarks) that are sub-images suitable to discriminate one class from the others. The set of  $\delta$ -landmarks of one class contains the sub-images having a significant dissimilarity with respect to the other classes. The class models consist of sets of  $\delta$ -landmarks and the dissimilarity between sub-images and  $\delta$ -landmarks is used to classify unknown forms.

One page representation based on a list of blocks is described in [39] where rectangular regions are stored in a Component Block List (CBL) and sorted considering the position of the bottom-left region corner. Page layouts are matched considering the block location and the size attributes. The CBL of one template page is called Template Block List (TBL) and the page classification is achieved by finding the most similar TBL to the CBL of the unknown page. The comparison is



**Fig. 7.4** Page representation with an adjacency graph

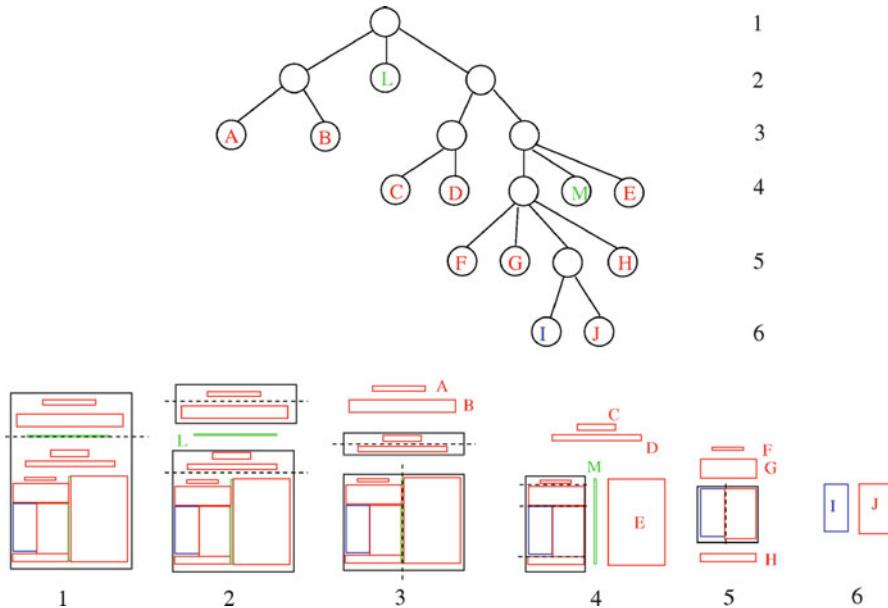
made with a simple block-matching algorithm that finds for each block in TBL the most similar block in CBL, according to size and position of blocks.

A similar approach can be applied in form classification by considering lists of ruling lines instead of lists of blocks. Jain and Liu [30] propose a form retrieval system where the form class is identified considering one form signature from the lists of horizontal and vertical frame lines. Collinear horizontal and vertical lines implicitly define a grid that contains the form cells. The form signature describes the number of cells contained in each grid element. To deal with missing lines in actual forms (due to noise or design differences among forms of the same class), several signatures are computed for each class considering alternative forms.

# Structural Representations

Structural representations of the page layout can be based either on graphs or on trees.

Graph nodes correspond in most cases to text regions that are described with numerical attributes computed from the position and size of the region, from information related to the text size, and from features computed from text lines. Graph edges represent the adjacency relationships between nodes. One example is shown in Fig. 7.4 where the page used in the previous examples is now described with one adjacency graph. Each region is in this case represented by one graph node and the edges link neighboring regions.



**Fig. 7.5** Example of X-Y tree

In [4] *Attributed Relational Graphs* (ARG) are used to represent the page layout by considering the neighboring relationship of zones in the page. Document genres (classes) are represented by First-Order Random Graphs (FORG). An FORG is trained from documents belonging to one class. An unknown page described with an ARG is then classified by identifying the FORG with maximum conditional probability.

A graph-based approach is proposed by Gordo and Valveny in [23]. In the cyclic polar page representation, the nodes of the graph correspond to regions in the page and are labeled with the area and the type of region (text or non-text). The edges connect each node with the center of mass of all the regions, and therefore, one complete bipartite graph is built. Edges are labeled with their length and with the angle formed with adjacent edges. This polar graph can be mapped to one sequence of items labeled with node attributes. A cyclic shift of the sequence defines one equivalent representation of the same page.

Taking into account the hierarchical nature of document images, it is natural to adopt tree-based representations of the page layout. Tree-based representations are appropriate also because some segmentation techniques, such as the recursive X-Y tree, are based on a recursive decomposition of the page (►Chap. 5 (Page Segmentation Techniques in Document Analysis)). The root of an X-Y tree is associated to the whole page and the corresponding region is recursively segmented, splitting the page along horizontal or vertical white spaces. The tree in Fig. 7.5 describes the regions in the page used in the previous examples in an X-Y tree.

The bottom part of the illustration represents the segmentations made at the various tree levels. In this example we show also cuts along horizontal and vertical ruling lines (e.g., the first cut is made along a horizontal line).

In several application domains (in particular, when dealing with forms) ruling lines are an essential element of the page layout. To deal with documents containing ruling lines, one modified X-Y tree (the MXY tree) has been introduced by Cesarini et al. [8] and used in some tree-based page classification systems (e.g., [2, 7, 19, 21]).

The X-Y tree-based approaches to page classification and retrieval take advantage of the limited variability of the tree representation when representing different, but structurally similar, layouts. On the other hand, the segmentation and representation with X-Y tree is difficult in case of noise, skew, and in general when dealing with non-Manhattan layouts.

An early tree-based representation of the page layout has been presented in [17] where a decision tree (the Geometric Tree, *GTree*) is used to model the layout classes by describing a hierarchy of possible object organizations. In analogy with X-Y trees, the pages are described with a recursive segmentation of the page along white spaces. In this representation, each example is one leaf of the *GTree* while intermediate nodes are used to identify cuts shared by different classes. Cuts shared by many documents are described in nodes close to the root, whereas class-specific cuts are contained in nodes closer to the leaves.

To classify one unknown document, the tree is traversed from the root to one leaf that identifies the page class. In each node encountered in this navigation, the input document is compared with the cuts defined in the node's children and the best path is followed.

Another approach that uses decision trees to classify pages represented with X-Y tree is described by Appiani et al. in [2]. Nodes in the Document Decision Tree (DDT) contain MXY trees because cuts along horizontal and vertical ruling lines are required to segment forms and other business documents that are addressed by the system. Both DDT building and traversal are based on sub-tree matching between as described in section “[Tree Distance](#)”.

The use of one X-Y tree representation for unsupervised page classification is described in [34]. Each node,  $v$ , describes a region with one feature vector  $f^v = (f_1^v, f_2^v, f_3^v, f_4^v)$ , where  $f_1^v$ ,  $f_2^v$ , and  $f_3^v$  denote the average character font size, the level in tree, and the  $X$  coordinate of the region center. If  $v$  is a node on a  $Y$  projection profile,  $f_4^v$  is the minimum vertical distance between  $v$  and its top or bottom sibling on the profile. The tree matching is performed with a tree-edit distance algorithm (section “[Tree Distance](#)”).

Even if X-Y trees and related data structure have been the prevailing tree representations adopted in document image classification and retrieval, other approaches have been considered as well. For instance, quadtrees are used in [41] to address form classification. Another hierarchical page representation is the L-S tree (Layout Structure Tree) proposed by Wei et al. in [53]. Even if using a different notation, the page decomposition and representation addressed by L-S trees is analogous to X-Y trees.

## Page Comparison

In this section we analyze how the distance between page descriptions is used to compare two pages. This distance can be used both to classify the pages and to rank the pages most similar to one query page. The main difficulties when computing the page similarity are due to layout differences induced by low-level segmentation errors and to variations in the design of documents that should be grouped in the same class.

When pages are represented with global features (section “[Global Page Representations](#)”), p-norms are natural choices to compute the distance between pages. For instance, [40] adopts the  $L_1$  distance to compare global vectors describing the pages.

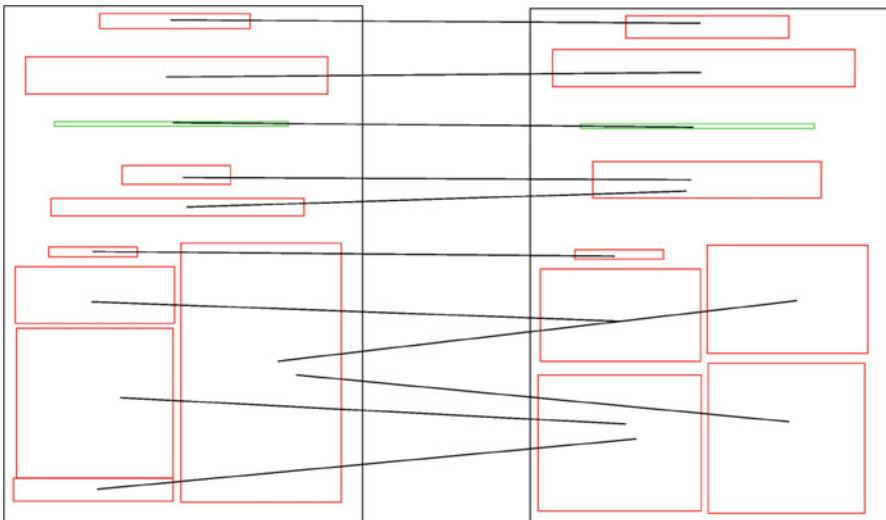
One widely adopted strategy to map images in a fixed-size feature vector is the zoning approach described in section “[Page Representation](#).“ When dealing with these representations, one significant problem is related to the misalignment of pages that are very similar but horizontally and/or vertically displaced. Approaches based on the edit distance can address this problem, but the high computational cost of these techniques limits its actual use on large datasets. The approach proposed in [26] is based on the representation of text regions considering the interval encoding that represents how far is a text bin from one background area. With this representation, it is possible to take care of page translations by using a less expensive  $L_1$  distance between the fixed-size vectorial representations.

Other global features are considered in [15] where the pages are represented by histograms describing the distributions of word height, character width, horizontal word spacing, line spacing, and line indentation. The similarity for each feature is computed considering the distance between the distributions by means of the Kullback–Leibler divergence. Two pages are compared by combining with an SVM classifier the five similarity measures together with a text-based similarity.

A related problem is addressed with the cyclic polar page layout representation proposed by Gordo and Valvenyl [23]. In this case two pages are compared considering the distance between the corresponding sequences. It is possible to measure the distance between two sequences by using either the edit distance or the Dynamic Time Warping algorithm provided that one suitable cost is defined for the transformation of one node into another.

## Block Distance

In the case of block-based page representations (section “[Representations Based on Lists of Blocks](#)”), the upright rectangular blocks identified by layout analysis tools are considered to compute the page similarity. Blocks in one page are matched with blocks in the other pages by minimizing one global page distance. The latter is computed by combining the distances between pairs of blocks.



**Fig. 7.6** Matching of blocks between two pages

Several methods that can be used to compute block distances are compared in [50]. Block comparison can take into account features that describe the position, size, and area of the blocks. The page comparison is based on an assignment of blocks from the query page to blocks in the reference layout that minimizes the overall distance. The latter is computed by summing up the block distances of the matched blocks. For instance, in Fig. 7.6 we show one hypothetic assignment of blocks in one page with blocks in a second page.

The block distances compared in [50] are the Manhattan distance of corner points, the overlapping area between blocks, and other simple block distances such as the difference in width and in height. Methods adopted to solve this *matching problem* should be tolerant to broken and merged blocks and are expected to be fast to compute. In [50] three alternative matching strategies are compared: the matching based on the assignment problem, the matching obtained by solving the minimum weight edge cover problem, and the use of the Earth Mover's Distance.

The computation of a similarity based on matching can be considered also when documents are represented with lists of lines. In [30] the similarity between one query page and one reference form is computed by searching for the optimal grid pair which results in the minimal overall cell count difference. The grid pairs considered correspond to all the signatures computed for each form. The similarity measure attempts to match the similar parts of two forms as much as possible in terms of the cell count difference. As a consequence, this measure is robust to incorrect line groupings.

Similarly to block matching, pages can be compared by aligning the bin rows that represent the pages [26]. The bin rows are matched using the interval encoding as described in section “[Global Page Representations](#).” In the case of classification,

this alignment is obtained by using Hidden Markov Models to model different layout classes. HMMs are used because pages are best characterized as sequences of vertical regions.

## Tree Distance

When the document layout is represented by means of a tree-based description, such as the X-Y tree, three main approaches can be used in the comparison: flatten the tree structure into one fixed-size representation, use tree-edit distances that take into account the hierarchical structure, or use decision trees where a suitable distance between trees is defined to navigate the decision tree.

One flat representation of X-Y trees is described in [9] where the tree is encoded into a fixed-size representation by counting the occurrences of tree patterns composed by three nodes in a way that is somehow similar to  $N$ -gram representation of text strings. The tree patterns are composed of three nodes connected one to the other by a path in the X-Y tree. Trees made by three nodes have two structures: one composed of a root and two children and one composed of a root, a child, and a child of the second node. Considering all the symbolic labels that can be assigned to nodes, a total of 384 possible tree patterns can be defined. In [9] this encoding is used by a feed-forward neural network to classify the pages.

Decision trees (in particular, the Geometric Tree, *GTree*) have been initially used by Dengel [17] to model the layout classes by describing a hierarchy for possible logical object arrangements.

More recently, Appiani et al. in [2] describe the classification of X-Y trees by means of a Document Decision Tree (DDT). DDT nodes contain X-Y trees that are compared with the input tree during classification. The tree comparison is made by looking for the maximum common sub-trees. During one generic step of the classification, the system compares the input X-Y tree to the trees stored in the decision tree nodes and follows the path with the minimum distance until a leaf with one class label is found.

Diligenti et al. [19] propose Hidden Tree Markov Models (HTMM) for modeling probability distributions defined over spaces of X-Y trees. Features are computed from the zone corresponding to each tree node. The generative model is a hidden tree of states which represent the zones. Since the model is a special case of Bayesian networks, both inference and parameter estimation are derived from the corresponding algorithms developed for Bayesian networks. In particular, the inference consists of computing all the conditional probabilities of the hidden states, given the evidence entered into the observation nodes (i.e., the labels of the tree).

One problem in the use of X-Y tree segmentation algorithms is that sometimes these do not produce similar trees starting from similar pages. Supervised classifiers often require the availability of a large enough training set that is expected to model the different trees generated by the segmentation algorithms. In some application domains, these training sets are not easily available and therefore alternative strategies should be explored. In [7] this problem is approached by considering a

training set expansion that is based on the manipulation of labeled X-Y trees with a tree grammar. The new training set is built by merging the labeled samples and the artificial ones. The artificial trees are generated with a tree grammar to simulate actual distortions occurring in page segmentation. For instance, one rule splits a text node into two text nodes simulating the presence of a paragraph break. Another rule is used to add one text node below an image node to simulate the presence of a caption below the illustration.

During classification, one unknown tree is classified by comparing it, using one tree-edit distance algorithm, with the trees in the expanded training set. One  $k-nn$  classifier is then used to determine the tree class. In analogy with the string edit distance, the tree-edit distance between two trees is the cost of the minimum-cost set of edit operations that are required to transform one tree into the other. In [7] the tree-edit distance algorithm proposed in [54] by Zhang and Shasha is used.

In [34] the pages are represented with X-Y trees and one edit distance based on Wang et al. algorithm [51] is used. The general algorithm is adapted by defining the distances between the X-Y tree nodes in case of substitutions, insertions, and deletions. In particular, the cost for replacing node  $v$  with node  $w$  (and vice versa) is defined by using a weighted Euclidean distance (the Karl Person distance) that takes into account the variance of each feature.

A related approach is described in [53] where pages are represented with L-S trees that are analogous to X-Y trees. Classes of documents are represented by computing the largest common sub-tree (called Document-Type Tree) in a collection of trees representing the training pages of one class. The class of an unknown page is obtained by finding the Document-Type Tree closest to the L-S tree computed from the page.

---

## Region Classification

Region classification is a task that is of interest both for layout analysis (►Chap. 5 (Page Segmentation Techniques in Document Analysis)) and for page classification and retrieval. This topic is discussed in this chapter because region classification techniques are similar to those used in page classification and because region classification is a component of some page comparison methods.

In this section we discuss classification methods used to identify the general region content (e.g., text vs. non-text) without considering the actual text in the region. On the opposite, when we aim at identifying the purpose of one text region on the basis of its contents (often recognized by an OCR engine), we deal with functional labeling applications.

The first region classification approaches used global features computed from the region together with linear classifiers designed with hand-tuned parameters [44]. Nowadays, in most cases trainable classifiers are used for region classification. For instance, in [14] texture features are adopted in combination with a decision tree to classify the regions into text or non-text.

Decision trees for region classification are described also in [1] where rectangular regions are found with a variant of the Run Length Smoothing Algorithm (RLSA) (see ►Chap. 5 (Page Segmentation Techniques in Document Analysis)). The regions are described with features based on their dimensions, the number of black pixels, and the number of black-white transitions along horizontal lines. Other features are based on statistics obtained from the run lengths in the region. The labels assigned to regions are text block, horizontal line, vertical line, picture (i.e., halftone images), and graphics (e.g., line drawings).

A similar region classification approach is described by Wang et al. in [52] where a 25-dimensional feature vector is used as input to 1 decision tree trained to classify regions among 9 classes: large text, small text, math, table, halftone, map/drawing, ruling, logo, and others.

The discrimination of text and non-text regions in handwritten documents is described in [28] where a top-down segmentation is followed by an SVM classifier. The four segmentation algorithms considered are X-Y decomposition, morphological smearing, Voronoi diagram, and white space analysis. Various types of features are used as input to the classifier. The first features are based on run-length histograms of black and white pixels along the horizontal, the vertical, and the two diagonal directions. Another set of features is related to the size distributions of the connected components in the region. The last group of features is based on a two-dimensional histogram of the joint distribution of widths and heights and a histogram of the nearest neighbor distances of the connected components.

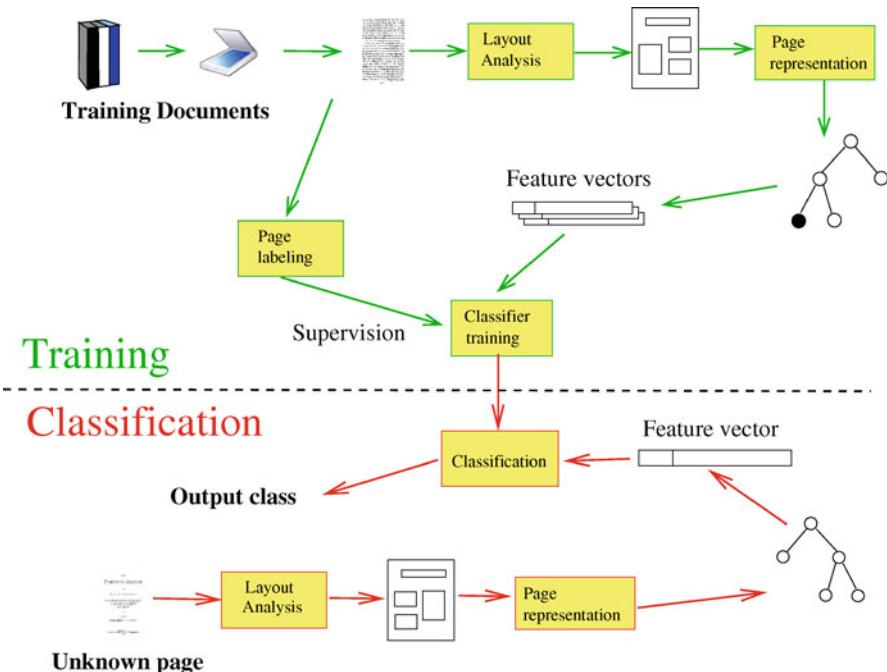
---

## Page Classification

Methods for page classification and page retrieval significantly overlap both from the methodological point of view and from the application one. Many papers test the proposed page similarity approaches in both page retrieval and classification contexts. In some cases page retrieval is actually made by exploiting the page classification results, and pages with the same class of the query are shown to the user. In the majority of cases to provide a quantitative evaluation of the system, the performance of page retrieval is measured taking into account a labeling of pages into a finite disjoint set of classes.

In this and in the next section, we summarize some significant approaches that have been proposed to combine page representation and page comparison in the context of page classification and retrieval. The criterion that we adopted for organizing the approaches in the two sections is based on the type of measure presented by the authors. In page classification we consider methods where the performance is measured with the classification precision (percentage of patterns in the test set that are correctly classified). In page retrieval we include methods where all the pages in the dataset are compared with the query and are then sorted according to some similarity measure.

The overall objective of page classification is to assign one label to each document image. In Fig. 7.7 we graphically depict the organization of the main



**Fig. 7.7** General approach for page classification

modules considered in most page classification systems. In the illustration the page representation is described with a tree, but other representations can be considered as well. The page labeling box requires a human annotation of the class that should be assigned to each page in the training set that is used to train the supervised classifier. The feature vectors encode the page representation in a format that is handled by the classifier both during the training and when classifying an unknown page as described in the bottom part of the illustration.

Form classification and automatic document organization are traditional applications in the office domain. Form classification is aimed at selecting an appropriate reading strategy for each form to be processed and often takes into account the presence of ruling lines in the page layout [29, 33, 47]. In other cases the goal is to group office documents, for instance, setting apart business letters from technical papers [18, 48].

More recently, page classification has been adopted in Digital Library applications where it can be used to discover page-level metadata (e.g., identifying the table of contents page) and to locate pages where to look for specific metadata (e.g., the title of a book can be found in the cover page). In this case the classification addresses scanned pages in journals and books [25, 31, 45].

Whereas labels for region classification are quite standard (e.g., we can look for text, graphics, and line drawing regions), a broad range of categories can be

considered in page classification. For instance, one document can be identified as *technical paper* or *commercial letter*; the publisher of one technical paper can be *IEEE* or *ACM*; the layout can be based on two columns or one column; and the type of page can be *title page*, *regular page*, or *index page*.

In [45] relevance-defined data are identified from human annotation of pages in the UW-I dataset in 12 visually different classes. Interestingly, many pages are often labeled with multiple classes, and this is a clear demonstration of the intrinsic ambiguity of page classification in some application domains. Users also provided scores of degree of similarity between the images and the classes, and this information was used to define the training set for the page classification task.

One important issue that must be addressed by page classification systems is the need to deal with generic classes that include heterogeneous pages. Pages in these classes should be grouped together from the application point of view but can have significantly different layouts. This organization of page labels can be dictated by application constraints it is not easy to be addressed by page classifiers that should otherwise finely distinguish between more homogeneous classes. For instance, in [2] 1 class contains more than 20 different subtypes of forms. Forms are addressed also in [3] where 1 class contains 205 pages not belonging to the training classes. In the classification of the first page of journal pages described in [5], one class actually contains three variations of the same logical class.

The page representation and distance computation (section “[Page Comparison](#)”) are combined to build a complete page classification system. In the rest of this section, we summarize some representative systems whose approaches for page representation and matching have been previously described in this chapter. To provide a synthetic view of the alternative approaches to page classification, we compare the main features of some representative page classification methods in Table 7.2.

Appiani et al. [2] combine a page representation based on MXY trees with one decision tree classification algorithm. The experiments are made on two datasets. The first dataset contains 753 single-page invoices split into 9 classes. In the training set, 20 pages per class are used. The classification accuracy is 97.8 %. The second experiment works with bank account notes that comprise four classes: batch header, check, account note, and enclosures. The last class actually contains more than 20 different types of forms. The training is made with 67 documents and the test with 541 documents. Considering the three main classes (batch header, check, account note), the classification accuracy is about 99 %.

The integration of MXY tree page representation with Hidden Tree Markov Models is described in [19]. In this paper the tests are made with 889 commercial invoices issued by 9 different companies that define the class. The number of samples per class is not balanced and is comprised between 46 and 191. With 800 training examples, the accuracy is 99.28 %.

Even if described as a form retrieval system, the technique described in [30] classifies the pages on the basis of the distance (see section “[Representations Based on Lists of Blocks](#)”) with respect to forms in the database. The experiments are made on a dataset containing 100 different types of forms including both blank forms

**Table 7.2** Main features of some representative page classification methods. Data: type of documents addressed; Representation: page representation adopted; Classification: method adopted for page comparison/classification; NC: number of classes; Ref: cited paper

Data	Representation	Classification	NC	Ref.
Journal/letter/ magazine	Zoning	$L_1$ distance	5	[26]
Journal (UW-I)/tax forms	Zoning at character level	Decision tree	12–20	[45]
Tax forms	Zoning at character level	SOM clustering	12–20	[45]
Synthetic forms	Global features	$L_1$ distance	1,000	[40]
Article first pages	List of blocks	Optimal block matching	9–161	[50]
Hand-filled forms	Discriminative blocks	Block distance	7	[3]
Tax forms	List of ruling lines	Optimal line matching	100	[30]
Account note/invoice	X-Y tree	Decision tree	3–9	[2]
Article pages	X-Y tree encoding	Artificial neural networks	5	[10]
Article pages	X-Y tree encoding	SOM clustering	5	[36]
Article first pages	X-Y tree	Tree-edit distance + K-medoids clustering	25–150	[34]

and filled-in forms. The experiments performed with 200 random queries provide a classification accuracy of 95 %.

Form classification is addressed in [3] considering a class model consisting of a set of  $\delta$ -landmarks. One document is assigned to one class if its sub-images match a significant number of the  $\delta$ -landmarks that define the class. Each class model is built from one reference image. In total, there are seven forms in the reference set that include pages with very similar layouts differing in a few text areas or words like field names and page numbers. Two experiments are reported. The first one is made with 753 actual form pages belonging to the reference set. The second experiment is made with 205 document images, mostly forms, not belonging to any of the 7 reference pages.

Form classification by means of global features matched with p-norms is described in [40]. The proposed approach is tested on a large dataset of prototype forms. Test pages are generated by simulating various image deformations caused by filled-in document contents, noise, and block segmentation errors. For each experiment 20,000 test images are generated. The classification accuracy is measured with a variable number of classes and decreases from 99.77 % with 50 classes to 99.25 % with 1,000 classes.

In [45] a decision tree classifier and Self-Organizing Maps are combined to classify documents in five main classes: cover, reference, title, table of contents, and form. Three main experiments have been performed to evaluate the use of decision tree and Self-Organizing Map classifiers on relevance-defined, user-defined, and explicit-instance classes.

With relevance-defined classes the decision tree classifier has an accuracy of 83 %. In user-defined classes pages from a collection of office documents are grouped in four classes: cover, title, table of contents, and references. This collection has been integrated with forms from the NIST Special Database 2 Structured Forms Reference obtaining 261 images split into 5 classes. In this case the decision tree classifier has an accuracy of nearly 90 %. In the explicit-instance classes experiment pages came from the previous NIST Special Database 2 that contains 5,590 pages split in 20 types of taxform pages. The training set was based on 2,000 random images and other 2,000 were used in the test set. On these data the decision tree classifier has a classification accuracy of 99.70 %. When using the SOM classifier (in a semi-supervised approach), a classification accuracy of 96.85 % is reported.

Cesarini et al. [10] describe a page classification system aimed at splitting documents (belonging to journals or monographs in Digital Libraries) on the basis of the type of page. The five classes are advertisement, first page, index, receipts, and regular. The structural representation is based on the Modified X-Y tree. The page is classified by using artificial neural networks working on a fixed-size encoding of the page MXY tree. The test set is fixed with 300 pages while the size of the training set varies from 30 to 300 pages.

Pages from five technical journals belonging to Digital Libraries are considered in the experiments described by Bagdanov and Worring [5]. In the classification task, the first page is considered to identify the corresponding journal from 857 articles. The classification task is complicated by the fact that one class contains three variations of first pages with one, two, and three text columns, respectively. The layout is represented with attributed graphs, which naturally leads to the use of First-Order Gaussian Graph (FOGG) as a classifier of document genre. In the experiments a variable number of training pages are considered.

The use of tree-edit distance for classifying book pages represented with MXY trees is discussed in [7]. The classification is made with  $k - nn$  where nearest training pages are computed by means of the tree-edit distance. The peculiarity of the approach is the use of training set expansion to improve classification performance. The experiments are made on two volumes of one nineteenth-century Encyclopedia containing 1,300 pages split into 7 classes (e.g., page with illustration, first page of a section, text on two columns) not evenly distributed.

A tree-based representation and subsequent classification by means of a tree comparison are presented in [53] where pages are described with L-S trees (similar to X-Y trees). The system is tested on a dataset composed of 40 pages for each of the 8 classes (letter, memo, call for papers, first page of articles in 5 different journals). Fifteen pages per class are used for training. The remaining 200 pages are used to test the classification that apart from some rejected pages (8 % of letters, 8 % of memos, and 20 % of calls-for-papers) correctly classifies all the documents.

Beusekom et al. in [50] compare several block distances and matching algorithms to classify pages in the MARG (Medical Article Records Ground-truth) database [22] that contains 815 first pages of scanned medical journals, labeled by layout type and journal. The page layout is represented by lists of blocks. The best results are reported when using the overlapping area as block distance and the matching

method based on the solutions for the minimum weight edge cover problem. The classification is performed with a 1-*nn* classifier. The error rate computed with a leave-one-out approach is 7.4 % when looking for the page type (9 classes) and 31.2 % when looking for the journal type (161 classes).

Global features computed from page zoning are described in [16]. The similarity of indexed pages with respect to the query page (either hand-drawn or selected by the user with a graphical interface) is computed by means of the Euclidean distance between 80-dimensional feature vectors. The experiments are performed on images belonging to seven classes (journal, business letter, brochure, handwritten, newspaper, catalog, magazine). A leave-one-out nearest neighbor classifier was considered in the experiment that involved 939 documents. For each test page, one of the three nearest documents is in the right class with a recognition rate of the 97 %.

The use of interval encoding features (section “[Global Page Representations](#)”) for page classification is proposed in [26] where each class is described with a Hidden Markov Model that represents the page as a sequence of vertical regions, each having some horizontal layout features (number of columns and widths and position of each column) and a variable height. The experiments are performed on 91 documents belonging to 5 classes (1-col-journal, 2-col-journal, 1-col-letter, 2-col-letter, magazine). The accuracy reported is comprised between 64 % (class 2-col-letter) and 100 % (class 2-col-journal).

---

## Page Retrieval

In some application domains it is difficult, or even impossible, to use the page classification paradigm. The problems arise when the classes are not defined in a standardized way, and therefore, the labels assigned to pages can be ambiguous or subjective. For instance, when dealing with books in Digital Libraries, some pages (e.g., Fig. 7.8) could be labeled as *two-column text* or *illustration* or *section start* or any combination of the above.

One possible solution to this problem relies on page clustering described in section “[Page Clustering](#). ” Alternatively, in some domains users can feel appropriate to use a query by example search and ask to the system to look for pages similar to one sample page. Document image retrieval using layout similarity offers to users a way to retrieve relevant pages that was possible before only by manually browsing documents, by either interacting with physical works or dealing with online images on DLs [35].

In Fig. 7.9 we describe the organization of the most common modules considered in page retrieval. There are several similarities with the page classification overall architecture depicted in Fig. 7.7. In particular the page encoding in the corresponding feature vector is often analogous to the encoding made in page classification. The main differences from the user point of view are the lack of an explicit page labeling that is otherwise required for page classification training and the different response of the system when consulting the dataset. In contrast with

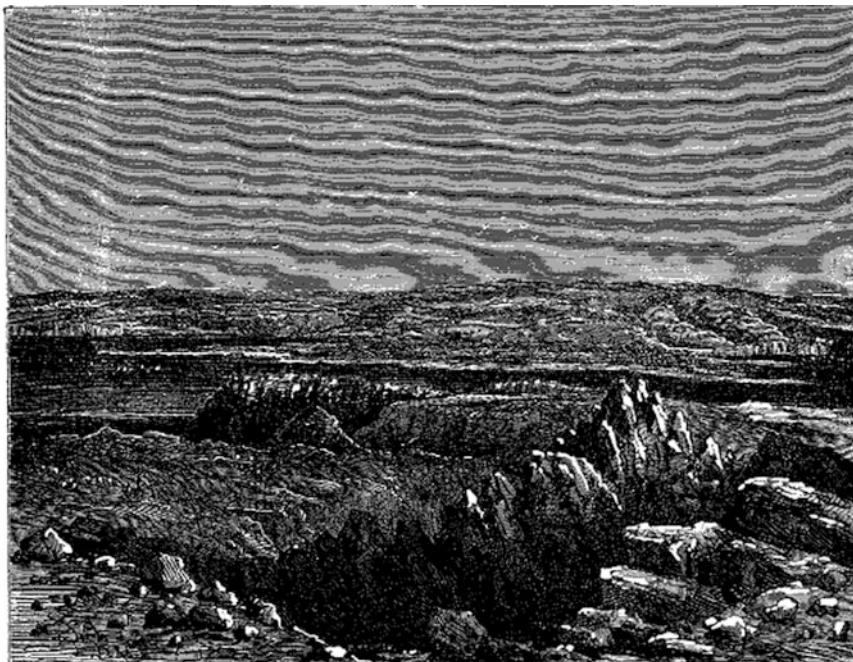


Fig. 385. — La mer Morte (vue prise de Masada).

L'expérience proposée par M. Aimé Girard est facile à exécuter en s'emparant des pratiques de Sétubal. Il est donc à désirer que ces essais s'accomplissent.

## CHAPITRE XV

### LES MARAIS SALANTS EN ESPAGNE, EN ITALIE, EN AUTRICHE, EN RUSSIE, ETC.

En décrivant la fabrication du sel extrait de la mer sur les côtes de la France et du Portugal, nous avons fait connaître tout ce qui se rattache à l'industrie salicicole. Dans les autres parties de l'Europe, cette industrie a moins d'importance, et emploie d'ailleurs les mêmes procédés de fabrication. Mais l'évaporation de l'eau de la mer à l'air

T. I.

libre est nécessairement liée au climat, à la température de chaque pays. On comprend donc que l'industrie qui nous occupe soit très-développée dans le midi de l'Europe et très-peu dans le nord.

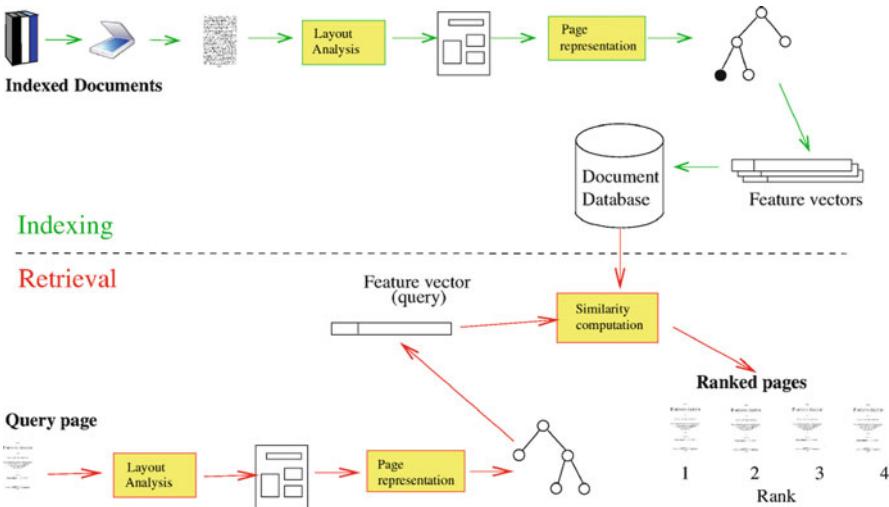
Après la France et le Portugal il faut citer l'Espagne et l'Italie, comme possédant un certain nombre de marais salants. Toutefois ces deux nations, malgré l'étendue de leur littoral marin, produisent beaucoup moins de sel que le Portugal.

Les marais salants de l'Espagne existent sur les côtes de la Méditerranée et aux îles Baléares; ceux de l'Italie, sur le littoral occidental de la Péninsule.

L'Autriche produit chaque année 75,000 tonnes de sel, dans des salines situées sur les bords de la mer Adriatique, en partie sur la côte de Dalmatie, en partie à Stagno,

81

**Fig. 7.8** Page with non-unique classes



**Fig. 7.9** General approach for page retrieval

**Table 7.3** Main features of some representative page retrieval methods. Data: type of documents addressed; Pages: number of page in the dataset; NC: number of classes; Measure: metric used to analyze the retrieval performance; Ref: cited paper

Data	Pages	NC	Measure	Ref.
Mixed documents	743	8	Receiver Operating Characteristic (ROC) curves	[24]
Article first page	537	4	Precision-recall plots	[6]
Historical journal and book	1,221	15	Top- $N$ precision	[11]
Forms	–	120	Top- $N$ precision	[21]
Mixed documents	2,550	18	Mean average precision	[27]
Born-digital documents	4,563	40	Rank error	[32]

page classification, the system provides to the user a ranked list of indexed pages rather than assigning a class to the query page.

To provide a synthetic view of the alternative methods, we compare some features of page retrieval systems in Table 7.3. The table also compares the various metrics used to analyze the retrieval performance. In the case of page retrieval, the approaches presented in the literature differ not only for the application domain addressed but also for the measure used to evaluate the proposed techniques.

Page similarity is computed in [24] by combining the probabilities of the unknown document  $d$  to belong to the class  $c_i$ . This is obtained by first computing the probability that each indexed document belongs to each class and then storing these probabilities in a vector  $d = \langle d_1, d_2, \dots, d_N \rangle$ . The probability is computed by considering the output of one SVM classifier trained for each class. An analogous vector is computed for the query document  $q$ . The similarity of the two documents is subsequently computed by considering the inner product between  $q$  and  $d$ .

The experiments are performed on a subset of the Girona Archives database (a collection of documents related to people passing the Spanish-French border from 1940 up to 1976) that contains 743 images divided in 8 different classes. The results are measured with Receiver Operating Characteristic (ROC) curves.

To adopt rectangular granulometries for page retrieval, Bagdanov and Worring [6] use each document image as a query and then rank the remaining documents according to the Euclidean distance between the size distributions of documents. Precision and recall are used to compare the performance of the different systems in this case.

The integration of global and structural features is proposed in [11]. Global features are based on the *book bounding box* while structural features are based on occurrences of tree patterns. The similarity score is computed by combining the similarity computed on the first set of features using the Euclidean distance with the similarity computed for the second group of values using the inner product. The experiments are performed on the Making of America (MOA) dataset (several issues of one nineteenth-century journal with a total of 608 pages) and on the Gallica one (613 pages from one book). The results are evaluated considering the Top- $N$  precision (the precision on the first  $N$  pages returned for each query). The best Top-10 precision for the whole database (that includes 15 classes) is 86.6 %.

Forms are represented by MXY trees and matched with a tree-edit distance in [21]. The tests are made on several datasets containing from 10 to 120 form types. In the case of 120 forms, the original samples are expanded by generating similar but different forms with geometrical modifications. In the experiments the precision when considering the top- $N$  results is computed with  $N$  varying from 2 to 25. This value varies from 25 to 100 % on the basis of the size of the results set.

In the retrieval system discussed in [27] instead of performing a complete document analysis, the text lines are detected and the page layout is described by means of the quadrilaterals generated by all pairs of text lines. The experiments are made on 2,555 documents split into 18 classes (e.g., text on one, two, or three columns). The database contains diverse documents including forms, academic papers, and handwritten pages in English and Arabic. To measure the performance two values are computed: the Mean Average Precision (MAP) for 100 documents and the Mean Average Normalized Rank (MANR). The MAP at 100 evaluates the ranking for the 100 top-ranked documents. ANR is defined as

$$ANR = \frac{1}{N \cdot N_w} \sum_{i=1}^{N_w} \left( R_i - \frac{N_w + 1}{2} \right),$$

where  $N$  is the number of documents in the set,  $N_w$  is the number of wanted documents in the set, and  $R_i$  is the rank of each wanted document in the set. ANR has a value of 0 when the wanted documents are all been sorted on top. The MAP measure on the previous dataset is comprised between 0.7 and 0.9.

In [32] documents are ranked by considering a page similarity computed with the cosine of the angle between two vectors representing the similarity of blocks in the page with respect to the set of Representative Blocks ( $RB$ ). The experiments are



**Fig. 7.10** Examples of page clusters

made on a collection of 845 technical documents accounting for 4,563 pages. The test is made considering 40 random pages as queries and then visually measuring the effectiveness of the ranking of the first 15 pages in the answer set. The computed rank error is defined as  $\sum_i \frac{|\text{ref}(D_i) - \text{rank}(D_i)|}{15}$ , where  $\text{ref}(D)$  and  $\text{rank}(D)$  are the position of document  $D$  in the reference ranking (manually annotated) and function

ranking, respectively. The best results reported in the paper provide on average 3.28 positions in the function ranking away from the reference.

---

## Page Clustering

One intermediate task between page classification and retrieval is page clustering. In this case the pages in a collection are not labeled by users and therefore it is not possible to build a supervised classifier. On the opposite, pages are clustered in an unsupervised way so as to facilitate page retrieval. One example of clustering (computed with the method described in [36]) is shown in Fig. 7.10 where each line contains pages grouped in the same cluster.

For instance, in [34] pages are represented by X-Y trees and one tree-edit distance algorithm is used together with the K-medoids clustering to group similar pages. The K-medoids algorithm is used rather than the classical K-means since it only needs the pairwise distance between pairs of objects (trees in this case).

Another clustering approach is proposed by Marinai et al. [36] and uses Self-Organizing Maps (SOM) to cluster pages represented by X-Y trees. The trees are encoded into a fixed-size representation by computing the occurrences of tree pattern as described in section “[Tree Distance](#).”

The fixed-size representation allows to compute the average of the patterns in clusters that is required by the SOM and the K-means clustering algorithms.

Self-Organizing Maps are used together with a zoning-based page representation in [45] to identify similar forms in the NIST Special Database 2 that contains 20 different form classes. Self-Organizing Maps are used to relax the need for labeled training samples. The SOM is used to first find clusters in the input data and then to identify each unknown form with one of the clusters.

---

## Conclusions

In this chapter we analyzed and compared various techniques for page classification and retrieval. The most important topic in both tasks is the choice of the approach used to represent the page layout. This representation is tightly related with the technique considered to compare the pages that is used by the page classification or retrieval algorithms. In particular, while pixel-based page representations are more appropriate when a fine-grained differentiation of the pages is required (used, for instance, in form classification), structural representations, e.g., based on trees or graphs, are more suitable when the generalization of the classifier/retrieval system is essential (for instance, in Digital Library applications).

Until recently in document image analysis, the page has been considered as the most common input to processing systems. One page can be easily converted into an image file both in the case of physical documents (where the conversion is made with digitization devices, such as scanners) and in the case of born-digital documents (where the image is generated by means of one suitable rendering

software). In the last few years with the advent of large-scale digitization projects and the availability of more powerful computational resources, the processing of whole books is becoming more and more common. In this case page classification can be used as a component of book processing systems.

On the other hand, when considering born-digital documents, the panorama of file formats is rapidly evolving in the last few years. Document distribution is nowadays in the large majority of cases delegated to the PDF format that is essentially a page-oriented format whose main purpose is the faithful representation of the page layout on a broad range of visualization and printing devices. However, the advent of e-book readers and tablet devices is pushing on the stage new formats, such as the e-pub format for e-books, that are essentially based on HTML and are intrinsically reflowable. With e-book readers the page is dynamically reflowed when reading the book and therefore it is more difficult to conceptually define the concept of *page*.

---

## Cross-References

- [Analysis of the Logical Layout of Documents](#)
  - [Image Based Retrieval and Keyword Spotting in Documents](#)
  - [Logo and Trademark Recognition](#)
  - [Page Segmentation Techniques in Document Analysis](#)
  - [Recognition of Tables and Forms](#)
- 

## References

1. Altamura O, Esposito F, Malerba D (2001) Transforming paper documents into XML format with WISDOM++. *Int J Doc Anal Recognit* 4(1):2–17
2. Appiani E, Cesarini F, Colla AM, Diligenti M, Gori M, Marinai S, Soda G (2001) Automatic document classification and indexing in high-volume applications. *Int J Doc Anal Recognit* 4(2):69–83
3. Arlandis J, Perez-Cortes J-C, Ungria E (2009) Identification of very similar filled-in forms with a reject option. In: Proceedings of the ICDAR, Barcelona, pp 246–250
4. Bagdanov AD, Worring M (2001) Fine-grained document genre classification using first order random graphs. In: Proceedings of the ICDAR, Seattle, pp 79–83
5. Bagdanov AD, Worring M (2003) First order Gaussian graphs for efficient structure classification. *Pattern Recognit* 36(3):1311–1324
6. Bagdanov AD, Worring M (2003) Multi-scale document description using rectangular granulometries. *Int J Doc Anal Recognit* 6:181–191
7. Baldi S, Marinai S, Soda G (2003) Using tree-grammars for training set expansion in page classification. In: Proceedings of the ICDAR, Edinburgh, pp 829–833
8. Cesarini F, Gori M, Marinai S, Soda G (1999) Structured document segmentation and representation by the modified X-Y tree. In: ICDAR, Bangalore, pp 563–566
9. Cesarini F, Lastri M, Marinai S, Soda G (2001) Encoding of modified X-Y trees for document classification. In: Proceedings of the ICDAR, Seattle, pp 1131–1136
10. Cesarini F, Lastri M, Marinai S, Soda G (2001) Page classification for meta-data extraction from digital collections. In: Mayr HC et al (eds) Database and expert systems applications. LNCS 2113. Springer, Berlin/New York, pp 82–91

11. Cesarini F, Marinai S, Soda G (2002) Retrieval by layout similarity of documents represented with MXY trees. In: Lopresti D, Hu J, Kashi R (eds) International workshop on document analysis systems, Princeton. LNCS 2423. Springer, pp 353–364
12. Chen N, Blostein D (2007) A survey of document image classification: problem statement, classifier architecture and performance evaluation. *Int J Doc Anal Recognit* 10(1):1–16
13. Chen F, Girgensohn A, Cooper M, Lu Y, Filby G (2012) Genre identification for office document search and browsing. *Int J Doc Anal Recognit* 15:167–182. doi:10.1007/s10032-011-0163-7
14. Chetverikov D, Liang J, Komubes J, Haralick RM (1996) Zone classification using texture features. In: International conference on pattern recognition, Vienna, pp 676–680
15. Collins-Thompson K, Nickolov R (2002) A clustering-based algorithm for automatic document separation. In: Proceedings of the SIGIR workshop on information retrieval and OCR, Tampere
16. Cullen JF, Hull JJ, Hart PE (1997) Document image database retrieval and browsing using texture analysis. In: Proceedings of the ICDAR, Ulm, pp 718–721
17. Dengel A (1993) Initial learning of document structure. In: Proceedings of the ICDAR, Tsukuba, pp 86–90
18. Dengel A, Dubiel F (1995) Clustering and classification of document structure-a machine learning approach. In: Proceedings of the ICDAR, Montreal, pp 587–591
19. Diligenti M, Frasconi P, Gori M (2003) Hidden Tree Markov models for document image classification. *IEEE Trans Pattern Anal Mach Intell* 25(4):519–523
20. Doermann D (1998) The indexing and retrieval of document images: a survey. *Comput Vis Image Underst* 70(3):287–298
21. Duygulu P, Atalay V (2002) A hierarchical representation of form documents for identification and retrieval. *Int J Doc Anal Recognit* 5(1):17–27
22. Ford G, Thoma GR (2003) Ground truth data for document image analysis. In: Proceedings of the symposium on document image understanding and technology, Greenbelt, pp 199–205
23. Gordo A, Valveny E (2009) A rotation invariant page layout descriptor for document classification and retrieval. In: Proceedings of the ICDAR, Barcelona, pp 481–485
24. Gordo A, Gibert J, Valveny E, Rusiñol M (2010) A kernel-based approach to document retrieval. In: International workshop on document analysis systems, Boston, pp 377–384
25. Hu J, Kashi R, Wilfong G (1999) Document image layout comparison and classification. In: Proceedings of the ICDAR, Bangalore, pp 285–288
26. Hu J, Kashi R, Wilfong G (2000) Comparison and classification of documents based on layout similarity. *Inf Retr* 2:227–243
27. Huang M, DeMenthon D, Doermann D, Golebiowski L (2005) Document ranking by layout relevance. In: Proceedings of the ICDAR, Seoul, pp 362–366
28. Indermuhle E, Bunke H, Shafait F, Breuel T (2010) Text versus non-text distinction in online handwritten documents. In: SAC, Sierre, pp 3–7
29. Ishitani Y (2000) Flexible and robust model matching based on association graph for form image understanding. *Pattern Anal Appl* 3(2):104–119
30. Jain AK, Liu J (2000) Image-based form document retrieval. *Pattern Recognit* 33:503–513
31. Kochi T, Saitoh T (1999) User-defined template for identifying document type and extracting information from documents. In: ICDAR, Bangalore, pp 127–130
32. Lecerf L, Chidlovskii B (2010) Scalable indexing for layout based document retrieval and ranking. ACM Symposium on Applied Computing, Sierre, pp 28–32
33. Lin JY, Lee C-W, Chen Z (1996) Identification of business forms using relationships between adjacency frames. *MVA* 9(2):56–64
34. Mao S, Nie L, Thoma GR (2005) Unsupervised style classification of document page images. IEEE International Conference on Image Processing, Genoa, pp 510–513
35. Marinai S (2006) A survey of document image retrieval in digital libraries. In: 9th colloque international francophone sur l'Écrit et le document, Fribourg, pp 193–198
36. Marinai S, Marino E, Soda G (2006) Tree clustering for layout-based document image retrieval. In: Proceedings of the international workshop on document image analysis for libraries 2006, Lyon, pp 243–253

37. Marinai S, Marino E, Soda G (2010) Table of contents recognition for converting PDF documents in e-book formats. In: Proceedings of the 10th ACM symposium on document engineering (DocEng'10), Manchester. New York, pp 73–76
38. Marinai S, Miotti B, Soda G (2011) Digital libraries and document image retrieval techniques: a survey. In: Biba M, Xhafa F (eds) Learning structure and schemas from documents. Volume 375 of studies in computational intelligence. Springer, Berlin/Heidelberg, pp 181–204
39. Peng H, Long F, Chi Z, Siu W-C (2001) Document image template matching based on component block list. *PRL* 22:1033–1042
40. Peng H, Long F, Chi Z (2003) Document image recognition based on template matching of component block projections. *IEEE Trans Pattern Anal Mach Intell* 25(9):1188–1192
41. Perea I, López D (2004) Syntactic modeling and recognition of document image. In: SSPR&SPR, Lisbon, pp 416–424
42. Qi X, Davison BD (2009) Web page classification: features and algorithms. *ACM Comput Surv* 41:12:1–12:31
43. Sebastiani F (2002) Machine learning in automated text categorization. *ACM Comput Surv* 34:1–47
44. Shih FY, Chen SS (1996) Adaptive document block segmentation and classification. *IEEE Trans SMC* 26(5):797–802
45. Shin C, Doermann DS, Rosenfeld A (2001) Classification of document pages using structure-based features. *Int J Doc Anal Recognit* 3(4):232–247
46. Takama Y, Mitsuhashi N (2005) Visual similarity comparison for web page retrieval. In: IEEE/WIC/ACM international conference on web intelligence (WI 2005), Compiegne, pp 301–304
47. Taylor SL, Fritzson R, Pastor JA (1992) Extraction of data from preprinted forms. *MVA* 5(5):211–222
48. Taylor SL, Lipshutz M, Nilson RW (1995) Classification and functional decomposition of business documents. In: ICDAR 95, Montreal, pp 563–566
49. Tzacheva A, El-Sonbaty Y, El-Kwae EA (2002) Document image matching using a maximal grid approach. *Document Recognition and Retrieval IX*, San Jose, pp 121–128
50. van Beusekom J, Keysers D, Shafait F, Breuel TM (2006) Distance measures for layout-based document image retrieval. In: Proceedings of the international workshop on document image analysis for libraries 2006, Lyon, pp 232–242
51. Wang JT-L, Zhang K, Jeong K, Shasha D (1994) A system for approximate tree matching. *IEEE Trans Knowl Data Eng* 6(4):559–571
52. Wang Y, Phillips IT, Haralick RM (2006) Document zone content classification and its performance evaluation. *Pattern Recognit* 39:57–73
53. Wei C-S, Liu Q, Wang JT-L, Ng PA (1997) Knowledge discovering for document classification using tree matching in TEXPROS. *Inf Sci* 100(1–4):255–310
54. Zhang K, Shasha D (1989) Simple fast algorithms for the editing distance between trees and related problems. *SIAM J Comput* 18(6):1245–1262

## Further Reading

Document image classification has been extensively surveyed by Chen and Blostein in [12] where one comprehensive comparison of the different applications and techniques adopted for page classification is provided. On the side of document image retrieval, one classical survey on the topic including both text-retrieval and layout-based approaches is [20]. One recent survey related to applications in Digital Libraries can be found in [38].

---

## Part C

# Text Recognition

Text recognition can be thought of as the patriarch of document image analysis. The earliest patents and reading machines were all focused on reading individual characters, primarily machine-printed characters in the English language. Thus it is no surprise that this part of the handbook is the largest, with seven chapters. It also shows how the field has drastically changed over the past four decades in individual chapters on essential sub-problems such as segmentation and language identification, while problems such as word recognition, handwriting recognition, and recognition of foreign languages have also evolved to the point where they are considered their own disciplines in the field.

For readers that are interested in any of these topics, the chapters in this part should not necessarily be seen as independent. There is a great deal of overlapping and the techniques that are prominent in machine print recognition, for example, may in fact be very relevant to related issues in Asian or Middle Eastern recognition. Readers are encouraged to at least skim each chapter to get a better picture of issues before delving deeper into a particular chapter.

One all-encompassing chapter is that of text segmentation contributed by Nicola Nobile and Ching Suen ([► Chap. 8](#) (Text Segmentation for Document Recognition)). It has been widely claimed by commercial OCR vendors that over 80 % of the machine print errors that occur in OCR systems are due to segmentation. Noise tends to cause characters to touch or break, and lines to overlap. The chapter addresses these basic problems, but then considers problems unique to historic and handwritten material.

Along with broadening the field beyond machine print recognition, document image analysis has expanded to deal with many languages, and writing scripts and the plethora of fonts brought on by advanced desktop publishing. [► Chapter 9](#) (Language, Script and Font Recognition) was contributed by Umapada Pal. It highlights the challenges and solutions provided by current approaches, and provides a framework for continuing to address their problems.

[► Chapters 10](#) (Machine-Printed Character Recognition), [► 11](#) (Handprinted Character and Word Recognition), and [► 12](#) (Continuous Handwritten Script Recognition) focus on the core of text recognition. [► Chapter 10](#) (Machine-Printed

Character Recognition) contributed by Premkuram Natarajan provides historical perspective on machine-printed character recognition. This was one of the first areas to be “commercialized” and while it is seen as a solved problem under ideal circumstances, research is still active on the more challenging aspects of noisy and poor quality data. Venu Govindaraju and Sergey Tulyakov address the complementary problem of handprinted text in ►Chap. 11 (Handprinted Character and Word Recognition). The problem is often seen as one step harder than machine print, but as you will read in this chapter, the techniques required are fundamentally different on many levels. ►Chapter 12 (Continuous Handwritten Script Recognition) authored by Horst Bunke and Volkmar Frinken focuses on a problem more commonly referred to as cursive script. The problem introduced fundamental challenges in dealing with different writing styles and the inability to perform accurate segmentation. Readers will appreciate the challenges of automating humans’ advanced ability to recognize content even when a majority of the individual characters are illegible.

Finally, Part C concludes with two chapters that focus on specific regions of the world, in part because of the fundamental differences in the scripts and character sets in both machine and handwritten material. In ►Chap. 13 (Middle Eastern Character Recognition), Abdel Belaïd and Mohamed Imran Razzak address problems arising with the connected scripts of the Middle East. Even for machine print, the connectedness of characters and the variations that occur, based on linguistic context, have forced the community to expand the thinking when it comes to both segmentation and classification. ►Chapter 14 (Asian Character Recognition) is authored by Ranga Setlur and Zhixin Shi. In addition to segmentation differences with other languages, languages such as Chinese have characters derived from ideograms, and typically have a character set that are orders of magnitude larger than other languages. This provides a unique set of challenges for classification that must be addressed by systems moving forward.

---

# Text Segmentation for Document Recognition

8

Nicola Nobile and Ching Y. Suen

## Contents

Introduction.....	258
Zone and Line Segmentation.....	259
Challenges.....	260
Noisy Documents.....	260
Historical Documents.....	262
Line Segmentation.....	263
Segmentation for Optical Character Recognition (OCR).....	272
Challenges.....	273
Touching Characters.....	273
Broken Characters.....	274
Lack of Baseline Information.....	275
Typefaces.....	276
Touching Italic Characters.....	278
Segmentation of Degraded Characters.....	282
Segmentation of Mathematical Expressions.....	283
Conclusion.....	286
Cross-References.....	288
References.....	288
Further Reading.....	290

---

N. Nobile

Centre for Pattern Recognition and Machine Intelligence (CENPARMI), Concordia University,  
Montréal, QC, Canada

e-mail: [nicola@cenparmi.concordia.ca](mailto:nicola@cenparmi.concordia.ca)

C.Y. Suen

Department of Computer Science and Software Engineering, Centre for Pattern Recognition and  
Machine Intelligence (CENPARMI), Concordia University, Montréal, QC, Canada

e-mail: [suen@encs.concordia.ca](mailto:suen@encs.concordia.ca); [suen@cenparmi.concordia.ca](mailto:suen@cenparmi.concordia.ca)

**Abstract**

Document segmentation is the process of dividing a document (handwritten or printed) into its base components (lines, words, characters). Once the zones (text and non-text) have been identified, the segmentation of the text elements can begin. Several challenges exist which need to be worked out in order to segment the elements correctly. For line segmentation, touching, broken, or overlapping text lines frequently occur. Handwritten documents have the additional challenge of curvilinear lines. Once a line has been segmented, it is processed to further segment it into characters. Similar problems of touching and broken elements exist for characters.

An added level of complexity exists since documents have a degree of noise which can come from scanning, photocopying, or from physical damage. Historical documents have some amount of degradation to them. In addition, variation of typefaces, for printed text, and styles for handwritten text bring new difficulties for segmentation and recognition algorithms.

This chapter contains descriptions of some methodologies, presented from recent research, that propose solutions that overcome these obstacles. Line segmentation solutions include horizontal projection, region growth techniques, probability density, and the level set method as possible, albeit partial, solutions. A method of angle stepping to detect angles for slanted lines is presented. Locating the boundaries of characters in historical, degraded ancient documents employs multi-level classifiers, and a level set active contour scheme as a possible solution. Mathematical expressions are generally more complex since the layout does not follow standard and typical text blocks. Lines can be composed of split sections (numerator and denominator), can have symbols spanning and overlapping other elements, and contain a higher concentration of superscript and subscript characters than regular text lines. Template matching is described as a partial solution to segment these characters.

The methods described here apply to both printed and handwritten. They have been tested on Latin-based scripts as well as Arabic, Dari, Farsi, Pashto, and Urdu.

**Keywords**

Anisotropic probability density • Broken • Characters • Contour • Curvilinear • Degradation • Features • Handwriting • Histogram • Historical • Level set • Lines • Line slant • Normalization • OCR • Pixel density • Projection • Recognition • Region growth • Segmentation • Template matching • Touching • Typefaces

---

**Introduction**

Document segmentation has been a difficult problem to solve through automation. The problem itself has evolved since its inception. Where initially, the problem was to recognize isolated printed words and characters, it has modernized itself over the

years in order to keep up with user needs. In later years, the research in this field became more complex. The progression continued by finding words and characters [29] in a line of text, then finding all text lines in a given printed document. As documents became more complex, it was then necessary to identify the regions of a document which are text and which are non-text – a common layout for newspapers and magazines.

Modern demands push the field into finding solutions for new problems. Such is the case with the Sarbanes-Oxley Act. This law requires every public company in the United States to electronically store all business records. Storing these document images is the easy part – searching them is much more difficult. These documents can have any layout ranging from standard business letters, receipts, purchase forms, and accounting tables to name a few, to charts, tables, and figures containing important textual information which need to be identified.

To handle this wide range of layouts, document processing systems [25] are required to identify the location of the text blocks and separate them from the non-text blocks. Of these text blocks, the lines of text need to be separated. The lines may be of different orientations from each other or a line may even be curved.

More recent applications include handwritten [15] documents usually co-existing with printed text in a document. Handwritten line segmentation and OCR are much more difficult [19–22, 28, 35] due to the greater diversity of handwriting styles compared with printed texts. Furthermore, machine-printed text lines are usually laid out straight and the skew and/or rotated texts are minimal, usually caused when it is – scanned or photocopied where the document is not aligned with the glass edge of the recording device. Handwritten documents are influenced mostly by proper line segmentation. The latest trend is geared towards page segmentation and OCR of Middle-Eastern languages such as Arabic [11, 30, 32], Farsi [12], and Urdu [23].

Other recent applications which use text segmentation and OCR technology are search engines used to search scanned documents. This is most useful for historical documents which would take too long to manually create an editable text version.

Word spotting is a relatively new field. The goal is to find specific words in documents. Depending on the system, accuracy relies greatly on the OCR [24] performance. Data mining can be used to find similar documents based on a user's reading patterns. Each has its own unique challenges and obstacles in OCR.

The remaining of this chapter discusses the latest research in line segmentation (section “[Zone and Line Segmentation](#)”) and OCR will be discussed in section “[Segmentation for Optical Character Recognition](#).”

---

## **Zone and Line Segmentation**

Document segmentation involves locating and separating the text from the non-text regions of a page. In ►[Chap. 5](#) (Page Segmentation Techniques in Document Analysis), the reader can find more information on document segmentation. In ►[Chap. 6](#) (Analysis of the Logical Layout of Documents), the reader can find details on how text blocks are located in complex document layouts.

Once a text region has been identified, it is passed to a line segmentation module which will locate and, if needed, separate the lines in that region. The scope of this chapter involves the processing of a region of text which contains no images and contains only text. The text may be arranged in an unnatural manner or it may contain noise. It is the job of the line segmentation module to locate the lines which have been naturally written and send this information to next module for further processing. Breaking down the problem from large objects (text and non-text regions) to smaller objects (characters or connected components) allows each module in the process to deal with fewer items, thereby speeding up the process and accuracy. However, any errors that occur in previous modules will affect subsequent tasks that follow. This has a negative effect on accuracy of all modules, be it OCR or word recognition.

## Challenges

Documents come in a variety of layouts, printing media, and languages. Older documents may have been scanned at a lower resolution and may show signs of degradation. Similarly, historical documents usually are in poor condition, so high resolution scanning would not improve the quality.

Regardless of the layout, media, or language, all digitized documents contain a common feature – noise. The only difference is the degree of noise present in a digital document. Another common challenge to overcome is the flow of the text. Ideally, all text would be aligned in a straight organized direction and well-spaced. This is usually not the case since it is observed that most scanned documents are rotated and contain some skew. Furthermore, consecutive text lines can be overlapping or worse, touch each other. Touching lines should be separated before further processing can be applied. Other common hurdles include broken lines, lines that do not contain baseline information, curvilinear text, and/or touching/broken characters.

Algorithms were developed to address these issues and a few of them are discussed in this section. Section “[Noisy Documents](#)” discusses and describes the types of noise in documents and what can be done to clean it. In section “[Historical Documents](#),” a description of noise specific to historical documents is presented. In section “[Line Segmentation](#),” a few line segmentation techniques are explained.

## Noisy Documents

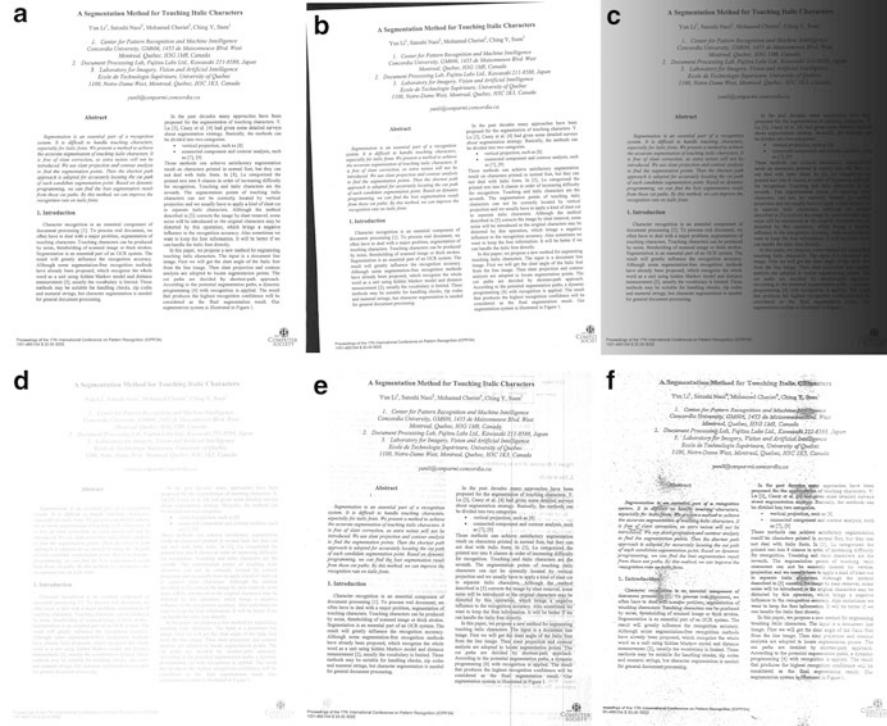
Noise is the appearance of objects in a document image which do not belong to or are not part of the original document. These objects can be the result of poor paper quality, digitizing, or a physically damaged document (e.g., handwriting over text, coffee spills, creases, scratches).

Several types of noise can appear in an image but the most common type is “salt and pepper” noise which contains randomly white (salt) and black (pepper) pixels appearing throughout the image. Some causes of this kind of noise can come from dead pixels on the capture device, errors from converting analog to digital,

**a** Character recognition is document processing [1]. T often have to deal with a ma touching characters. Touchin by noise, thresholding of sca Segmentation is an essential result will greatly influence

**b** Character recognition is document processing [1]. T often have to deal with a ma touching characters. Touchin by noise, thresholding of sca Segmentation is an essential result will greatly influence

**Fig. 8.1** Salt and pepper noise example (before (a) and after (b) cleaning)



**Fig. 8.2** Common photocopying artifacts; (a) original, (b) black borders, (c) uneven toner, (d) too Light, (e) too dark, (f) 10th generation copy

from quantization, using long shutter speeds or a high ISO in digital cameras, and transmission errors. This noise is commonly corrected by using a median filter. Figure 8.1 shows an example of an image containing salt and pepper noise before and after cleaning.

Noise can be produced by the way a document was handled. For example, poor photocopying is one source that results in noisy and degraded images. Figure 8.2 displays some common problems resulting in poor photocopying.

In Fig. 8.2b, the document image contains black borders around the perimeter. This usually occurs when the document is slanted and the cover of the scanner was left open during photocopying, or the photocopier has a dark background under the lid when it is closed. Although these borders do not usually interfere with the text of the document, the segmentation preprocessing must still identify these unwanted objects and remove them.

Another common problem that results from photocopying (or sometimes printing) is the uneven distribution of the toner which leaves part of the document lighter than the rest. Special care is needed to identify this problem and to correct it. If not identified, the lighter part of the document may disappear during the binarization process if a fixed threshold is used. This can be seen in Fig. 8.2c.

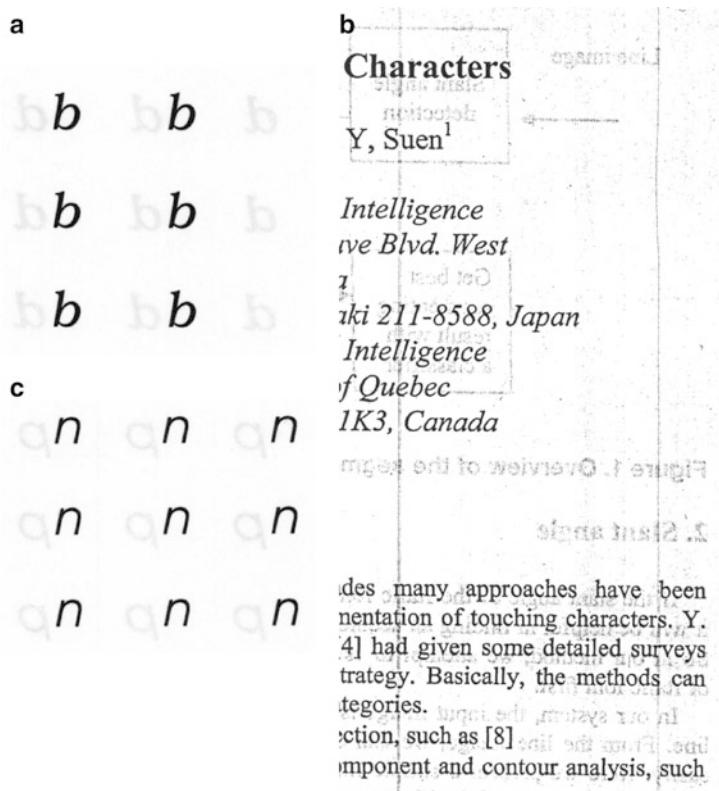
Some photocopiers have the option to make the copy lighter or darker than the original. Making the document too light may cause some characters to be too faint to recognize. Making the page darker may introduce a dark scanning “texture” as seen in Fig. 8.2e which interferes and overlaps the text giving the segmentation a difficult challenge to correct. Finally, another common routine people do is to make copies of copies. This can be repeated several times with each new generation inheriting the previous generation problems (i.e., noise) while introducing new unwanted artifacts. Furthermore, since a photocopy is not 100 % identical to the original, the quality of the document degrades with each new generation.

Figure 8.2f shows an example of a simple document which went through ten copy generations on the same photocopier. This shows the additive characteristic of nth-generation copying. Flaws in the photocopier(s) will show in the copy in addition to the previous flawed versions. Other copier functions can affect the quality such as resizing and resolution. Fax machines usually print in a lower resolution (usually between 100 and 200 DPI). Additionally, dirty copier or scanner glass or damaged capture devices will lead to noisier images.

The process of scanning itself can sometimes lead to degraded documents regardless of the quality of the document. Some examples of unwanted scanning items include scanning at low resolution, with a dirty or defective scanner, rotated or upside-down scans, and ghost images. Ghost images appear when the scanned page is double-sided or there is another page behind the scanned page during scanning. If the paper is thin enough, or the scanner lamp is too bright, ink from the second page will faintly appear on the scanned image. Also, some newer scanners can scan in duplex mode in such a way that both pages are scanned at the same time. This leads to the second page ink to “bleed” through the scan and appear in the image. Figure 8.3 shows examples of “ghost images.”

## Historical Documents

The current trend for digital libraries and online publishers is to make their books and magazines available online in digital form. This is not a problem with recent documents that have been written using word processors. However, old historical documents need to be scanned and sent through an OCR. The challenges with old



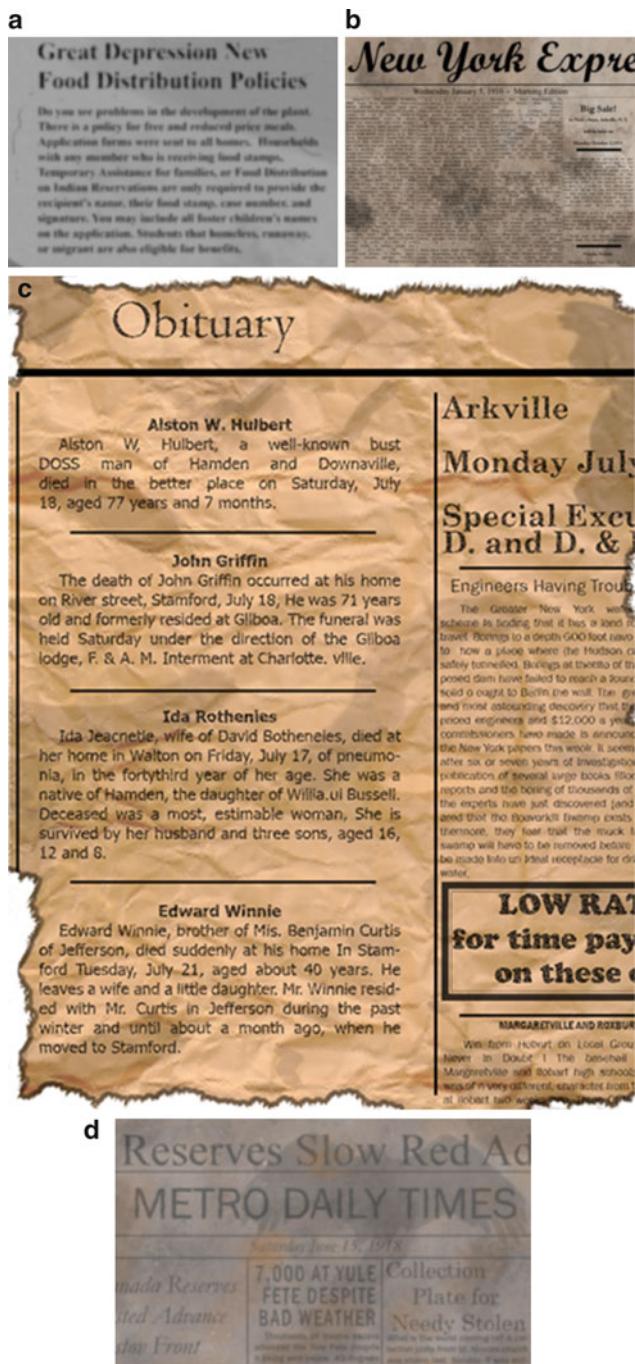
**Fig. 8.3** Scanning ghost (show-through) examples

documents are usually related to the quality of the paper. Aged paper shows signs of fading, degrading, destruction from infestation, or degrading from environmental elements such as heat, oxygen, sunlight, humidity, and general paper decay. Any of these issues result in a poor image quality and therefore making it more difficult to segment characters. Figure 8.4 shows a few examples of aged-looking documents.

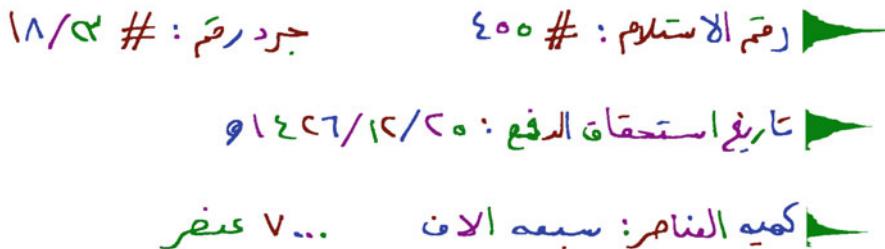
## Line Segmentation

Given a digital text image, a line segmentation algorithm locates and extracts each text line from the image for further processing. Several types of software, such as OCR, word segmentation, and word spotting, make use of individual lines of text. The challenges for line segmentation are listed as follows:

1. Overlapping line boundaries
2. Touching lines
3. Broken lines
4. Lack of baseline information



**Fig. 8.4** Examples of old historical books/documents



**Fig. 8.5** Highlighted connected components of Arabic text (with horizontal projection)

5. Curvilinear text
6. Piecewise linear text
7. Touching characters and words within a line

Several approaches exist to solve these problems and they fall into one of two categories – top-down and bottom-up [1]. In a top-down approach, a line segmentation algorithm uses large features of a line in order to determine its boundaries.

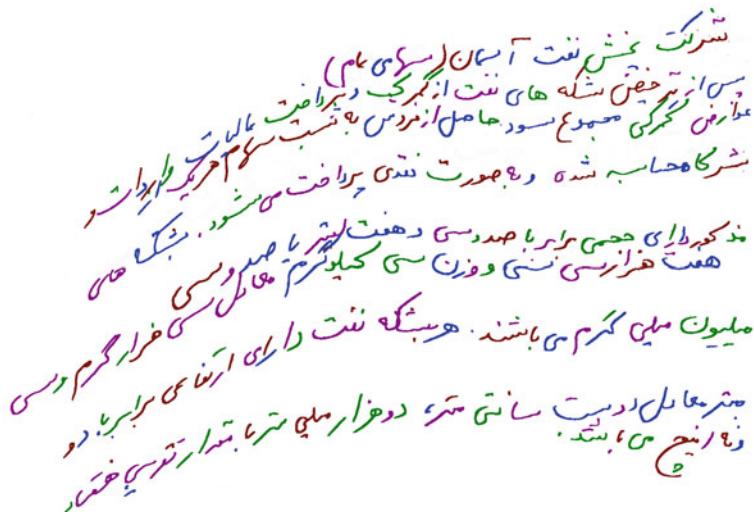
Bottom-up starts from the smallest element of a document image – the pixel. By grouping touching pixels, connected components are generated. In Fig. 8.5, a sample Arabic handwritten text is shown with the different connected components displayed in various colors.

Researchers in image processing use connected components as the building blocks for analysis. In the case of text segmentation, they are used to locate characters and symbols having a specific relationship – they could be components on the same line or belonging to a complex mathematical formula (spanning several lines and containing symbols).

Segmenting the lines in Fig. 8.5 is a rather easy task. To the right of the three lines are the horizontal projections (in green). The gaps between the lines have no value in the projection and therefore can be used to determine where one line begins and another ends.

However, not all texts are written as neatly as the sample shown in Fig. 8.5. Most look like the example in Fig. 8.6 where handwritten text may begin in a straight line but taper off towards the end (right to left) [18]. This may be due to writers “drifting” off or it can be from a photocopy of the inner binding of a book. Figure 8.6 shows examples of overlapping lines as well as some touching lines.

Other elements that can add to the difficulties are broken lines, lack of baseline information, and unique language features. Languages from different regions make some techniques, which are successful with one family of languages, less effective on other types of languages. For example, using connected components on printed English text can help identify individual isolated characters in a line of text. However, for Arabic printed text, neighboring characters are frequently touching and not isolated. Additionally, for Arabic, the character shape/style changes depending on where in a word the character appears (beginning, middle, or end).



**Fig. 8.6** Highlighted connected components in curvilinear, overlapping, and touching lines of handwritten Arabic text

**COSUVWXZ**  
**cosuvwxz**

**Fig. 8.7** English printed characters having same upper and lower case shapes

Lack of baseline information occurs when all the characters on the line have the same height and each of those characters begin and end at the same  $y$ -positions. Figure 8.7 shows the English printed characters which fall into this category.

When this situation occurs, the case of the characters cannot be determined – even though the letters are correct. This can be a problem later on when the segmented line is passed to an OCR.

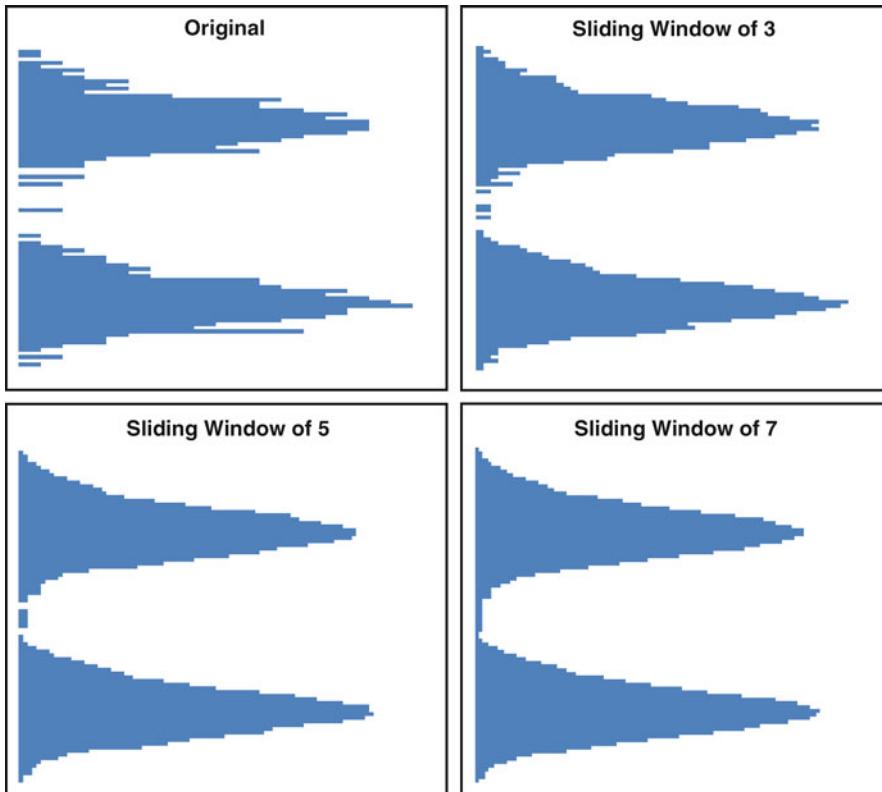
Several methods exist to counter these challenges in order to correctly segment lines from a document. The following lists some techniques used:

1. Horizontal projection
2. Region growth techniques
3. Probability density
4. Level set method

Each has its own advantages and disadvantages and they are discussed in greater detail in the remaining of this chapter.

### Horizontal Projection

Using a horizontal projection is probably the easiest way to segment lines [16]. A horizontal projection is a histogram, or count, of all the foreground pixels in a document for each horizontal scan line. Figure 8.5 shows horizontal projections to the right of the text. This method is ideal when the lines are well separated and



**Fig. 8.8** Horizontal projection smoothing with different window sizes

are not overlapping or touching each other. In this situation, the projection will contain non-zero values for scan lines crossing a line of text. Gaps between lines will have values of zero thereby effectively marking the start and ends of the text line boundaries.

Small values may be present in the gaps if some noise is present. In this case, a threshold value or a smoothing algorithm can be applied to reduce the noise. Smoothing can also be used to fill in broken lines.

Figure 8.8 shows the horizontal projection for a short two line text block. The original contains some noise between the lines and the projection within each line is not smooth. There are too many high peak-to-valley values for adjacent scan lines. This is sometimes an indicator of broken text or some special noise, such as a cross-out [17], is present. When a sliding window of 3 pixels is applied, the peaks fall and the “holes” are filled in. A sliding window of seven eliminates all the influence of the gaps and noise within the lines. The peaks between the lines, which were caused by noise, have been reduced to small enough values which can be detected by a small threshold value.

The horizontal projection is best suited for printed text. Printed text is more consistent when it comes to maintaining gaps between the lines and for keeping the lines horizontally straight. Handwritten text can have curvilinear, touching, and overlapping text as seen in the example in Fig. 8.6.

## Region Growth Techniques

Region growth techniques are methods which group neighboring pixels in an image into subregions. Pixels in a subregion share a common feature that no other pixel, in other subregions, has. A feature can be anything from the color of the pixel, the intensity, or texture, for example.

Several methods exist but all should obey the following rules:

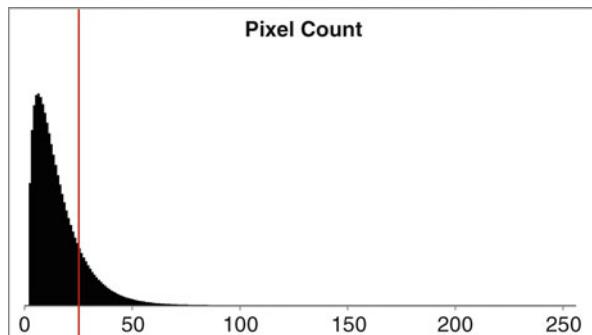
1. Every pixel in the document must belong to a subregion.
2. Pixels in a subregion must be spatially connected with one another.
3. No two subregions have a common pixel member. That is, each pixel can belong to only one subregion.
4. Each pixel in a subregion must satisfy a logical predicate based on a feature. For example, each pixel color.
5. Applying a predicate function from one subregion to the pixels of another subregion should return false (or the empty set). That is, pixels not belonging to the first subregion cannot satisfy its predicate function.

The process of building the subregions is a bottom-up approach. It begins with “seedpoints.” A seedpoint is an initial point, or pixel, in a digital document from which to begin to build a subregion from. More than 1 pixel can be added to a subregion’s set of seedpoints. From a seedpoint, the neighboring pixels are examined and are added to the subregion based on the criterion established for subregion pixel membership, and they do not belong to any other subregion’s set of pixels. Neighboring pixels are pixels which are immediately adjacent – either by 4-connectivity or 8-connectivity connection. The next step continues by following the same membership confirmation of neighboring pixels based around the newly added pixels from the previous iteration. This iterative process ends when no new pixels have been added to the set.

Choosing initial seedpoints is an important step and can be facilitated by using overall document statistics. For example, in our case, if text is needed to be removed from a background [27], the histogram of a grayscale image can be viewed and a value can be chosen where the foreground pixels are prevalent. The criterion for this subregion could be if the pixel grayscale value is less than a certain value. This value is shown as the red line in Fig. 8.9.

Of course this is not limited to just pixel counts. An alternative could be to assign a pixel on the edge of an object as our seedpoint and add neighboring pixels which are also edge pixels. This would give a set of points which form the outline of the object in which the initial seedpoint resides on. Or a criterion based on texture can be used. Applications using texture-based criteria are mainly used for image segmentation such as locating contaminated areas in meat products, defects in lumber moving through a sawmill, or cancerous regions in cell tissue, but can also be used for text segmentation.

**Fig. 8.9** Sample text document pixel intensity histogram



Note that choosing the initial seedpoints is important. This method will only find the pixels in the document which satisfy the criterion and are connected to the initial seedpoints. Points which meet the subregion's criterion but cannot be linked to any of the seedpoints will be neglected. In our example above, at least one point from each character in the text document will need to be added to our initial set of seedpoints in order to have full coverage. However, the advantage is that most will be filtered out and will not be included in the segmentation result.

These methods of segmentation rely heavily on a good choice of initial seedpoints. In addition, this method is highly computationally expensive.

### Probability Density

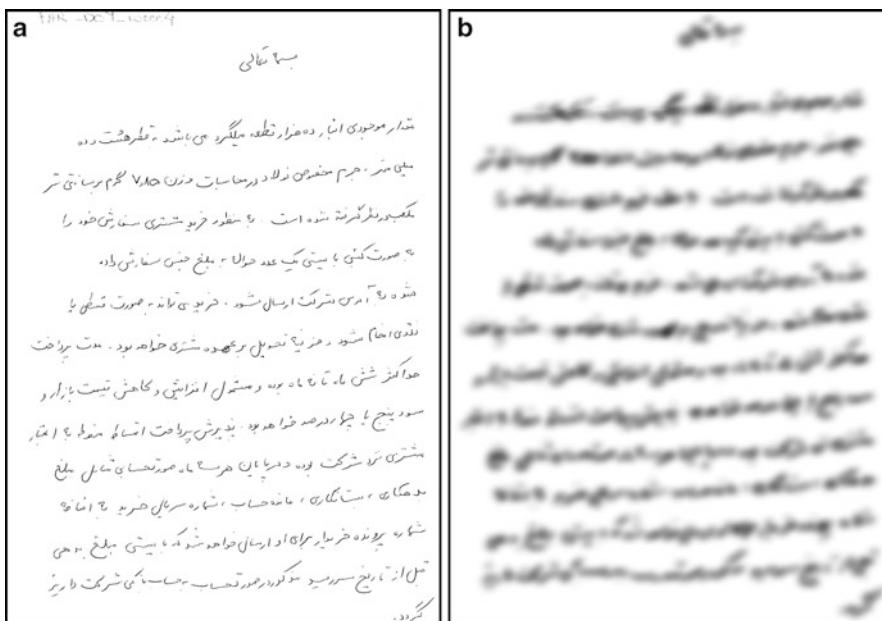
A digital image can be considered as a two-dimensional array of pixel intensities or colors. From this array, algorithms can be applied to obtain features and statistical information [14] (such as a projection, density, and centroids). However, obtaining these features can be a time-consuming procedure and may not provide the best information for line segmentation.

Probability densities themselves can be considered a feature of a document. It is computed once and information can be obtained directly or they can be used for processing by other methodologies.

Peaks on a probability map of a text document correspond to the lines of text. Valleys correspond to the boundaries between consecutive text lines. In general, the probability density represents the distribution of the lines in a document. The values are higher (denser) within text lines and lower (thinner) in the spaces between the lines and in the margins.

Probability densities are usually continuous for most applications. However, for document processing, the functions are not known. Therefore, a discrete estimate is performed. To generate a probability density for a document, a value is computed using an anisotropic Gaussian kernel defined as [2, 37]:

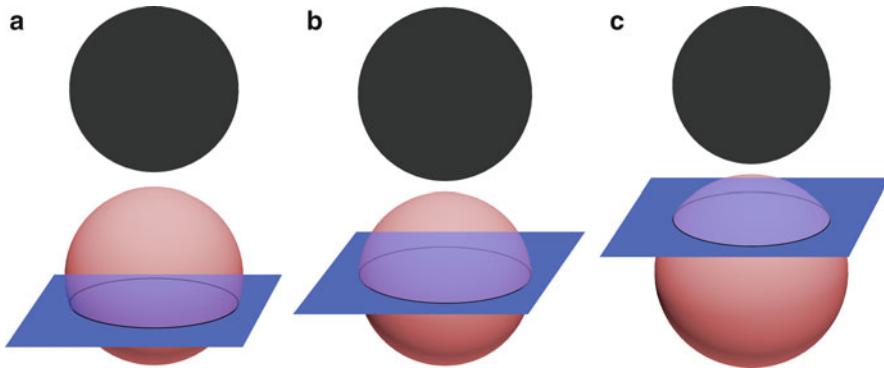
$$f(x, y) = Ae^{-\left[\frac{(x - b_x)^2}{2\sigma_x^2} + \frac{(y - b_y)^2}{2\sigma_y^2}\right]} \quad (8.1)$$



**Fig. 8.10** (a) Original handwritten document, (b) probability density using anisotropic Gaussian probability density function

where  $b_x$  is the x-axis shift and  $b_y$  is the shift for the y-axis and  $\sigma_x$  and  $\sigma_y$  are the standard deviation-curve spread parameters in the x and y-directions, respectively.  $A$  is the amplitude of the function. Note that if  $\sigma_x = \sigma_y$ , then our Gaussian function will be isotropic. Having an isotropic Gaussian function is undesirable in the case of text line segmentation because text, printed or handwritten, is usually written horizontally with a little skew. This is the case with most languages. Having a large vertical spread will cause text lines, which are close to each other, to merge. Instead, it is preferred to use an anisotropic spread where the height (y-axis) is shorter and the horizontal reach (along the x-axis) is longer. This reduces the chances of consecutive lines being merged and increases the likelihood that pixels on consecutive characters, on the same line, will have a large influence on the probability density.

A new grayscale image is created from applying the Gaussian kernel. This image represents the intensity around the black pixels by expanding the area of influence of a pixel based on all the black pixels in its immediate neighborhood. The more black pixels in a neighborhood, the darker the grayscale image will be in that area. The surrounding pixel intensities will depend on the distance to the original pixel and for this reason, the Gaussian kernel generates a grayscale image representation with intensity values ranging from 0 to 255. The grayscale image is a representation of the probabilities that corresponding pixels in the original document belong to a text line or not. Figure 8.10 shows an original handwritten document and the grayscale



**Fig. 8.11** Level set cross sections

image representing the probability density when applying an anisotropic Gaussian probability function.

It can be seen that the text line locations are more prevalent in the probability density image in Fig. 8.10. In addition, noise and pixels in the margins tend to disappear since they are isolated from the text and, therefore, have a little support from neighboring white pixels.

### Level Set Method

The level set method [31] was developed by mathematicians Stanley Osher and James Sethian in 1988. The method was conceived to track moving objects and shapes in consecutive video frames and has been extended to image segmentation. It is used to separate foreground objects from the background. Visually, a level set is a cross section (in the  $xy$ -plane) of a two-dimensional object projected into a three-dimensional model as seen in Fig. 8.11. The corresponding three-dimensional graph is shown below each shape and is called the level set function defining the shape. The three-dimensional surface represents the motion of a two-dimensional shape.

The boundary of the shape is called the zero level set. The level set function is zero at the boundary points. The shape contains the points which are inside the boundary – where the level set function is positive. Any point inside the two-dimensional shape generates a positive level set function value. Any point outside will produce negative level set function values. As time advances, the shape can change form or topology as seen in Fig. 8.11b, c. Keeping track of the transformations of the original shape requires a great deal of work mostly for times when the shapes divide or join. Using the corresponding level set is much easier since it now only needs to track the zero level set (the cross section in the  $xy$ -plane).

The level set method can be used to separate the text regions from the background for text line segmentation. This is an iterative process where an initial zero level set is evolved according to a partial differential equation. The direction of growth

of the boundary over time is guided by its partial derivatives and an external vector field [3]:

$$\frac{\partial f}{\partial t} + \vec{S} + \nabla f + V_N |\nabla f| = b_k |\nabla f| \quad (8.2)$$

$\frac{\partial f}{\partial t}$  is the boundary movement depending on the vector field  $\vec{S}$ . The normal direction is represented by the gradient  $\nabla f$  and the velocity  $V_N$ . The curvature is defined by  $b_k$ . The zero level can therefore grow, shrink, or remain at rest based on Eq. 8.2. Experiments performed by Li et al. [4] showed that boundaries grew faster within text lines – where black pixel densities are large and slower when the gaps are approached. Taking advantage of the knowledge that text lines are horizontally written (for most languages), the boundaries can be set to grow faster in the horizontal direction. In addition, the curvature at the ends of a line is larger than the curvatures at the top and bottom. Knowing this, fewer iterations may be needed to reach the goal.

## Segmentation for Optical Character Recognition (OCR)

Any character recognition system is highly dependent on the quality of the segmentation algorithm that precedes it. Segmentation is the process of dividing a document image into smaller elements. The purpose is to simplify a problem into a form which can be more easily manageable. In this chapter, research performed on segmentation of printed characters from a document is discussed. Characters include those found in the ASCII character set as well as some accented characters and symbols. Some research has also been done on segmenting mathematical symbols and characters found in non-Latin documents. Only English documents will be the language we will focus on but the techniques can be applied to most other languages.

Optical Character Recognition (OCR) [26, 38, 39] has been available since 1929. The idea of an OCR system or device has been around since the early twentieth century when several patents for OCR systems were granted. In the 1960s, the OCR-A font was designed to make it easier for both humans to read and for machines to perform OCR on the printed texts. OCR-A is a fixed-width Sans Serif font (see Fig. 8.12).

OCR-A was designed as a simple font with little complications. Each character was evenly spaced and had little extraneous features. Some companies would use this font to print serial numbers, airline tickets, and utility bills, to name a few examples. This limited OCR systems to the commercial environment. Existing printed material, such as literature, business, and legal documents, printed in a font other than OCR-A, was not accurately “read” by the existing OCR systems which were trained specifically on the OCR-A font.

In 1974, the Kurzweil Computer Products company developed an OCR which could read documents printed in any normal font. A few years later, the OCR system was sold commercially to the general public.

**Fig. 8.12** OCR-A font sample

A B C D E F G H I J K L M N O P Q R S T U V W X Y Z  
a b c d e f g h i j k l m n o p q r s t u v w x y z  
0 1 2 3 4 5 6 7 8 9

OCR has several uses. Some practical examples for OCR systems are license plate recognition, book scanning for digital libraries, mail address reader, text to speech, form processing, computer vision, and sign recognition.

OCR systems have improved greatly since the early pioneer years. However, even the current systems must deal with several obstacles that are inherent with any printed document. There are several problems which can occur that the segmentation procedure must overcome in order to obtain the best segmentation results possible. More on the challenges of segmentation of printed documents are described in section “[Region Growth Techniques](#).”

In this section, “[Segmentation for Optical Character Recognition \(OCR\)](#),” some recent research done to overcome the challenges in order to improve printed character segmentation is presented.

## Challenges

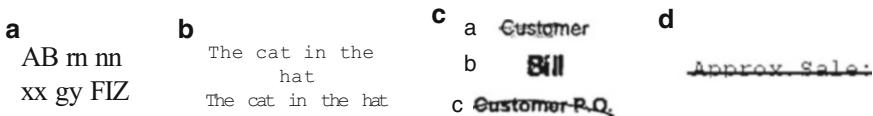
Recognizing printed texts is not a trivial problem. Several obstacles can appear on a document which hinders the segmentation procedure. Because of the vast number of document types, fonts, paper quality, and even how the documents were digitized, this results in an almost endless number of variations for an OCR system to contend with. Since it is not possible to train on all these combinations, researchers instead train on common features which most documents contain.

Ideally, the characters on a printed page would appear clean and isolated, free from any obstructions or damages. However, this is rarely the case. Problems arise that cause the characters on a page to be fragmented, distorted [33], or even disappear. Below is a list of some of the common problems which a segmentation algorithm must overcome.

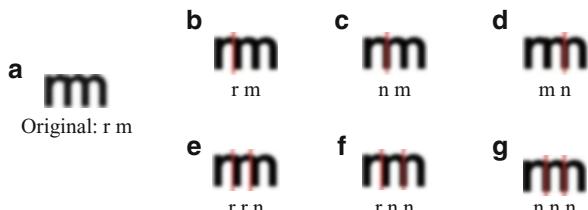
## Touching Characters

The problems regarding poor document image capture can directly affect the quality of the printed characters on the document image. One common problem, which is of great importance for segmentation, is the separation of touching characters [36]. There are several reasons that lead to touching characters. They occur frequently either “naturally” or as a result of poor image capture. Documents that have text printed using a typewriter may have touching characters as a result of a malfunction such as a small-space advance, misaligned typebars, or even if character keys were pressed simultaneously.

Serif fonts have a higher rate of touching characters because of their serifs. When the serifs of neighboring characters are facing each other, the chance of



**Fig. 8.13** Touching characters resulting from (a) Serif font, (b) kerning, (c) noise, (d) strikethrough



**Fig. 8.14** Touching characters with potential cutting points

touching is higher as seen in Fig. 8.13a. Kerning, or the spacing between characters in proportional fonts, is another source of this problem. If the spacing is too small, characters will overlap. Most word processors allow the user to change the default kerning which can lead to more touching characters even for non-serif and fixed-spaced typefaces. Figure 8.13b shows an example of a sample text at regular spacing and when kerning is manually changed.

Noise is any unwanted component on the document image. This includes objects appearing as a result of scanning or photocopying or handwritten strokes overlapping the text. Figure 8.13c shows an example where regular pepper noise can cause characters to touch. The bottom image shows an example where a person crossed out some text causing the characters to be fully connected. Although striking out characters is not really considered as real noise, it has the same effect as the handwritten strike-out as previously mentioned. Figure 8.13d shows a strike-out example.

These examples show some sources responsible for touching characters which are a problem for the segmentation module because it is often difficult to accurately determine the cutting point to split them. Some common difficult cases are “rn” misclassified as the letter “m” and vice versa, “w” with “vv,” “ri” and “n,” “in” with “m,” and “rm” with “nn,” to name a few. Figure 8.14 shows the touching character pair “rm” and some potential segmentation cutting candidates.

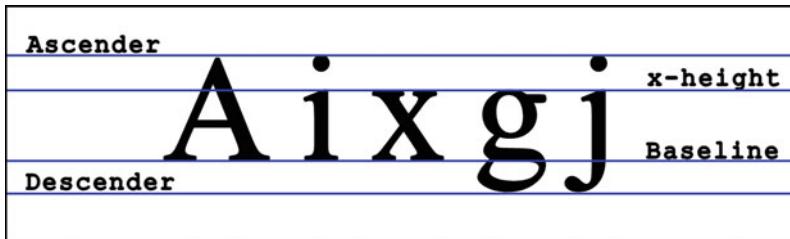
More cutting points can be found in this example; however, Fig. 8.14 shows an example of the difficulty in printed touching character segmentation.

## Broken Characters

Another major concern with segmentation is the case of broken characters. As with touching characters, several reasons can cause characters to be fragmented, for example, uneven ink, faded patches of text from old documents, or damaged paper.

**a** Phone : **b** Date **c** 73521

**Fig. 8.15** Broken character examples



**Fig. 8.16** Text line reference points

Some binarization algorithms during preprocessing will cause these uneven images to develop broken strokes if the binarization threshold is too low, for example.

Repairing broken characters involves an opposite procedure than that for segmenting touching characters. Here, separate components must be merged together to form a character. This could pose a bigger challenge than solving touching characters since some components may not be part of a character but just happened to be in the general area. Figure 8.15 shows some examples of broken characters. Figure 8.15a, b appear to have random breaks in the text. However, a consistent break across the text in Fig. 8.15c may be attributed to a scanner streak or defective scanner.

### Lack of Baseline Information

The baseline is an imaginary line where most of the characters reside on. Figure 8.16 shows the location of the baseline and other reference points of a text line. Most of the time, a line will have a combination of characters with different heights which makes distinguishing between upper- and lowercase much easier. However, for lines where all the character heights are the same, distinguishing between uppercase and lowercase for some characters becomes difficult. The characters which have similar upper- and lowercase shapes are C, K, O, S, U, V, W, X, and Z. Additionally, digits “0” and “1,” which are often misrecognized as “O” and “I,” respectively, also fall into this category.

A line that contains any combination of these characters, mixed with symbols of the same height, may lead to misclassifications in character case. Some examples of case confusing strings are “ZOO” with “zoo,” “COWS” with “cows,” and “VOX” with “vox.” Although the text is correctly recognized in terms of spelling, in some situations, the case may be important if a list of acronyms, codes, or passwords

appear on the document, for example. When all characters in a line have the same height, the segmentation module will not be able to provide the recognizer with the case information.

## Typefaces

There are thousands of typefaces available for creating digital documents. There have been studies that show that some fonts are more legible than others using OCRs [5]. The main difficulty lies in the fact that it is time-consuming to train a system on every known typeface and for every character. Additionally, new typefaces are introduced every day, which would require such a system to be retrained. A more generic approach to segmentation is preferred to avoid this obstacle. However, such a system must deal with the variations in the fonts. Font features that a segmentation module must take into account are: Serif and Sans Serif, point size, proportional or fixed spacing, boldness, and italicized text.

As mentioned before, Serif fonts lead to an increase in touching characters. Section “[Historical Documents](#)” described in more detail the reasons why this is a problem for segmentation. Similarly, proportional spaced fonts, which have different spacing values for each character, also tend to have more cases of touching characters since the text is more compact than fixed-spaced fonts.

Point size can be a problem when the size is either too small or too large. If too small, then features may not appear as evident as with larger point sizes. Figure 8.17 shows two popular typefaces, Courier New and Times New Roman, at various point sizes. The smallest point size displayed (6) in either font shows how features are less pronounced than the larger characters. Some features which are important to segmentation and OCR systems include stroke widths, curves, loops, and stems for Serif fonts. These features are severely reduced in the smaller fonts and, consequently, provide less information to work with. Furthermore, feature calculations will not be as accurate for smaller fonts such as finding the curvature rate for some strokes.

A common practice in OCR is to binarize, normalize [13], and skeletonize a character image to a fixed bounding box size. Downscaling from a larger size to a smaller size may lead to a loss of information as pixels may be dropped. The bigger problem occurs when an image is scaled up from a small point size to the normalized size. This procedure introduces jaggy edges and blocky pixels. As seen in Fig. 8.18, the letter “a” produces a different shape when the smaller image is scaled up than the larger image. In fact, the normalized image derived from the smaller 6-point image does not resemble the letter “a” anymore. The skeletonized images of the normalized ones also show that the smaller “a” now looks more like the digit “8” or the letter “B” than it does “a.” The larger 36-point “a” maintained the shape when scaled up to the normalized size. Even though the edges still appear jagged, the final skeleton appears correct.

Two more features that can lead to problems are bold and italicized fonts. Bold fonts have thicker strokes than the regular normal font. As previously mentioned,

**Fig. 8.17** Courier New and Times New Roman typefaces at different font sizes

	Courier New	Times New Roman	Point Size	abc	abc	abc	abc	abc
	abc	abc	6	abc	abc	abc	abc	abc
	abc	abc	8	abc	abc	abc	abc	abc
			10					
			12					
			16					

**Fig. 8.18** Example of scaling problems due to normalization



**Fig. 8.19** (a) Bold, and (b) italic examples leading to segmentation problems



this leads to more touching characters. Figure 8.19a shows both normal and bold fonts for the Courier New typeface. The normal font for this Serif typeface has no touching characters. When the same text is bolded, the serifs touch. This, as previously discussed, poses a problem for segmentation.

Italicized characters cause challenges for those segmentation algorithms that make use of vertical projection to perform character segmentation. In Fig. 8.19b, the normal font produces a vertical projection with discernible breaks which helps segmentation by easily identifying most character separation spaces. A red line is displayed if the projection in that column contains 1 pixel or less. Only one failed case exists in this example and it is the missed spacing between “f” and “g.”

However, the italicized vertical projection loses most of the inter-character spacing which requires the segmentation algorithm to find other ways to separate the characters.

One last issue deals with nonstandard typefaces. Common typefaces are easier to segment because training is usually performed on them. However, when a less frequent or “artistic” typeface is used, segmentation will usually have trouble with character separation due to the fact that it has not seen (was not trained on) this typeface before. Table 8.1 shows examples of four commonly used typefaces (two Serif and two Sans Serif) at the top. The bottom row of the table shows

**Table 8.1** Examples of four common typefaces and four rarely used typefaces

Segmentation Method	Segmentation Method	Segmentation Method	Segmentation Method
Times New Roman	Courier New	Arial	Frutiger Linotype
Segmentation Method	Segmentation Method	Segmentation Method	Segmentation Method
Letter Gothic	Haettenschweiler	Impact	Harry Potter

examples of more stylized fonts which are less legible. In fact, it was shown that typefaces with lower human legibility have higher OCR error rates [5].

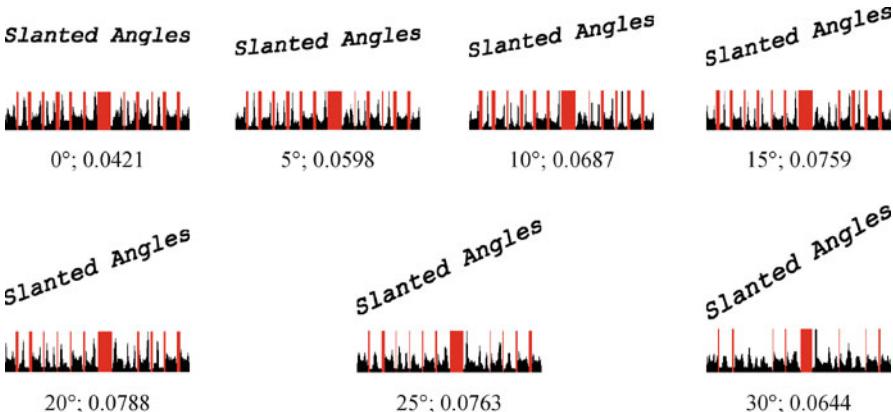
## Touching Italic Characters

In the paper “[A Segmentation Method for Touching Italic Characters](#)” [6], the authors describe a method of solving two common problems which occur together – touching and italic characters. Touching italic characters are the seventh most difficult case (out of eight) for the recognition of printed text [7]. The common practices of using vertical projection or connected component and contour analysis alone do not produce acceptable results and are not suitable for italic fonts. Vertical projection is not useful here as was described in Fig. 8.19 of section [Typefaces](#). Slant correction is not used by the authors because, in general, it introduces noise and distorts the original characters. Instead, the authors proposed a combination of slant projection and contour analysis to segment touching italic characters. This proposed method is independent of slant correction and keeps the font information.

The algorithm accepts an image of a printed text line as input. After cleaning and other preprocessing, the authors perform a four-step procedure to segmentation. The first step is to find the slant angle. Again, this is done without the need to deskew the image. Second, the slant projection, combined with contour analysis, is used to find potential segmentation points. A shortest-path approach is used to determine all the potential cut paths. Based on these potential paths, dynamic programming using recognition is employed to find the paths with the highest recognition confidence values. These paths will be considered to be the final segmentation result. Following are more details on each step.

### Slant Angle

To determine the slant angle of the italicized text line, the image is rotated by angles between  $0^\circ$  and  $30^\circ$  using steps of  $1^\circ$ . For each rotated angle, a vertical histogram is calculated. When finished, each histogram is normalized to values between 0.0 and 1.0. The variance for each histogram is computed, and the angle which contains the highest variance is considered to be the slant angle. Figure 8.20 shows some rotated images between this range (using  $5^\circ$  steps) and their associated vertical histograms. The red lines in the projection indicate a gap between projection peaks. This usually



**Fig. 8.20** Rotated line images with vertical histograms, rotation angle, and variance values



**Fig. 8.21** Slant projection lines using angle obtained from Fig. 8.20

indicates a segmentation point. The variance value is computed for each histogram as well. In our limited example, the line image with the highest variance is for the angle at 20°.

### Segmentation Points

The next step is to find all the connected components in the line. To determine if a connected component needs to be segmented, the aspect ratio and a classifier confidence output are used. If the aspect ratio is above a threshold value or the classifier confidence value is below a threshold value, the connected component is considered to contain touching characters and therefore segmentation will be performed. To find the segmentation points, the authors used two methods – slant projection and contour feature points.

### Slant Projection

Since the slant angle has already been found (see Fig. 8.20), the next step is to compute the slant projection. This is similar to a vertical projection; however, the projection line is along the slant lines. Figure 8.21 shows some of the slant lines from our example of 20°.

The points along a slant line can be computed using the following equation:

$$X_i = X_0 - i * \tan \theta \quad i = 1 \dots n \quad (8.3)$$

where  $i$  is the y-coordinate of a point,  $\theta$  is the slant angle, in our case  $20^\circ$ , and  $n$  is the height of the image. The points  $X_i$  that lie on the line starting with point  $(X_0, 0)$  can be calculated for all  $i$  from  $[0, n]$ . All slant projection values are computed for  $X_0 \in [0, w]$ , where  $w = \text{width of the image}$ .

After the slant projection histogram is computed, it is smoothed and then the local minima points are found. They represent potential segmentation points. Redundant segmentation candidates can be removed if too many appear clustered together. If the distance between two candidates is less than a threshold value, the candidate with the smaller projection value is removed. Additionally, if a vertical line along a segmentation point crosses the character more than twice, it is likely to be incorrect and the candidate point can be removed.

### Contour Feature Points

Some true segmentation points may be missed by the slant projection. This oversight usually occurs when two characters are heavily touching and as a result, will have large slant projection values at the points where the characters are touching. The true segmentation point will be missed because those values will exceed the threshold. To compensate for this, contour feature points are located and used to find more candidate segmentation points. Only the outer contour features are used in this step. To find these feature points requires the following steps:

1. Find the outer contour of each connected component.
2. A point on the upper contour is considered a candidate if, when joined with its two neighboring pixels, forms an upward-facing angle (i.e., a “v” shape).
3. A point on the lower contour is considered a candidate if, when joined with its two neighboring pixels, forms a downward facing angle (i.e., a “” shape).

Figure 8.22 shows the upper and lower candidate feature points. Once the candidate points have been located, they are used to find the cut paths. The traditional method of vertical cutting will not give good performance for italic fonts. Therefore, a curved path is preferred. An ideal cut path is one that goes through the least number of black pixels and follows the slant angle of the italic font. To employ this methodology, penalty costs are used when building a path. Table 8.2 shows the penalty costs for building a segmentation cut path.

Starting from an initial point  $(X_0, 0)$ , points along the imaginary line which lies along the slant angle can be generated by points  $(X_1, 1), (X_2, 2), \dots, (X_n, n)$ . A normal directional move is a move from one point to a neighboring point that follows the slant angle. A deviated move is one that drifts away from the slant angle. The authors found that the best path can be found by only considering the points from  $X_i - 3$  to  $X_i + 3$  on any given row. Therefore, each segmentation point can have an associated cut path.

**Fig. 8.22** Outer contour with candidate segmentation points



**Table 8.2** Segmentation cut path penalty costs

Move to	Direction	Cost
White	Normal	0
Black	Normal	10
White	Deviated	1
Black	Deviated	14

After the connected components have been cut into two, dynamic programming matching is applied. Each segmented component is passed to a neural network classifier and the confidence value returned is used. Those components with high recognition confidence values are kept; the remaining ones are rejected.

## Results

This method was applied to 50 strongly touching/italicized text lines collected by CENPARMI. This dataset consists of 311 touching italic connected components – several of which consist of more than 2 touching characters. The total number of characters is 1,969 from all 50 lines. All the true segmentation points were found as well as the cut paths corresponding to them except for two. Recognition results generated some errors due to the simplicity of the neural network used. Table 8.3 shows the recognition rates using different segmentation methods along with the new method proposed by the authors.

Each method sent the segmented connected components to the same simple neural network classifier in order to omit the classifier influence in the segmentation accuracy results. The new method shows great improvement over the others for touching italicized character segmentation.

The proposed method has both advantages and disadvantages; advantages include the fact that no slant correction is performed. This preserves the character images to be used in their original states without introducing noise and character distortion, as is sometimes the case when performing slant correction.

As a disadvantage, this method assumes the text line to be entirely italicized, which is rare in English documents. Therefore, the usefulness of the method is limited. Similarly, all the characters should be slanted at the same angle and written using the same typeface in order for the method to outperform other methods. Practically, the system may be too slow for commercial purposes. For one reason, slant angles are computed 30 times in order to find the best slant angle. This process

**Table 8.3** Recognition rates using various methods

Segmentation method	Recognition rate (%)
Vertical projection	78.21
Contour analysis	83.24
Vertical projection and contour analysis	87.77
Method [6]	90.66

includes rotating the image and computing a vertical projection for each of the 30 angles. Secondly, calling a neural network classifier for each connected component also adds to the processing time, and therefore, slows down the entire system.

## Segmentation of Degraded Characters

Several researchers have studied the difficult problem of segmenting characters from degraded documents. The source of the degradation could have originated from historical documents [34] or from recent documents that have gone through wear and tear (i.e., coffee spills, sun damage, heat, or other environmental factors) or just through the natural and gradual deterioration of the medium.

In the paper by Moghaddam et al. [8], a method was introduced using multi-level classifiers, and a level set active contour scheme was used to locate the boundaries of the characters in degraded ancient documents. This research aimed to correct two types of problems in segmentation – restoring degraded characters and fixing broken characters.

A degraded grayscale image of a character extracted from a document is input into the algorithm. This degraded image may contain missing pixels and elements from neighboring characters that may appear in the image. Additionally, the average stroke width,  $w$ , is provided to the program as well. The aim is to reconstruct the damaged character as close to the true shape as possible. This is done using the level-set method.

The first step is to obtain the average stroke width,  $w$ . Since the documents are degraded, the character images will contain noncontinuous and thinned strokes; therefore, the  $w$  is reduced to  $w/2$  to compensate for this. The strokes are then converted into a structure usable by the level-set method. A Stroke Map (SM) [9] is created for this purpose. A SM contains all possible stroke pixels in an image by using a kernel method and the average stroke width,  $w$ .

After the SM has been computed, a Stroke Cavity Map (SCM), which is a representation of all the probable stroke pixels in an image, is generated. This includes all pixels in the SM with those that are between two SM pixels and less than  $w$ . The following binary kernel provided by the authors was used to create the SCM:

$$K_{r1,r2}(r) = \begin{cases} 1 & \|r_1 - r_2\| \leq w \& r \in R(r_1, r_2, t) \\ 0 & \text{otherwise} \end{cases} \quad (8.4)$$

where  $R(r_1, r_2, t)$  is a rectangle from  $r_1$  to  $r_2$  of height  $t$ . To take advantage of the fact that the provided image is in the center, the SCM is computed by using a modified spatial decay transform [8]. This transform requires the average character width and height in the entire text from which the degraded character image was taken.

When the SCM is completed, the pixel intensities are computed by using a normalized smoothed histogram. Finally, contours in the level set are found by matching those that intersect the surface at the zero level,  $z = 0$ . This is approximated as a signed distance function  $\Phi(r)$ . When the level set function  $\Phi(r_i) >= 0$ , this signifies that the pixel is part of the text; otherwise, it is labeled as a background pixel. The function is guided by a governing equation which attracts stroke pixels and repels background pixels.

The results for character restoration reported by the authors look very promising. Although it does not perform the segmentation on its own, it can provide a huge assistance to algorithms which attempt to locate and segment degraded characters in historical documents. The average stroke width and the pre-segmented character images must be provided in order for this method to function. In addition, the characters must be centered in the image. Testing needs to be improved – in particular, on more samples and on automated outputs. Although the experiments showed promising results, it was performed only on a few manually extracted sample images.

## Segmentation of Mathematical Expressions

The methods used to segment regular printed text will provide little help when segmenting most mathematical expressions. Mathematical expressions are composed of several more characters, symbols, and valid overlapping components. The expressions include typefaces with superscript and subscript fonts which make recognition much more difficult due to the normalization problem. Table 8.4 shows some common mathematical formulae and some of the unique properties which make them different from regular printed text such that regular segmentation algorithms would not work on them.

Aside from the superscripts and subscripts, there are objects such as the summation symbol ( $\Sigma$ ) and parentheses in Table 8.4(b) and (d) which span more than one row of text and may have some small components below and above it. There also exists some text in the middle of two lines such as  $(a_n \cos)$  in Table 8.4(b). There are numerators and denominators vertically positioned and separated by a dividing line. In addition, new special symbols such as  $\Sigma$ ,  $\pm$ ,  $\pi$ ,  $\infty$ ,  $\sqrt{\cdot}$ ,  $\leq$ , and  $\geq$  and Greek letters are now part of the character set that will need to be recognized and therefore segmented. Mathematical expressions are two-dimensional as opposed to regular text which is just one-dimensional in nature. Segmentation algorithms must now be adapted to segment components in the vertical direction as is shown in some of the examples of Table 8.4. Therefore, methods such as projection will not be very useful for expression segmentation. The segmentation of overlapping and

**Table 8.4** Common equation expressions

$a^2 + b^2 = c^2$	$f(x) = a_0 + \sum_{n=1}^{\infty} (a_n \cos \frac{n\pi x}{L} + b_n \sin \frac{n\pi x}{L})$	$x = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$
(a)	(b)	(c)
$(x+a)^n = \sum_{k=0}^n \binom{n}{k} x^k a^{n-k}$	$e^x = 1 + \frac{x}{1!} + \frac{x^2}{2!} + \frac{x^3}{3!} + \dots, \quad -\infty < x < \infty$	
(d)	(e)	

touching characters is the most difficult part of expression segmentation. According to the authors [10], more than half of the misclassifications are caused by touching characters.

A four-step procedure proposed in [10] begins by identifying expressions in regular text. Each located expression is then segmented into its connected components. Touching characters are detected and segmented. Finally, each extracted component is passed to a classifier for recognition.

The main part, the detection of touching characters, is done by passing the component to a classifier for an initial recognition result. The result from the classifier is used to compare the component with specific precomputed features of that recognized character. Some features used for comparison are the aspect ratio (*height/width*) and the peripheral features. If there is a difference in these feature values with the initial recognized touching pair, then the component will be considered as touching and will be passed to the segmentation module. Otherwise, it is regarded as one character and sent directly to the main classifier for recognition.

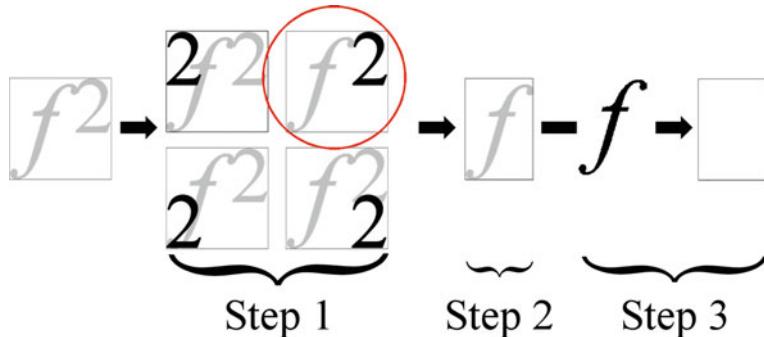
The segmentation will split a component into two characters. Again, an initial classifier is used to assist and guide the segmentation routine. Several touching character pairs were synthesized and trained on. The segmentation will pass the component to an initial classifier, trained on geometric features, to determine which touching pair it most closely matches with. That is, if the difference is larger than a pre-calculated threshold, it is considered to be a touching character pair candidate. Otherwise, it is believed to be a single character and passed directly to the main classifier. To overcome font variations, the current component is matched with previously recognized components from the same document.

Assuming the component is composed of two touching characters, segmentation is a four-step procedure:

1. Search for the first component character
2. Create a residual image
3. Identify the second component character
4. Verification

Searching for the first character begins by choosing an image from the document of a recognized character. This character is used as a template to see if a match is found within one of the four quadrant areas of the touching character image. If a match is found, then this is considered to be the first component character.

The residual image is created by subtracting a thickened version of the character image found in step one from the touching components. A thickened version of



**Fig. 8.23** Touching character segmentation using template matching

the known character is used in order to remove the small differences between the two images. This leaves the other character image as the remainder. Using this first residual image, the same step can be applied to find the identity of this remaining character. An image is taken from the set of single classified images from the same document and is matched with an unknown image. If a match is found, the procedure stops and proceeds to the verification step; otherwise, it continues with another known image and repeats the step. A match is found if subtracting the known character from the residual image leaves an empty image. Figure 8.23 shows the steps used to identify the touching characters.

The final step is to verify the result. Verification is done by creating a synthetic image of the touching characters that were identified in the first and third steps and by using two known characters and merging them. The original touching character image is then matched with the synthetic image by subtracting the two images. If the subtraction returns a blank image, then the verification step will have confirmed the segmentation result.

The verification step is useful to “undo” false-positives. If the difference between the touching characters image with the synthesized touching image does not yield a blank image, then the decision is rejected and a new search is performed to find the true touching pair.

If there exists more than one known component to match with, the representative component is used. The representative component is found by a sequential clustering method where each component is added to its closest cluster and clusters are dynamically split when the variance exceeds a predetermined threshold. Each component in a cluster will have the same result in both the detection and segmentation stages. Therefore, one representative (the centroid) component is used for comparison during both stages.

Results reported in [10] show that out of a test set of about 140,000 connected components, 2,978 were touching characters. The initial classifier, used to determine if a component contains touching characters, performed at 92.9 % – where 60 % of the misrecognition were due to touching characters. Using the clustering method,

**Fig. 8.24** Touching pair containing a broken character



the 140,000 components were grouped into 13,291 clusters. Of these, 909 clusters contained touching characters.

The performance was 96 % successful in detecting touching characters. For segmentation, about 51 % of the actual touching characters were successfully segmented into their two characters. Although not a high value, this does reduce the errors due to touching characters by half. Analysis showed that the three major reasons for the segmentation failure of the 49 % touching characters were (i) either one or both of the characters were not represented in the known isolated characters, (ii) the subtraction step of correct matching character images produced a small but significant remainder that was enough to declare a non-match, and (iii) touching characters were segmented into two incorrect characters (i.e., see Fig. 8.14). Of the three causes, the first was responsible for 40 % of the total errors.

Recognition was reported to be 95.1 %. This technique reduced the number of misrecognitions to 70 % from the initial classifier.

One advantage of this technique is the adaptiveness to the document font style and size. Characters used for matching come from the same document so excessive training is not required. However, this assumes that each character in a touching pair has a known isolated representative character. For short documents, the probability that an isolated representative character exists is smaller.

Further investigation showed that the speed increased the overall performance by one-tenth when not using clustering.

The limitation of the technique is bounded by the availability of known characters to match with those in the touching pairs. When a known character is missing from the document, this causes the technique to fail. The authors claim that the technique can be extended to segmentation of three touching characters. While this is possible, it adds an order of complexity to the system especially because of the large character set being used. Additionally, documents that contain broken characters will fail. For example, Fig. 8.24 shows the touching character pair “DO” with the “O” broken. Using the template-matching technique will not detect the “O” because subtracting a known identified “O” from this one will leave too many leftover pixels. Therefore, the “O” will not be classified, or worse, misclassified.

---

## Conclusion

This chapter presented research and approaches to solving common problems associated with printed document zoning, line, and character segmentation. The solutions must overcome the vast diversity of document layouts, printing

media, and resolution. They must also be able to handle documents written in any language [19]. Historical documents have a set of difficulties such as degraded media and possibly physical missing sections from wear and tear. Wrinkles and creases in the paper will cause problems for segmentation algorithms.

Once these obstacles have been overcome and the text zones have been identified, text segmentation begins. Various techniques for performing this task were discussed. The structural-based procedure using horizontal projection performs very well but requires that the text lines not overlap each other and have a relatively horizontal orientation.

Region growth techniques are an alternative structural method which group neighboring pixels of an image into subregions. Pixels in a subregion share a common feature. Although region growth techniques are better for handling touching, skewed, and overlapping lines, these segmentation methods rely heavily on a good choice of initial seedpoints. Furthermore, they are highly computationally expensive. An advantage is that most noise will not be included in the segmentation result.

A common statistical approach is to use probability density features. By treating a document as a two-dimensional array of pixel intensities, algorithms are used to extract statistical information from this array. These features can be used to determine the most likely line and edge boundaries of text. Probability densities are good for eliminating noise and pixels which are distant from any text, such as pixels in the margins. They are also well suited for handling broken characters within a line. However, obtaining these features can be a time-consuming procedure and may not provide the best information for line segmentation.

Level set methods are used to separate foreground objects from the background in a document. This is ideal for text line segmentation to separate text lines in front of other elements such as a photo or ghost images. The procedure is an iterative process beginning with an initial zero level set and is evolved according to a partial differential equation. This has the disadvantage of having a high computational requirement. The advantage is shown where it is observed that the boundary of a level set grew faster inside text lines that is, where black pixel densities are large. Slower growth was observed when the gaps were approached.

In addition, the segmentation challenges faced for character recognition was described. It was mentioned that several factors affect segmentation such as the typeface, font size, and bold and italicized text. Because of these and other features, this leads to structural problems such as broken and touching characters. A lack of baseline information (when all characters in a text line are the same height) can also impede segmentation. Several algorithms to solve these types of problems as well as slanted lines were pointed out. Degraded characters, which are prevalent in historical documents or documents which have gone through a great deal of wear and tear, need to be specially handled in order to reconstruct missing and damaged characters. Alternative special cases are mathematical expressions. The text layout is not in the form as a regular text block. More than one line can be contained within a larger line. Subscripts, superscript, symbolic characters, and mathematical symbols are introduced. In addition, the kerning (spacing between the characters) is

not predictable as it is with a known typeface. Because of the variation of kerning and font sizes, the majority of misclassifications are caused by touching characters.

There has been a great amount of research on segmentation. Several solutions to common segmentation difficulties were presented. It is unlikely any one solution will achieve a complete result. It is common practice that more than one algorithm is used to obtain the best segmentation possible.

---

## Cross-References

- ▶ [Analysis of the Logical Layout of Documents](#)
  - ▶ [Imaging Techniques in Document Analysis Process](#)
  - ▶ [Machine-Printed Character Recognition](#)
  - ▶ [Page Segmentation Techniques in Document Analysis](#)
- 

## References

1. Li Y, Zheng Y, Doermann D, Jaeger S (2008) Script-Independent text line segmentation in freestyle handwritten documents. *IEEE Trans Pattern Anal Mach Intell* 30(8):1313–1329
2. Brodić D (2010) Optimization of the anisotropic Gaussian kernel for text segmentation and parameter extraction. In: Theoretical computer science. Springer, Brisbane, pp 140–152
3. Sumengen B (2004) Variational image segmentation and curve evolution on natural images. Ph. D. Thesis, University of California, Santa Barbara
4. Li Y, Zheng Y, Doermann D, Jaeger S (2006) A new algorithm for detecting text line in handwritten documents. In: Tenth international workshop on frontiers in handwriting recognition, La Baule, pp 35–40
5. Suen C, Nikfal S, Li Y, Zhang Y, Nobile N (2010) Evaluation of typeface legibility. In: ATypI, Dublin, Sept 2010
6. Li Y, Naoi S, Cheriet M, Suen C (2004) A segmentation method for touching Italic characters. In: International conference on pattern recognition (ICPR), Cambridge, pp 594–597, Aug 2004
7. Lu Y (1995) Machine printed character segmentation – an overview. *Pattern Recognit* 28: 67–80
8. Moghaddam R, Rivest-Hénault D, Cheriet M (2009) Restoration and segmentation of highly degraded characters using a shape-independent level set approach and multi-level classifiers. In: International conference on document analysis and recognition (ICDAR), Barcelona, pp 828–832, July 2009
9. Moghaddam R, Cheriet M (2009) RSLDI: restoration of single-sided low-quality document images. *Pattern Recognit* 42(12):3355–3364
10. Nomura A, Michishita K, Uchida S, Suzuki M (2003) Detection and segmentation of touching characters in mathematical expressions. In: Seventh international conference on document analysis and recognition – ICDAR2003, Edinburgh, pp 126–130
11. Ball G, Srihari S, Srinivasan H (2006) Segmentation-Based and segmentation-free approaches to Arabic word spotting. In: Proceedings of the international workshop on frontiers in handwriting recognition (IWFHR-10), La Baule, pp 53–58, Oct 2006
12. Liu C, Suen C (2008) A new benchmark on the recognition of handwritten Bangla and Farsi numeral characters. In: Proceedings of eleventh international conference on frontiers in handwriting recognition (ICFHR 2008), Montreal, pp 278–283
13. Liu C, Nakashima K, Sako H, Fujisawa H (2004) Handwritten Digit Recognition: Investigation of Normalization and Feature Extraction Techniques. *Pattern Recognition* 37(2):265–279

14. McLachlan G (1992) Discriminant Analysis and Statistical Pattern Recognition. Wiley Interscience, New York
15. Shi M, Fujisawa Y, Wakabayashi T, Kimura F (2002) Handwritten Numeral Recognition Using Gradient and Curvature of Gray Scale Image. *Pattern Recognition* 35(10):2051–2059
16. Li Y, Zheng Y, Doermann D (2006) Detecting Text Lines in Handwritten Documents. In: International Conference on Pattern Recognition, Hong Kong, vol 2, pp 1030–1033
17. Likforman-Sulem L, Vinciarelli A (2008) HMM-based Offline Recognition of Handwritten Words Crossed Out with Different Kinds of Strokes. In: Eleventh International Conference on Frontiers in Handwriting Recognition, Montreal, pp 70–75
18. Zheng D, Sun J, Naoi S, Hotta Y, Minagawa A, Suwa M, Fujimoto K (2008) Handwritten Email address recognition with syntax and lexicons. In: Eleventh international conference on frontiers in handwriting recognition, Montreal, pp 119–124
19. Kessentini Y, Paquet T, Benhamadou A (2008) A multi-stream HMM-based approach for off-line multi-script handwritten word recognition. In: Eleventh international conference on frontiers in handwriting recognition, Montreal, pp 147–152
20. Fei Y, Liu C-L (2008) Handwritten text line segmentation by clustering with distance metric learning. In: Eleventh international conference on frontiers in handwriting recognition, Montreal, pp 229–234
21. Roy P, Pal U, LLados J (2008) Morphology based handwritten line segmentation using foreground and background information. In: Eleventh international conference on frontiers in handwriting recognition, Montreal, pp 241–246
22. Du X, Pan W, Bui T (2008) Text line segmentation in handwritten documents using Mumford-Shah model. In: Eleventh international conference on frontiers in handwriting recognition, Montreal, pp 253–258
23. Liu C-L, Suen C (2008) A new benchmark on the recognition of handwritten Bangla and Farsi numeral characters. In: Eleventh international conference on frontiers in handwriting recognition, Montreal, pp 278–283
24. Mori S, Nishida H, Yamada H (1999) Optical character recognition. Wiley-Interscience, New York
25. Chaudhuri B (2007) Digital document processing: major directions and recent advances. Springer, London
26. Bunke H, Wang P (1997) Handbook of character recognition and document image analysis. World Scientific, Singapore
27. Garain U, Paquet T, Heutte L (2006) On foreground – background separation in low quality document images. *Int J Doc Anal Recognit* 8(1):47–63
28. Morita M, Sabourin R, Bortolozzi F, Suen C (2004) Segmentation and recognition of handwritten dates: an HMM-MLP hybrid approach. *Int J Doc Anal Recognit* 6(4):248–262
29. Hase H, Yoneda M, Tokai S, Kato J, Suen C (2004) Color segmentation for text extraction. *Int J Doc Anal Recognit* 6(4):271–284
30. Sarhan A (2009) Arabic character recognition using a combination of k-means and k-NN algorithms. *Int J Comput Process Lang* 22(4):305–320
31. Karthik S, Hemanth V, Balaji V, Soman K (2012) Level set methodology for Tamil document image binarization and segmentation. *Int J Comput Appl* 39(9):7–12
32. Ouwyed N, Belaid A (2008) Multi-Oriented text line extraction from handwritten Arabic documents. In: Eighth IAPR international workshop on document analysis systems, Nara, pp 339–346
33. Pan P, Zhu Y, Sun J, Naoi S (2011) Recognizing characters with severe perspective distortion using hash tables and perspective invariants. In: International conference on document analysis and recognition, Beijing, pp 548–552
34. Silva G, Lins R (2011) An automatic method for enhancing character recognition in degraded historical documents. In: International conference on document analysis and recognition, Beijing, pp 553–557
35. Saabni R, El-Sana J (2011) Language-Independent text lines extraction using seam carving. In: International conference on document analysis and recognition, Beijing, pp 563–568

36. Kang L, Doermann D (2011) Template based segmentation of touching components in handwritten text lines. In: International conference on document analysis and recognition, Beijing, pp 569–573
37. Bukhari S, Shafait F, Breuel T (2011) Text-Line extraction using a convolution of isotropic Gaussian filter with a set of line filters. In: International conference on document analysis and recognition, Beijing, pp 579–583
38. Marinai S, Fujisawa H (eds) (2010) Machine learning in document analysis and recognition, 1st edn. Studies in computational intelligence, vol 90. Springer, Berlin
39. Cheriet M, Kharma N, Liu C-L, Suen C (2007) Character Recognition systems: a guide for students and practitioners. Wiley, Hoboken

## Further Reading

- Bunke H, Wang P (1997) Handbook of character recognition and document image analysis. World Scientific, Singapore
- Chaudhuri B (2007) Digital document processing: major directions and recent advances. Springer, London
- Cheriet M, Kharma N, Liu C-L, Suen C (2007) Character recognition systems: a guide for students and practitioners. Wiley, Hoboken
- Li H, Doermann D, Zheng Y (2008) Handwritten document image processing: identification, matching, and indexing of handwriting in noisy document images. VDM, Saarbrücken
- Marinai S, Fujisawa H (eds) (2010) Machine learning in document analysis and recognition, 1st edn. Studies in computational intelligence, vol 90. Springer, Berlin



# Language, Script, and Font Recognition

9

Umapada Pal and Niladri Sekhar Dash

## Contents

Introduction.....	292
Language Identification.....	294
Language Overview.....	294
Origin of Language.....	296
Difficulties in Language Identification.....	300
Existing Approaches for Language Identification.....	301
Script Identification.....	302
Script Overview.....	302
Single- and Multiscript Documents.....	305
Script Identification Technology and Challenges.....	307
Machine-Printed Script Identification.....	313
Handwritten Script Identification.....	316
Font and Style Recognition.....	317
Font Terminology.....	322
Font Generation.....	322
Font Variation.....	322
Recognition Strategies for Font and Style.....	323
Conclusions.....	325
Cross-References.....	326
Notes and Comments.....	327
References.....	327
Further Reading.....	330

---

The original version of this chapter was revised: the second author name which was missing has now been added. The correction to this chapter is available at [https://doi.org/10.1007/978-0-85729-859-1\\_40](https://doi.org/10.1007/978-0-85729-859-1_40)

U. Pal (✉)

Computer Vision and Pattern Recognition Unit, Indian Statistical Institute, Kolkata, India  
e-mail: [umapada@isical.ac.in](mailto:umapada@isical.ac.in)

N.S. Dash

Linguistic Research Unit, Indian Statistical Institute, Kolkata, India  
e-mail: [niladri@isical.ac.in](mailto:niladri@isical.ac.in)

**Abstract**

Automatic identification of a language within a text document containing multiple scripts and fonts is a challenging task, as it is not only linked with the shape, size, and style of the characters and symbols used in the formation of the text but also admixed with more crucial factors such as the forms and size of pages, layout of written text, spacing between text lines, design of characters, density of information, directionality of text composition, etc. Therefore, successful management of the various types of information in the act of character, script, and language recognition requires an intelligent system that can elegantly deal with all these factors and issues along with other secondary factors such as language identity, writing system, ethnicity, anthropology, etc. Due to such complexities, identification of script vis-à-vis language has been a real challenge in optical character recognition (OCR) and information retrieval technology. Considering the global upsurge of so-called minor and/or unknown languages, it has become a technological challenge to develop automatic or semiautomatic systems that can identify a language vis-à-vis a script in which a particular piece of text document is composed. Bearing these issues in mind, an attempt is initiated in this chapter to address some of the methods and approaches developed so far for language, script, and font recognition for written text documents. The first section, after presenting a general overview of language, deals with the information about the origin of language, the difficulties faced in language identification, and the existing approaches to language identification. The second section presents an overview of script, differentiates between single- and multiscript documents, describes script identification technologies and the challenges involved therein, focuses on the process of machine-printed script identification, and then addresses the issues involved in handwritten script identification. The third section tries to define font terminologies, addresses the problems involved in font generation, refers to the phenomenon of font variation in a language, and discusses strategies for font and style recognition. Thus, the chapter depicts a panoramic portrait of the three basic components involved in OCR technology: the problems and issues involved, the milestones achieved so far, and the challenges that still lie ahead.

**Keywords**

Document image analysis • Natural language processing • OCR technology • Language and script identification • Font and style recognition • Multiscript OCR • Handwriting recognition • Orthography • Grapheme • Diacritic • Allograph

---

**Introduction**

The science that tries to understand language and its properties from the perspective of human cognition and communication considers script and font as two valuable components. Interpretation and analysis of script and font provide necessary insight

to understand how people take advantage of these elements to fabricate a network of information interchange through encoding knowledge that is sharable across the members of communities in diachronic and synchronic dimensions. In this frame, both script and font are important avenues via which language achieves continuation of its existence and growth through all spatiotemporal barriers across generations. On the axis of empirical science, understanding script and font is a tiny step towards understanding the more complex process of encoding knowledge and information within a collectively approved set of symbols that stand as an approximate visual representative of the sounds used in the spoken form of a language. Also, understanding the form and function of script and font becomes relevant in the wider frame of language policy and language planning, where these two elements work as crucial factors in language survival, language growth, mass literacy, and grass-roots education.

On the other hand, the technology that tries to understand language from the perspective of machine learning and knowledge representation considers script and font as two valuable pools of information whose systematic interpretation can make a computer system more robust and penetrative in extraction of information embedded within written texts formed from scripts with different fonts, styles, and designs. Therefore, it becomes necessary for technologies such as optical character recognition (OCR) to understand in minute detail the fine-grained aspects linked with scripts and fonts of languages, since better understanding of the features of scripts and fonts increases the ability of a system to interpret and extract information required for developing the technology.

Keeping both of these missions in view, in this chapter an attempt is made to address language identification, script identification, and font-cum-style recognition, which are three important areas of OCR technology. These are also important problems in the human cognition processes as people fail to identify a language, script or font if they are not previously trained with necessary information regarding these elements. This chapter, however, does not address these issues, since human recognition of language, script, and font is more directly linked with human psychology and the cognition process rather than with machine learning and technology development. Therefore, the primary goal of this chapter is to present a short yet insightful survey on the problems and issues relating to the development of tools and technology for machine recognition of language, script, and font of a document as well as to refer to the achievements made in these domains. This chapter also aims to provide some glimpses on the present state of the art of this technology and to draw the attention of readers towards new directions in this scientific endeavor.

The chapter is broadly divided into three main sections, covering language identification (section “[Language Identification](#)”), script identification (section “[Script Identification](#)”), and font and style recognition (section “[Font and Style Recognition](#)”). Each section has several subsections where specific problems in each broad area are adequately addressed. An overview of language is provided in section “[Language Overview](#),” the origin of language in section “[Origin of Language](#),” difficulties faced in language identification in section “[Difficulties](#)

in [Language Identification](#),” and existing approaches to language identification in section “[Existing Approaches for Language Identification](#).” The third section further subdivided into “Script Overview”, “Single- and Multiscript Documents”, “Script Identification Technology and Challenges”, “Machine-Printed Script Identification”, and “Handwritten Script Identification”. The fourth section defines font terminology (section “[Font Terminology](#)”), problems of font generation (section “[Font Generation](#)”), font variation (section “[Font Variation](#)”), and font and style recognition strategies (section “[Recognition Strategies for Font and Style](#)”). Thus, this chapter presents a general overview on the three basic components involved in document analysis technology. From this chapter, the reader will primarily learn about three important aspects of document analysis technology: the problems and issues involved, the milestones achieved so far, and the challenges that still remain ahead.

---

## Language Identification

### Language Overview

The general use of the term *language* is embedded with different shades of meaning, which a trained person can decipher if (s)he seriously takes into account the finer sense variations invoked by the term. In general, the term refers to the act of exchange of verbal linguistic signals (known as speech) within a given situation with fellow interlocutors on some topics of social relevance in a particular social setting. Thus, the act of speaking becomes a social-cognitive event that activates the *linguistic* and *communicative competence* as well as *speech repertoire* of the speakers involved in the spoken interaction. On the individual scale, this particular linguistic event of communication and information interchange becomes an *idiolect* when the inherent implication of the general term is minimized to refer to the use of the system by an individual at a particular point in time and place in different sociocultural settings.

The term *language* can also be used to refer to a particular variety of speech or writing produced by the members of a particular social group; For instance, set expressions and phrases such as *scientific language*, *technical language*, *media language*, *corporate language*, *business language*, *adult language*, *child language*, *woman language*, *secret language*, *underworld language*, etc. actually refer to specific varieties of language formed and used by different social groups in specific sociocultural contexts to address specific functional needs.

In the area of language acquisition and language teaching, on the other hand, phrases such as *first language*, *second language*, *third language*, *mother language*, *foreign language*, etc. refer to an abstract system of linguistic properties and communication that combines the collective totality of both speech and writing behaviors of the members of the speech community. It also refers to the innate knowledge of the system by the members of the community.

On the scale of time, the term *language* is used in both synchronic and diachronic senses. In the synchronic sense, it refers to the language used at a particular point in time at one or many places, such as *Modern English*, *Modern French* or *Modern Spanish*, whereas in the diachronic sense, it refers to the language as found to be used at various points of time in the history of its birth and growth, such as *Old English*, *Medieval English*, *Victorian English*, *Modern English*, etc.

In the realm of society and culture, the term *language* is used to classify and cluster groups of different speech varieties, such as *pidgins* and *creoles*, that have unique linguistic identities in relation to the standard varieties used by the members of a society.

All the examples and varieties mentioned above invariably fall under the generic term *natural language*, which is often contrasted with those artificially constructed systems that are used to expound one or more conceptual areas, such as *mathematical language*, *computer language*, *formal language*, *logical language*, etc. This contrast between natural and artificial may be further expanded to include the uniqueness of languages such as *Volapuk* and *Esperanto*, which are artificially devised for providing a universally approved platform for the purpose of communication among people across the world.

At an abstract level, the term *language* may be postulated as a bundle of features of human behavior – the universal properties that are present in all human speeches and writing systems but strikingly missing from animal communication systems. In this sense, the term may be characterized by some *design features*, such as *productivity*, *prevarication*, *duality of patterning*, *learnability*, etc., as proposed by linguists [1, 2]. In the purest abstract sense, however, the term *language* refers to the *innate biological faculty* of an individual through which a human being is empowered to learn and use natural language(s) in all possible ways. This sense is implicit within the concept known as language acquisition device (LAD) – an area of intensive research in *biolinguistics*, *psycholinguistics*, *language acquisition*, and *cognitive linguistics* [4].

We can learn about the language used by a speech community by studying the varieties or dialects used by the members of that community as well as by developing a realistic policy concerning the selection and use of different varieties at different geocultural locations. This is the area of geolinguistics and ecolinguistics, where the term *language* acquires a unique identity to refer to the dialects and other regional varieties endowed with unique geocultural environments and natural elements.

In a similar fashion, the term *language* may enter into the sphere of technical intricacies when we talk about issues such as *language teaching*, *language learning*, *language planning*, and *language policy*. In these areas, we require adequate knowledge to understand the *first language* (i.e., mother tongue), which is distinguishable from the *second language* (i.e., a language other than one's mother tongue) used for several practical purposes, such as *government work*, *education*, *administration*, *migration*, *rehabilitation*, *political campaigns*, *tours and travel*, *commercial activities*, etc.

Since the term *language* is polysemous, it is used to refer to not only human languages but also a variety of other systems of communication – both human and nonhuman – which are natural but not *language* in the true sense of the term; For instance, let us consider phrases such as *sign language*, *body language* or *animal language*. One must agree that the term *language* is used in these cases in a highly metaphorical or figurative sense. These are termed so because these communication systems, in spite of their unique functional identities, share some attributes which are common to the features of natural languages.

Several languages use two different words to translate the English word *language*; For instance, French uses *langage* and *langue*, Italian uses *linguaggio* and *lingua*, Spanish uses *lenguaje* and *lengua*, etc. In each of these languages, the difference between the two words correlates with the difference between the two senses evoked by the English word; For example, in French, while *langage* refers to the language in general, *langue* refers to particular languages. This happens because, in English, it is assumed that a human being not only possesses a *language* (e.g., English, Chinese, German, Bengali, Hindi, etc.) but also has the *language* (i.e., the language faculty) due to which a human being is able to communicate with fellow members. Although it is known that possession of the *language faculty* clearly distinguishes human beings from animals [2], the important fact to note is that one cannot possess (or use) a natural language unless one possesses (or uses) some particular language.

The discussion above shows that the question “What is language?” actually carries with it a presupposition that each natural language spoken in the world is a specific instance of something more general – a unique communication system gifted to the human race. Since linguists are concerned primarily with natural languages, they want to know whether all natural languages have something in common, not being shared by other, human or nonhuman communication systems, so that it is possible to apply to each of them the term *language* and deny the use of that term to other systems of communication. This implies that we need to verify whether it is possible to assign the characteristic features of human languages to other systems of communication before we can call them *language*.

## **Origin of Language**

The debate about the origin of language has continued for many years, with scholars debating whether it is possible to account for natural language by using only the basic mechanisms of learning or whether one needs to postulate some special built-in language devices for this purpose. Scholars such as Skinner [3], who believed that human language was nothing but a *learning-only mechanism*, argued that childhood conditioning or modeling can account for the complexities involved in a natural language. On the other hand, Chomsky [4], Pinker and Bloom [5], and Pinker [6], who support the Cartesian and Darwinian models of language acquisition and learning, believe that the phenomenon of the ease and speed with which a human

child learns a natural language requires something more than a mere behavioristic model to understand how human language originated or evolved.

From analysis of the brain, it is observed that, in the case of mammals except for human beings, both hemispheres look very much alike. In the case of human brains, the hemispheres are different in shape and function. It is assumed that, in the prehistoric period, due to continuous struggles (both physical and cerebral) of human beings with hostile Nature, one of the hemispheres of the human brain developed with reduced capacity. As a result, the neural network in the human brain, instead of spreading in all directions, mostly expanded in linear order within the brain. Due to this development, the left hemisphere is hardly successful to relate things in a normal full-blown multidimensional scheme. However, this feature becomes an advantage as the reduced capability of the left hemisphere proves to be highly useful for ordering things at a linear level. And that is exactly what a human language needs – the capacity of the brain for linear arrangement of linguistic elements. Human language needs the ability of the brain to convert full-blown multidimensional natural events to be reproduced in linear sequences of sounds, and vice versa.

While it is certain that human speech developed long before the advent of writing systems (as it is observed that many human societies have speech but no writing system), no one knows for certain how human languages originated or evolved.

The question of all ages and civilizations is: When and how did language emerge? Did it begin at the time when *Homo sapiens* came into the world nearly 4–5 million years ago? Or did it start with the evolution of *Modern Man* (i.e., Cro-Magnon) nearly 125,000 years ago? We are not sure about the actual antiquity of the origin of language, as there is no evidence to justify our assumptions. Similarly, we are not sure if Neanderthal Man could speak. Although he had a brain that was larger than normal human beings of the Modern Age, his voice box was positioned much higher in his throat (like that of apes), and this position is not very convenient for producing fluent speech. The assumption, therefore, is that it is highly unlikely that Neanderthal Man could speak, although this has been an intriguing question for centuries. Even then, the following question still lingers: How did humans obtain *language* which is so different from the nonverbal signals produced by humans (e.g., *gesture, facial expression, kinesis, proxemics, body movement, smiling, posture*, etc.) as well as from the verbal signals (e.g., *barking, roaring, howling, growling, calling, chirping, twittering*, etc.) produced by animals?

Animals often make use of various physical signals and gestures, which represent their specific biological needs and responses. They, however, cannot produce and use symbols as a human being does in an arbitrary fashion following the conventions of the speech community to which the individual belongs. In the animal world, a particular physical signal or posture usually refers to a particular piece of information, and the interface between the signal and the information is always iconic across time and space. The feature of multisemanticity, which is one of the most important attributes of human language, is lacking from “animal language,” and therefore, animal language fails to provide necessary clues and insights to trace the origin of human language. Human language, as a codified system of

symbols, is framed with several layers through taxonomic organization of linguistic properties such as *phoneme*, *morpheme*, *word*, *sentence*, *meaning*, *text*, and *discourse*.

The oversimplified history of the evolution of human civilization postulates that *Homo sapiens* evolved, as a subdivision, from the hominoid family through the following stages:

- (a) Humans split from the apes nearly 3 million years ago.
- (b) The tool-using *Homo habilis* (i.e., handy man) emerged nearly 2 million years ago.
- (c) *Homo erectus* (i.e., upright man) came into existence nearly 1.5 million years ago. He was able to use fire.
- (d) Archaic *Homo sapiens* (i.e., archaic wise man) arrived nearly 300,000 years ago.
- (e) *Homo sapiens* (i.e., modern human) came into being nearly 200,000 years ago.
- (f) *Homo sapiens* started using stone and other raw materials such as bone and clay nearly 50,000 years ago.
- (g) Around this time, *Homo sapiens* started carving and graving on cave walls. This may be assumed as a carnal stage of communication that later evolved into language.

It is assumed that the characteristic features of grammar of a natural language (such as the distinction in formation of a sentence and a phrase, the organization of inflection classes in paradigm, etc.) may provide necessary clues about the prehistory of a language. When the vocal tract of *Homo sapiens* was reshaped in the course of evolution, it provided a better platform for syllabically organized speech output. This perhaps made it possible to increase the vocabulary of humans, which, in return, helped to develop linear sequences of syllables to frame grammar vis-à-vis syntax with active participation of the neural mechanism that controlled syllable structure. Analysis of sentences of natural languages reveals several features based on which it makes sense to assume syntax as a byproduct of characteristics of syllables (e.g., grammatical *subjects* may be the byproducts of onset margins of speech). This hypothesis comes closer to evidence acquired from biological anthropology, ape language studies, and brain neurophysiology [6].

For generations, one of the primary questions of mainstream linguistics has been how language came into being. Speculations to reply to this question are myriad. Some people considered that human language was invented by our earliest ancestors, who had genetic and physiological properties to develop complex sounds and organize these into meaningful strings to form larger constructions such as words and sentences. This theory is known as the monogenetic theory or monogenesis [6]. On the other hand, the polygenetic theory or polygenesis assumes that human language was invented many times by many peoples at different points in time. For this reason, it is possible to reconstruct the earlier forms of a language, but one cannot go far through the cycles of change before the meanders truly obliterate any possibility of reconstructing protoforms. Scholars suggest that one can, at best,

go back only 10,000 years or so. After that, the trail is lost in the river of oblivion. So, there is no chance for us to know what lay before this.

In the last century, linguists developed a method known as historical reconstruction, in which, based on linguistic evidence from modern languages, one can sufficiently reliably reconstruct the protoforms of a language not available today. Some scholars believed that, by applying this process, they would be able to trace the linguistic evidence to prehistoric periods. However, the primary limitation of this approach is that the method of hypothetical reconstruction can, at best, supply evidence of phonetic and morphemic patterns from nearly 10,000 years ago or so, but it cannot go beyond this period and patterns. Thus, this method also fails to guide the search for the antiquity that could have helped determine the origin of language.

Other speculative linguists imagine that language might have originated from antique systems of communication and information exchange deployed by our ancestors at different junctures of human civilization. To establish their hypothesis, they refer to the similarities observed between animal communication systems and human language. They assume that calls, shouts, and songs of animals might be the sources of human language. If one agrees with this view, one might say that it is true that human language is certainly the result of an evolution of the system of information interchange used in the animal world. In that case, the question of the existence of a “system” in human language is at stake. In the animal world, there are some unique systems of communication, which animals use for fixed biological needs. A honeybee, for instance, when finding a source of pollen, goes back to its hive and dances in a particular manner through which it is able to convey to its fellow bees the kind, direction, and distance of the source. Gibbons, on the other hand, can produce a variety of calls, each being different from the other. Each call has a different connotation, reference, and implication for the members of the clan. This means that a call about the presence of a predator is characteristically different from a mating call or a call made to report the availability of food at some location.

The conclusive inference is that the “language of animals” is fundamentally different from that of humans, because contrary to human language, the “language of animals” is genetically inherited. Moreover, using a fixed number of calls, an animal can share information about different situations or events, such as the position of danger, attack by a predator, location of food, calling for mating, assembling members for collective action, etc. The “language of animals” is a closed world, confined within a fixed set of signals. However, human language is an open-ended world, which is abstract, generative, and free from all kinds of confinement.

We cannot say for sure when and how human language originated or evolved. However, we can definitely say that, when the closed world of signals of the animal world unfolded into an open world of signs and symbols, human language evolved as the most powerful device of thought, expression, and communication.

## Difficulties in Language Identification

The issue of language identification within a frame of multilingualism and globalization is not only linked with optical character recognition (OCR) or language technology but also interlinked with more crucial issues relating to language and nation, linguistic identity, ethnology, anthropology, diaspora, and linguistic-cum-cultural imperialism. The rapid growth in literacy, transportation, and communication has significantly increased the number of mother tongues which were once submerged under some dominating languages at various unknown geographical locations but have now surfaced to demand their separate linguistic identities. These languages require the sincere support of modern technology for their survival, growth, and expansion. At present, the world wide web (WWW) is full of texts produced and presented in all major as well as in those so-called minor and unknown languages. In this context, it is necessary to develop a system that can automatically identify a language or variety to attest to and establish its unique linguistic identity in relation to other languages in the global frame. A system is required to identify to which language a particular piece of text document found on the WWW belongs.

There are, however, a variety of issues in tagging information to identify a language as a distinct variety, such as the following:

- (a) **Dynamicity:** Because of the dynamic nature of human languages, it is very difficult to obtain complete knowledge of a particular language or variety, and therefore it is nearly impossible to create a static categorization scheme for all natural languages.
- (b) **Classification:** This is another difficult issue, as one can use different definitions as working parameters to categorize languages. Also, different purposes may lead one to classify languages in different ways.
- (c) **Definition:** Existing systems of language identification are highly inconsistent in their use of the definition of *language*. In many cases, these systems list those features that are not language specific but generic.
- (d) **Coverage:** There are, at present, more than 6,800 languages spoken in the world. Present systems of language identification are not powerful enough to cover all languages and scale them properly.
- (e) **Documentation:** Existing technology for language identification does not properly document the category to which a particular language or variety belongs. In many cases, the technology provides only the name of a language, which is not adequate for proper identification of a language.

What is understood from the limitations specified above is that any attempt to categorize languages of the world must presume some workable definition of *language*, since there is no single objective definition for *language*. However, when adopting a workable definition to be used in categorizing languages, one has to consider several issues as stated below:

- (a) The degree of actual linguistic similarities between speech varieties
- (b) Intelligibility among the speakers of the languages in communication with one another
- (c) Literacy and ability of the speakers in sharing common literature

- (d) Ethnic identities and self-perception of the language communities
- (e) Other perceptions and attitudes based on political or social issues

A workable definition may be formulated by combining all the factors stated above. However, the combination of the factors will depend on the needs and purpose of the definition maker.

From the technological point of view, script identification may help in identification of a language or variety. However, this strategy will not work for those languages which have no script or writing system. Moreover, it will lead to wrong identification of languages as there are many languages which use one script; For instance, English, German, French, Spanish, Portuguese, and others use the Roman script, although these are different languages. Furthermore, if the text of a particular language is transcribed into another script, the script-based process of identifying a language can also be deceptive. For these reasons, one has to go beyond the level of script or writing system to trace some more language-specific properties and features that may work in a more reliable manner in the act of language identification. In linguistics, when the script fails to provide the information required for language identification, attempts are made to leverage the direction of writing, vocabulary, language-specific terms and words, idiomatic phrases and expressions, grammar and syntax, place and person names, names of geocultural ideas, concepts, and items, etc., which, due to their unique linguistic identity, provide vital clues for proper identification of a language.

## **Existing Approaches for Language Identification**

As mentioned earlier, in India there are about 22 official languages [70], and 2 or more languages can be written using a single script; For example, the Hindi, Nepali, Rajasthani, Sanskrit, and Marathi languages are written with the Devanagari script, whereas the Bangla, Manipuri, and Assamese languages are written with the Bangla script, etc. Thus, language identification and script identification techniques are different, and the script identification technique cannot be used for language identification, which requires more linguistic information about the languages. The problem of determining the language of a document image has a number of important applications in the field of document analysis, such as indexing and sorting of large collections of such images, or as a precursor to optical character recognition (OCR).

The language identification task can be divided into two categories: (i) image-based language identification, and (ii) text-based language identification. Although many commercial systems bypass language identification by simply looking at the OCR result, it seems better to have a system for language identification which may help to enhance the rate of accuracy of an OCR system. Current methods of discriminating paper documents make use of lexical information derived from optical character recognition (OCR) followed by reference to multiple dictionaries to determine the language. Busch et al. [7] proposed a technique for identification of English, French, and German languages from image documents. The method first

makes generalizations about images of characters, then performs gross classification of the isolated characters and agglomerates these class identities into spatially isolated tokens. Other image-based language identification methods have been proposed by Hochberg et al. [9], Sibun and Spitz [10], Beesley [11], etc.

Although there exist many works on text-based identification of non-Indian languages [11–17], to the best of our knowledge, there has been no effective effort towards text-based identification of Indian languages. There exists only one work on text-based identification for Indian languages of web documents, using the N-gram language model for this purpose [18]. So, there is a need for research in this area for multilingual and multiscript countries such as India.

---

## Script Identification

### Script Overview

A script is defined as a graphic representation of the writing system used to write statements and views expressible in a natural language. This implies that a script of a language refers to a particular mode and style of writing and the set of characters used in it. The writing system, which is realized through script, is considered as one of the most versatile strategies used for symbolic representation of speech sounds available in a language. It is designed and deployed in such a manner that it becomes maximally suitable for expressing thoughts, ideas, and views expressible in a language.

A writing system of a natural language is usually endowed with the following six sets of properties [19]:

- (a) It contains a set of predefined graphic symbols and signs, which are individually termed as *characters* or *graphemes* and collectively called as *script*.
- (b) It contains a well-defined set of rules, which are understood and shared by the members of a speech community for representing the characters in written form.
- (c) It includes a set of conventions, which can arbitrarily assign linguistic entity to the symbols, determine their ordering, and define their relations to one another.
- (d) It represents the spoken form of a language whose surface constructions and properties are represented, to a large extent, truly by the symbols.
- (e) It includes a predefined set of rules which can be recalled for interpreting these symbols for capturing the phonetic entities they represent.
- (f) It possesses some physical means for representing these symbols by application to a permanent or semipermanent medium so that all the symbols are interpreted visually as well as through tactile perception.

The study of writing systems of different languages over the centuries has evolved as an independent discipline of investigation (known as *paleography*) in which one tries to investigate and explore the origin of a writing system of a language as well as examine the forms and functions of the individual characters and symbols which are included in the script for writing a language. Within this frame of study, *orthography* demands the full attention of an investigator as it directly relates

to the methods and rules deployed in writing a linguistic form. Since a writing system is a strategy for symbolic representation of thoughts and ideas expressed in speech of a language, it may be considered *complete* to the extent to which it is capable of representing all that may be expressed in the spoken form of the language. A *script*, therefore, is a unique and highly codified collection of characters designed for visual representation of speech sounds used in a natural language. It represents a well-defined and systematic arrangement of distinct graphic symbols (i.e., characters) in specific patterns and orders known as the *alphabet* of a language.

Structural analysis and interpretation of function of the orthographic symbols used in a script of a language are important to the analysis, description, and application of a language because many features of a language are actually encoded in these symbols. In general, information about the form and function of orthographic symbols used in a writing system (or script) of a language becomes indispensable in several areas of applied linguistics, such as *transcription*, *transliteration*, *dictionary compilation*, *language documentation*, *language teaching*, and *language planning* [19]. Also, information in this field becomes useful in several domains of language technology for developing systems and tools for *optical character recognition*, *spoken text transcription*, *cryptography*, *computer keyboard design*, *text-to-speech* and *speech-to-text conversion*, *linguistic knowledge representation*, *machine learning*, *automatic language recognition*, *language digitization*, etc.

The concepts of *grapheme*, *allograph*, *glyph*, etc. become relevant in the study of the script of a language, because these elements contribute to the formation of the script as well as in designing fonts and typefaces for printing and publication. The term *grapheme* is generally used to refer to the specific atomic units of a given writing system. In this sense, graphemes are *minimally significant* orthographic components which, taken together, form a set of building blocks based on which the text of a given writing system is constructed following the rules of correspondence and usage; For instance, in the English writing system, examples of graphemes include the basic letters of the English alphabet, punctuation marks, and a few other graphic symbols such as those used for numerals and diacritics.

An individual grapheme may be represented in various ways in the script of a language. Each variant may be distinct visually and different in formal features from the others, but all these variants are interpreted as members representing the same grapheme. These individual variants are usually known as *allographs* of the grapheme. It is not mandatory that each language script should have a set of allographs along with the set of graphemes. In fact, there are many language scripts where there are no or only a few allographs; For instance, the Roman script does not have any allographs for the graphemes used in the script. On the other hand, the Devanagari and Bengali scripts possess a set of allographs, which are regularly used as orthographic variants of the graphemes. The preference for an allograph over a grapheme in the act of writing a piece of text is usually controlled by various linguistic and extralinguistic factors, such as the following:

- (a) The rules and norms of a writing system of a natural language
- (b) The ease and clarity in visual representation of a written text
- (c) The goal for minimization of time, labor, and energy in writing

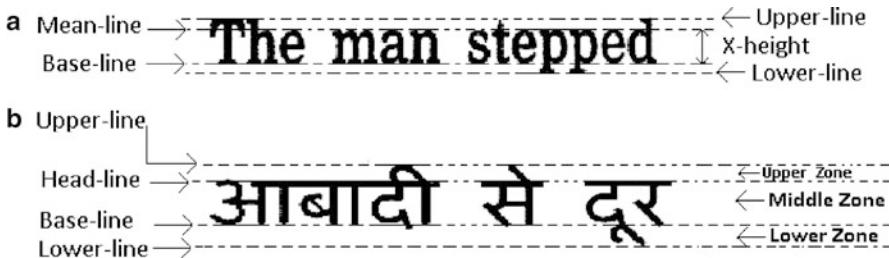
- (d) The tools and mediums used for writing a text
- (e) The stylistic choices of the text composers
- (f) The precision and beauty of the graphic representation of a text
- (g) The unconscious features of handwriting of an individual writer

On the other hand, terms such as *glyph* and *sign* are normally used to refer to the physical aspects and properties of a grapheme, allograph, or diacritic symbol used in the script of a language. These are often made up of straight and twisted lines, strokes, curves, hooks, loops, and other kinds of forms, which are combined together in a different order, sequence, and design to form graphemes, allographs, diacritics, and other symbols used in the script.

The evolution of the human thinking process over the millenniums has been instrumental to the origin, development, and evolution of writing systems (i.e., scripts) of languages. The human thinking process has clearly established a strong connection between the linguistic and cognitive abilities of human beings, as people carefully encode and decode information within a written text for successful communication and information interchange. This implies that the script of a language is one of the most effective means for knowledge representation and communication, as orthographic symbols are used to encode and decode knowledge, convert auditory signals into visual images, help to think deductively, transfer information, and order words to construct meaningful sentences.

Since script is defined as a graphic representation of the writing system used to convert auditory statements into visual forms, a *script class*, therefore, refers to a particular style of writing and the order in which the characters are arranged in it. Moreover, as it is already known that there hardly exists a 1:1 mapping between script and language, it is a foregone conclusion that languages across the world are typeset in different scripts. This means that a particular script may be used by many languages with or without any perceivable variation in the shape and size of the alphabets; For instance, the Roman script is used for *English*, *French*, *German*, and some other European languages; the Devanagari script is used for some Aryan languages including *Sanskrit*, *Hindi*, *Bhojpuri*, *Marathi*, etc.; and the Bengali script is used for *Bengali*, *Assamese*, *Manipuri*, and other languages. Moreover, a language can have more than one script for visual representation of its texts; For instance, the *Santhali* language is written in four different scripts, namely the Devanagari, Bengali, Oriya, and Alchiki scripts, while the *Konkani* language is written in the Kannada, Devanagari, and Roman scripts.

Writing systems of languages may differ structurally, genealogically, geographically, stylistically, etc. One can therefore categorize them in several ways, such as according to their type (classifying according to how the system works), family (classifying according to genealogical relations), and region (classifying according to geographical regions). Also, writing systems can be conveniently classified into broad “types” depending on the way they represent their underlying languages. Bearing these issues in mind, writing systems may be divided into six major types [20]: (a) logographic, (b) syllabic, (c) alphabetic, (d) abjads, (e) abugidas, and (f) featural. Details about these types are discussed in ►Chap. 13 (Middle Eastern Character Recognition) by Abdel Belaid in this handbook.



**Fig. 9.1** Different zones of (a) English and (b) Devanagari lines

In the actual act of writing also there are many differences based on the directionality and case type of characters in scripts. While characters in most scripts are written in left to right direction, the characters in Arabic, Urdu, and some other scripts are written from right to left direction. Similarly, in the case of some scripts (e.g., Latin, Cyrillic, etc.) there are both upper-case and lower-case characters, while for some other scripts (e.g., all Indian scripts, Chinese, Japanese, Arabic, etc.) there is no concept of upper and lower case.

One can identify a text line as English upper case if the upper-line and lower-line coincide with the mean-line and base-line, respectively. By the mean-line (or base-line) of a text line we mean the imaginary horizontal line which contains most of the uppermost (or lowermost) points of the characters of a line. We call the uppermost and lowermost boundary lines of a text line as the upper-line and lower-line (Fig. 9.1). From the structural shape of the script characters, we notice that partitioning of Chinese and Arabic text lines into three zones is very difficult, while an English text line can easily be partitioned into three different zones ordered in three tiers: the upper-zone denotes the portion above the mean-line, the middle-zone covers the portion between the mean-line and the base-line, while the lower-zone covers the portion below the base-line. Bengali and Devanagari text lines can also be partitioned into three zones. Different zones in English and Devanagari text lines are shown in Fig. 9.1. Since the shapes of zone-wise features of characters in the Bengali and Devanagari scripts are not similar, these distinguishable shapes are used as the features for proper identification of Bengali and Devanagari script lines.

## Single- and Multiscript Documents

Based on its content, a document can be classified as either a single- or multiscript document. If a document contains text of only one script, we call it a single-script document. If a document contains text of two or more scripts, we call it bispert or multiscript. There are many countries where two or more scripts are used; For example, a bispert (Japanese and Roman/English) Japanese document is shown in Fig. 9.2, and a triscript (Bangla, Devanagari, and Roman/English) Indian document

この変形を説明するために Bartlett は、知識を組織化し、記憶中の情報を組織化するための構造であるスキーマ schemas あるいはスキマータ schemata を人間が持っていると仮定した。スキーマや、フレーム frames などの知識表現の形式については、後に詳細に解説する。しかし、スキマータもフレームとともに、われわれの記憶がどのように組織化されているかを説明するものである。それらは、語、文、リストを再現する方法についてはあまり考慮しておらず、むしろ記憶のより高度で全体的機能についての側面に焦点を当てている。例えば現実世界の理解にどのようにいたるのか、あるいは、日常的あるいは非日常的な出来事を解釈するために知識をどのように利用しているのか、などがその例である。以下の文章について考えてみよう。

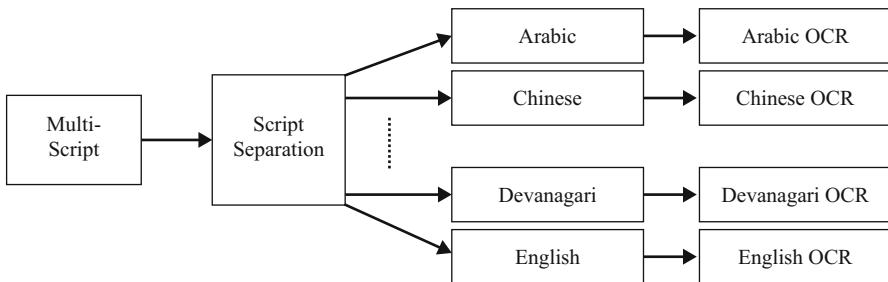
**Fig. 9.2** Example bisscript document containing English and Japanese text

<p>परवरिश ( पर्वरिश्, parvarish )  <i>nf.</i> पालन-पोषण, लालन-पालन,  support, fostering.</p> <p>परवल ( पर्वल्, parval ) <i>nm.</i> पटल,  a kind of kitchen vegetable.</p> <p>परवশ <i>see</i> परवশ !</p> <p>परবা ( पর্বতা, parvā ) <i>nf.</i> প্রত্যেক  পক্ষের অর্থম তিথি, the first day  of each lunar fortnight.—<i>nf.</i></p> <p>পরোয়া, চিন্তা, care, anxiety.</p> <p>পরবান ( पर্ববান्, parvān ) <i>nm.</i>  প্রমাণ, সত্ত্ব, proof.</p> <p>পরবানগী ( पर্ববান্গী, parvangi )  <i>nf.</i> অনুমতি, আদেশ, ইজাজত,  permission, order.</p>
--

**Fig. 9.3** Example multascript (English, Bangla, and Devanagari) document

is shown in Fig. 9.3. So, there is a need for the development of multascript OCR for such countries.

OCR for multascript document pages can be carried out using one of the following two options: (1) development of a generalized OCR system which can recognize all characters of the alphabets of the possible scripts present in the document pages, or (2) development of a script separation scheme to identify the different scripts present in the document pages with the generation of individual OCR for each script alphabet. Development of a generalized OCR system for multascript documents is more difficult than single-script OCR development. This is because the features necessary for character recognition depend on the structural



**Fig. 9.4** Multiscript OCR technology

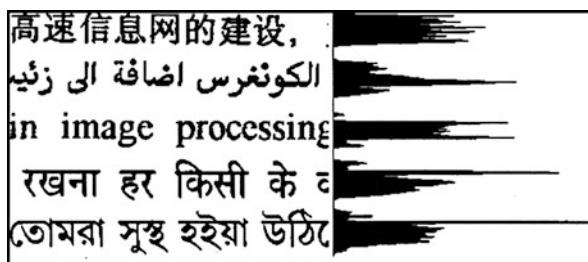
properties, style, and nature of writing, which generally differ from one script to another; For example, features used for recognition of Roman/English alphabets are, in general, not good for recognizing Chinese logograms. Also, the large number of characters in all script alphabets is an added difficulty with this technique. On the other hand, the second option is simpler, with multiscript OCR technology generally working in two stages: (1) identification of different script portions from a document, and (2) feeding of individual script portions to the appropriate OCR system. A flow diagram of multiscript OCR is shown in Fig. 9.4.

## Script Identification Technology and Challenges

Script identification from a document can be done in different modes such as page-wise, paragraph-wise, line-wise, and word-wise. Word-wise script identification is the most difficult task, as the number of characters in a word is small and hence proper script features from the word may not be obtained for its correct identification. Many pieces of work have been done on script identification from printed text [20], whereas only a few pieces of work have been aimed at identification of different scripts from handwritten documents.

Script identification also serves as an essential precursor for recognizing the language in which a document is written. This is necessary for further processing of the document, such as routing, indexing, or translation. For scripts used by only one language, script identification itself accomplishes language identification. For scripts shared by many languages, script recognition acts as the first level of classification, followed by language identification within the script. Script recognition also helps in text area identification, video indexing and retrieval, and document sorting in digital libraries when dealing with a multiscript environment. Text area detection refers to either segmenting out text blocks from other nontextual regions such as halftones, images, line drawings, etc., in a document image, or extracting text printed against textured backgrounds and/or embedded in images within a document. To do this, the system takes advantage of script-specific distinctive characteristics of text which make it stand out from other nontextual parts of the

**Fig. 9.5** Horizontal projection profiles of a text document containing Chinese, Arabic, English, Devanagari, and Bangla text lines (top to bottom)



document. Text extraction is also required in images and videos for content-based browsing.

One powerful index for image/video retrieval is the text appearing in them. Efficient indexing and retrieval of digital images/videos in an international scenario therefore requires text extraction followed by script identification and then character recognition. Similarly, text found in documents can be used for their annotation, indexing, sorting, and retrieval. Thus, script identification plays an important role in building a digital library containing documents written in different scripts.

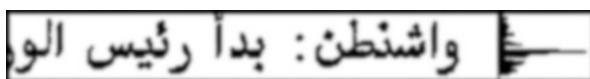
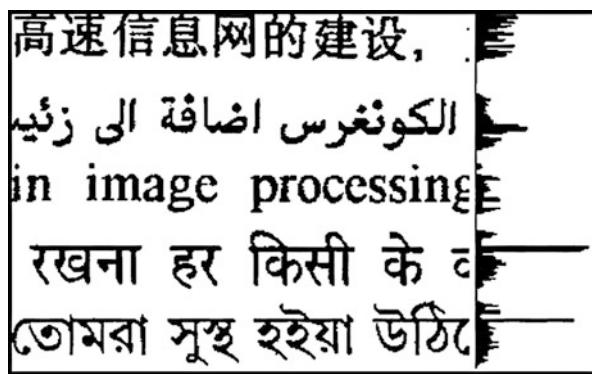
Automatic script identification is crucial to meet the growing demand for electronic processing of volumes of documents written in different scripts. This is important for business transactions across Europe and the Orient, and has great significance in a country such as India, which has many official state languages and scripts. Due to this, there has been growing interest in multiscript OCR technology during recent years.

Script identification relies on the fact that each script has a unique spatial distribution and visual attributes that make it possible to distinguish it from other scripts. So, the basic task involved in script recognition is to devise a technique to discover such distinctive features from a given document and then classify the document's script accordingly.

Many features have been proposed for script identification. Some of the features used in script identification are discussed below. Projection profile is one of the important features used by various researchers for line-wise script identification; For example, see Fig. 9.5, where the horizontal projection profiles of a text document containing Chinese, Arabic, English, Devanagari, and Bangla text lines are shown. From this figure it can be seen that some of the text lines have different behavior in their projection profile. Arabic, Devanagari, and Bangla text lines have only one, large peak in their profile, while Chinese and English do not have this feature but rather have two or more peaks of similar height. Also it can be noted that, in Arabic, the peak occurs at the middle of the text line, while in Bangla and Devanagari the peak occurs at the upper half of the text line.

Run information is also used as a feature in script identification. The maximum run for each row can provide a distinct feature to classify it from others; see Fig. 9.6, where the row-wise longest horizontal run is shown. It can be seen from this figure that Bangla and Devanagari text has a very large run compared with others, which is due to touching of characters in a word through the head-line. Because of the

**Fig. 9.6** Row-wise longest horizontal run in Chinese, Arabic, English, Devanagari, and Bangla text lines (from top to bottom)



**Fig. 9.7** Example of an Arabic text line where a long horizontal run is obtained. Note that, here, the long horizontal run appears at the lower-half of the text line

base-line, sometimes large runs may be obtained from Arabic lines (as seen in Fig. 9.7), but its position is different from others. In Arabic, large runs generally occur in the base-line part, while in Bangla and Devanagari they occur in the mean-line portion.

The crossing count is another important feature for identifying some scripts from others; For example, Chinese script has a higher crossing count than English or Arabic. However, the crossing feature is not very robust due to noise where images may be broken or touched. To tackle this problem, a run length smoothing algorithm (RLSA) concept has been used for script line identification. The RLSA is applied to a binary sequence in which white pixels are represented by 0's and black pixels by 1's. The algorithm transforms a binary sequence X into an output sequence Y according to the following rules: (1) 0's between two consecutive 1's in X are changed to 1's in Y if the number of 0's is less than or equal to a predefined limit L. (2) 1's in X are unchanged in Y. (3) 0's at the boundaries of X are unchanged in Y. For example, in row-wise RLSA with  $L = 5$ , the sequence X is mapped into Y as follows:

$$X : 00001100000001001000100100000011100$$

$$Y : 000011000000011111111100000011100$$

The RLSA is applied row by row as well as column by column to a text line, yielding two distinct bit maps which are then combined by a logical AND operation. The original text lines and the result of applying RLSA are shown in Fig. 9.8. The parameter L can be fixed using text line height information.

**Fig. 9.8** Original text lines (top) and modified RLSA results (bottom) for (a) Chinese, (b) Roman, and (c) Arabic script

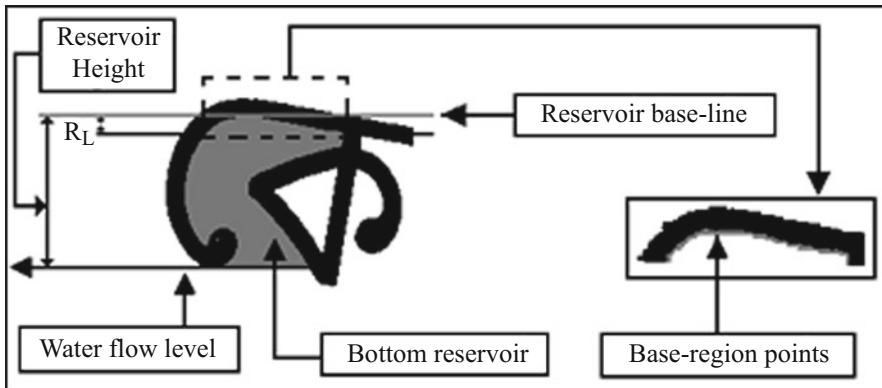


From Fig. 9.8 it can be noted that most of the character portion gets filled up when the RLSA is applied to a Chinese text line, which is not true for English or Arabic text lines. The percentage of character filled-up area of the RLSA output text line with respect to its original version can be used as a measure for script identification. Based on this feature, a Chinese text line can easily be separated from an English one.

Features based on the water reservoir principle are also widely used in script identification. The water reservoir principle is as follows: If water is poured from a given side of a component, the cavity regions of the component where water will be stored are considered as the reservoirs of that component. Characters in a word may touch, and thus two touching consecutive characters in a word create large cavity regions (space) between them and hence large reservoirs [21]. Details of the water reservoir principle and its different properties can be found in [21]. However, here we give a short description of different properties used in the scheme to ease readability.

**Top (bottom) reservoir:** By the top (bottom) reservoir of a component we mean the reservoir obtained when water is poured from the top (bottom) of the component. The bottom reservoir of a component is visualized as a top reservoir when water is poured from the top after rotating the component by  $180^\circ$ .

**Left (right) reservoir:** If water is poured from the left (right) side of a component, the cavity regions of the component where water will be stored are considered as the left (right) reservoir. The left (right) reservoir of a component is visualized as a top reservoir when water is poured from the top after rotating the component by  $90^\circ$  clockwise (anticlockwise).



**Fig. 9.9** Reservoir base-line and water flow level in a bottom reservoir. Base-region points of the bottom reservoir are marked grey in the zoomed part of the component

**Water reservoir area:** By the area of a reservoir we mean the area of the cavity region where water will be stored. The number of points (pixels) inside a reservoir is computed, and this number is considered as the area of the reservoir.

**Water flow level:** The level from which water overflows from a reservoir is called the water flow level of the reservoir (Fig. 9.9).

**Reservoir base-line:** A line passing through the deepest point of a reservoir and parallel to the water flow level of the reservoir is called the reservoir base-line (Fig. 9.9).

**Height of a reservoir:** By the height of a reservoir we mean the depth of water in the reservoir. In other words, the height of a reservoir is the normal distance between the reservoir base-line and the water flow level of the reservoir (Fig. 9.9).

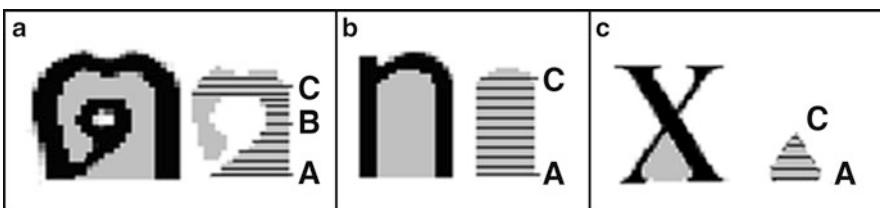
**Base-region points:** By the *base-region* points of a reservoir we mean some of the reservoir's surface (border) points whose perpendicular distances are less than  $R_L$  from the reservoir base-line. Here,  $R_L$  is the stroke width of the component. *Base-region* points of a bottom reservoir are marked by grey in Fig. 9.9. The stroke width ( $R_L$ ) is the statistical mode of the black run lengths of the component. For a component,  $R_L$  is calculated as follows: A component is first scanned row-wise (horizontally) and then column-wise (vertically). If  $n$  different runs of length  $r_1, r_2, \dots, r_n$  with frequencies  $f_1, f_2, \dots, f_n$ , respectively, are obtained after these two scans of the component, then the value of  $R_L$  will be  $r_i$  if  $f_i = \max(f_j)$ ,  $j = 1, 2, \dots, n$ .

Different reservoir-based features can give various distinctive properties that are useful for script identification. Some of these distinctive features are discussed below.

From Fig. 9.10, it can be seen that the water flow level of the top reservoirs of most English characters is the top of the character, whereas this is not true for the characters of the Arabic script. This distinctive feature obtained based on the reservoir is very helpful for identification of Roman and Arabic.

**Fig. 9.10** Example water reservoirs obtained for some (a) English and (b) Arabic characters. Here, reservoirs are marked by *dots*

a H J K L M N U V W X Y k u v w x y  
الصعدين الدولي والمحلي على حد سو b

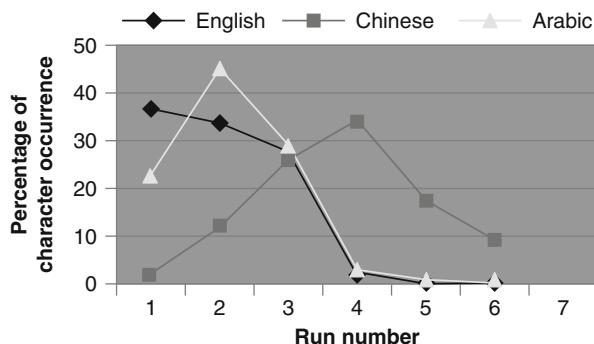


**Fig. 9.11** Shape of bottom reservoir shown in (a) Thai and (b, c) English characters. Reservoir widths of different rows are shown by horizontal lines

Bottom reservoirs also play an important role in script identification; For example, there is a distinct difference between the shapes of the bottom reservoirs in some English and Thai characters, and this reservoir-based shape difference helps in identification of Roman and Thai scripts. To find this shape difference, we compute the width of the reservoir in every row and note the change of these widths. (By the reservoir width of a row we mean the distance between two border points of that reservoir in that row.) Because of the irregular shape of the water reservoir of Thai characters, the reservoir width for different rows will be different as we move towards the bottom of the reservoir starting from the water flow level. However, for English characters, the reservoir width for different rows will either remain the same or tend to decrease as we move upwards from the reservoir flow level, because of the regular shape of the reservoir for English characters. For an illustration, see Fig. 9.11, which shows the shape of the bottom reservoir of one Thai and one English character. For the Thai character, the reservoir width decreases from A to B and again increases from B to C. For the English character, the reservoir width generally remains the same or decreases from A to C.

Run number information is also used as a feature for script identification; For example, one of the most distinctive and inherent characteristics of the Chinese script is the existence of many characters with four or more vertical black runs. In the English and Arabic alphabets, the number of such characters is very low. If the character-wise maximum run numbers of English, Arabic, and Chinese text lines are computed, such distinctive behavior can indeed be observed; For example, see Fig. 9.12, where run statistics are computed from 50,000, 40,000, and 35,000 characters of English, Chinese, and Arabic text, respectively. From this figure it can be observed that the graphs for Arabic and English show similar behavior after run number 3. Also, it can be observed that the graph obtained from the Chinese text has a different nature. In a Chinese text line, about 57 % of characters have four or more black runs. In English, there is only one character ("g") which has four vertical runs.

**Fig. 9.12** Run numbers with their occurrence percentage of characters for English, Chinese, and Arabic texts



We observe that the percentage of characters with one or two vertical black runs in a Chinese text line is very low (about 14 %), whereas in English and Arabic text there are many characters (more than 65 %) with one or two runs. In this regard, Chinese script shows different behavior from the English and Arabic scripts. This simple but important criterion can be used for separation of Chinese from English and Arabic texts.

Among other important features, moments, the concavity distribution, textural features, fractal features, GLCM, wavelets, Gabor features, etc. are used [20]. Different classifiers such as LDA, MLP, KNN, GMM, SVM, etc. are used for different script identification purposes.

### Machine-Printed Script Identification

Among the pieces of earlier work on separation of different scripts, the first attempt to address this problem was due to Spitz [22–24]. In 1990, he described a method for classification of individual text from a document containing English and Japanese script [22]. Later, he [24] proposed a technique for distinguishing Han- and Latin-based scripts on the basis of the spatial relationships of features related to the character structures. Identification within the Han-based script class (Chinese, Japanese, Korean) is performed by analysis of the distribution of optical density in text images. Latin-based script is identified using a technique based on character shape codes, and finding the frequency of language-specific patterns.

Different scripts have distinctive visual appearance. So, a block of text may be regarded as a distinct texture pattern, and this property has been used for script identification. Generally, texture-based approaches are used for block-wise script identification and do not require component analysis. Thus, a texture-based approach may be called a global approach. Representative features for each script are obtained by average values of texture features extracted from training document images. For an input document written in unknown script, similar features are extracted. These features are then compared with the representative features of each script in order to identify the script of the input document. Many researchers have

used texture features for script identification [7, 25–27]. An identification scheme using cluster-based templates for scripts was proposed by Hochberg et al. [25, 26]. They developed a script identification scheme for machine-printed documents in 13 scripts: Arabic, Armenian, Burmese, Chinese, Cyrillic, Devanagari, Ethiopic, Greek, Hebrew, Japanese, Korean, Latin, and Thai. Here, templates for each script were created by clustering textual symbols from a training set. Script identification was then done by comparing textual symbols of an unknown document with those templates. Busch et al. [7] evaluated different textural features such as gray-level co-occurrence matrix features, wavelet energy features, wavelet log-mean deviation features, wavelet co-occurrence signatures, wavelet scale co-occurrence signatures, etc. for the purpose of script identification. They considered English, Chinese, Greek, Cyrillic, Hebrew, Hindi, Japanese, and Persian scripts for their experiment.

Gabor filtering has been commonly used for script identification from printed documents [27, 28]. Using rotation-invariant texture features, Tan [27] described a method for identification of Chinese, English, Greek, Russian, Malayalam, and Persian text. Rotation-invariant texture features are computed based on an extension of the popular multichannel Gabor filtering technique, and their effectiveness was tested with 300 randomly rotated samples of 15 Brodatz textures. These features were then used for script identification from machine-printed documents. Recently, Pam et al. [29] proposed a technique for identification of Chinese, Japanese, Korean, and English scripts using steerable Gabor filters. Here, a Gabor filter bank is first appropriately designed so that rotation-invariant features can handle scripts that are similar in shape. Next, the steerable property of Gabor filters is exploited to reduce the high computational cost resulting from the frequent image filtering, which is a common problem encountered in Gabor-filter-related applications. Gabor filters have also been used to identify scripts from handwritten documents [9].

Ding et al. [30] proposed a method for separating two classes of scripts: European (comprising Roman and Cyrillic scripts) and Oriental (comprising Chinese, Japanese, and Korean scripts), using several discriminating structural and statistical features. Zhang and Ding [31] presented a cluster-based bilingual (English and Chinese) script identification approach. The approach proposed by Lee et al. [32] focuses heavily on reliable character extraction and then uses a variety of decision procedures, some handcrafted and others trainable, to detect document scripts. Ablavsky and Stevens [33] proposed an approach for Russian and English documents that applies a large pool of image features to a small training sample and uses subset feature selection techniques to automatically select a subset with the greatest discriminating power. At run time, they used a classifier coupled with an evidence accumulation engine to report a script label, once a preset likelihood threshold had been reached. Elgammal and Ismail [34] proposed a scheme for identification of Arabic and English text from a document. Features obtained mainly from horizontal projection profiles and run length histograms are used in their scheme. Wood et al. [35] described an approach using filtered pixel projection profiles.

Fractal-based features are also used for script identification. Tao and Tang [36] proposed an approach based on the modified fractal signature (MFS) and modified fractal features (MFF) for discrimination of Oriental and Euramerican scripts.

India has 22 official languages, and 11 scripts are used. In India many documents are written in two or more scripts/languages. Many features, e.g., Gabor features, texture features, water reservoir-based features, head-line, run length smoothing algorithm, edge-based features, directional features, convexity, etc. have been used. To provide an outlook for the reader, details of work done in the Indian scenario are presented below.

The existing pieces of work on Indian script identification mainly rely on local approaches and can be divided into two types. One type of work deals with line-wise script identification. Here, it is assumed that a single line contains only one script. Another type of work deals with word-wise script identification. Here, it is assumed that a single text line may contain words in two or more scripts. In this case, each text line is segmented into individual words and the script of each word of a line is then identified. Word-wise script identification is more difficult and challenging than line-wise identification because of the smaller number of characters and hence lower amount of information obtained.

Although India is a multilingual and multiscript country, generally a document contains three languages. Under the three-language formula, the document may be printed in English, Devanagari, and one of the other Indian state official languages; For example, a document of the West Bengal State government contains English, Devanagari, and Bangla text. To take care of such trilingual (triplet) documents, Pal and Chaudhuri [37] proposed an automatic scheme for separating text lines for almost all triplets of script forms. To do so, the triplets are grouped into five classes according to their characteristics, and shape-based features are employed to separate them without any expensive OCR-like algorithms. Later, they [38] proposed a generalized scheme for line-wise identification of all Indian scripts from a single document.

Chaudhuri and Sheth [39] proposed a trainable script identification strategy for Indian documents containing Devanagari, English, and Telugu scripts. In this work, three trainable classification schemes are proposed for identification of Indian scripts. The first scheme is based upon a frequency-domain representation of the horizontal profile of the textual blocks. The other two schemes are based on connected components extracted from the textual region. These schemes exploit properties of the Gabor filter and the width-to-height ratio of the connected components.

Pal and Chaudhuri [40] proposed an automatic technique for identification of different script lines from a printed document containing the five most popular scripts in the world, namely Roman, Chinese, Arabic, Devanagari, and Bangla. Here, the head-line information is first used to separate Devanagari and Bangla script lines into one group and English, Chinese, and Arabic script lines into another group. Next, from the first group, Bangla script lines are distinguished from Devanagari using some structural features. Similarly, using the vertical run number and run length smoothing [41], the Chinese text lines are identified from the second

group, and then using features obtained from the concept of the water reservoir overflow analogy [42] as well as statistical features, English text lines are separated from Arabic.

One of the pioneering works on word-wise script identification for Indian languages may be credited to Pal and Chaudhuri [43]. Here, *Shirorekha* features and other different stroke features such as vertical line, horizontal line, slanted line, angular shapes, etc. are used in a tree classifier for word-wise identification of Devanagari, English, and Bangla scripts.

Patil and Subbareddy [44] proposed a neural network-based system for word-wise identification of English, Hindi, and Kannada language scripts. In this system, features are first extracted in two stages. In the first stage, the document image is dilated using  $3 \times 3$  masks in horizontal, vertical, right diagonal, and left diagonal directions. In the second stage, the average pixel distribution is found in these resulting images. Based on these features, a modular neural network technique is used for the identification.

Dhanya et al. [45, 46] proposed a Gabor filter-based technique for word-wise segmentation of bilingual documents containing English and Tamil scripts. They proposed two different approaches. In the first method, words are divided into three distinct spatial zones. The spatial spread of a word in the upper and lower zones, together with the character density, is used to identify the script. The second method analyzes the directional energy distribution of a word using Gabor filters with suitable frequencies and orientations.

Manthalkar and Biswas [47] proposed a rotation-invariant texture classification technique for script identification from Indian documents containing Bangla, Kannada, Malayalam, Oriya, Telugu, Marathi, and English.

## **Handwritten Script Identification**

Although there are many pieces of work on script identification from printed documents, only a few pieces of work are available for handwritten documents.

Hochberg et al. [9] proposed a feature-based approach for script identification in handwritten documents and achieved 88 % accuracy from a corpus of 496 handwritten documents in distinguishing Arabic, Chinese, Cyrillic, Devanagari, Japanese, and Latin. In their method, a handwritten document is characterized in terms of the mean, standard deviation, relative vertical centroid, relative horizontal centroid, number of holes, sphericity, and aspect ratio of the connected components in a document page. A set of Fisher linear discriminants (FLDs), one for each pair of script classes, is used for classification. The document is finally assigned to the script class to which it is classified most often.

Fractal features have been used by Roy et al. [49] for script separation of handwritten documents. A fractal [49] is defined as a set for which the Hausdorff–Besikovich dimension is strictly larger than the topological dimension. The fractal dimension is a useful method to quantify the complexity of feature details present in an image. The fractal dimension is an important characteristic of fractals because

it contains information about their geometric structures. When employing fractal analysis, researchers typically estimate the fractal dimension from an image and use it for the desired purpose.

Although Gabor function-based script recognition schemes have shown good performance, their application is limited to machine-printed documents only. Variations in writing style, character size, and interline and interword spacing make the recognition process difficult and unreliable when these techniques are applied directly to handwritten documents. Therefore, it is necessary to preprocess document images prior to application of the Gabor filter to compensate for the different variations present. This has been addressed in the texture-based script identification scheme proposed in [49]. In the preprocessing stage, the algorithm employs denoising, thinning, pruning, m-connectivity, and text size normalization in sequence. Texture features are then extracted using a multichannel Gabor filter. Finally, different scripts are classified using fuzzy classification. In this proposed system, an overall accuracy of 91.6 % was achieved in classifying documents handwritten in four different scripts, namely Latin, Devanagari, Bengali, and Telugu.

Finally, to give an idea of the recognition accuracies of different systems, Table 9.1 summarizes some of the benchmark work reported for printed and handwritten script recognition. Various features and classifiers along with the scripts used by different researchers are also listed in the table, as mentioned in [20].

---

## Font and Style Recognition

A language, if it has a writing system, invariably has a set of typographic symbols and signs (called characters) to represent the sounds used in the language. The complete set of these symbols is known as script, in which each character is rendered in different typefaces, shapes, sizes, designs, and angles – collectively known as font. This means that a font may be defined as a combination of sorts used to compose characters with a unique design, size, and style for a particular script. This signifies that a language can have several fonts to represent the same script; For instance, the Roman script has several fonts developed in different shapes, sizes, and designs, although all of them collectively represent the same script. Moreover, a font of a particular design may vary from its nearest peer based on certain characteristically discernible properties such as height,<sup>1</sup> weight,<sup>2</sup> width,<sup>3</sup> size,<sup>4</sup> slope,<sup>5</sup> curvature, thickness, boldness, etc., which are treated as feature-based parameters in identification of a particular font from the pool of many fonts used for a script.

The differences in fonts are necessitated due to various reasons – starting from personal choice to text content to text readability to typeface compatibility with a computer system. The font used in legal texts, for instance, may vary in design and shape from the font used in medical texts or in texts meant for use by young language learners. Such variation in font use for different text types triggers the question of choice of font and style of writing and also posits problems for font and style recognition by both man and machine, as proper recognition of a font often contributes to correct recognition of a text vis-à-vis a language.

**Table 9.1** Script identification results using different methods applied to printed and handwritten documents

Method	Feature	Classifier	Script type	Document type	Script identification mode	Identification accuracy
Splitz [24]	Upward concavity distribution	Var. comparison	Latin, Han	Printed	Page-wise	100 %
	Optical density	LDA + Eucl. dist.	Chinese, Japanese, Korean			
Lam et al. [50]	Hor. proj. height distribution Circles, ellipses, ver. strokes	Stat. classifier Freq. of occurrr.	Latin, Oriental scripts Chinese, Japanese, Korean	Printed	Page-wise	95 %
Hochberg et al. [25]	Textual symbols	Hamming dist. classifier	Arabic, Armenian, Devanagari, Chinese, Cyrillic, Burmese, Ethiopic, Japanese, Hebrew, Greek, Korean, Latin, Thai	Printed	Page-wise	96 %
Hochberg et al. [51]	Textual symbols	Hamming dist. classifier		Printed	Word-wise	Not available
Pal et al. [52]	Headlines, strokes, ver. runs, lowermost pt., water resv.	Freq. of occurrr.	Devanagari, Bengali, Chinese, Arabic, Latin	Printed	Line-wise	97.33 %
Elgammal et al. [34]	Hor. proj. peak, moments, run-length distribution	Feedforward NN	Arabic, Latin	Printed	Line-wise	96.80 %

Jawahar et al. [54]	Headline, context info.	PCA + SVM	Devanagari, Telugu	Printed	Word-wise	92.3–99.86 %
Chanda et al. [55]	Headline, ver. strokes, tick leftright profiles, water resv., deviation, loop, left incline	Freq. of occur.	Devanagari, Bengali, Latin, Malayalam, Gujurati, Telugu	Printed	Word-wise	97.92 %
Wood et al. [35]	Horizontal/vertical proj.	–	Arabic, Cyrillic, Korean, Latin	Printed	Page-wise	Not available
Jain et al. [56]	Texture feature using discriminating masks	MLP	Latin, Chinese	Printed	Page-wise	Not available
Tan [27]	Gabor filter-based texture feature	Weighted Eucl. dist.	Chinese, Greek, Malayalam, Latin, Russian, Persian	Printed	Page-wise	96.70 %
Peake et al. [28]	GLCM features	KNN classifier	Chinese, Greek, Malayalam, Latin, Russian, Persian, Korean	Printed	Page-wise	77.14 %
Singhal et al. [57]	Gabor filter-based texture feature	Fuzzy classifier	Devanagari, Bengali, Telugu, Latin	Handwritten	Page-wise	91.60 %

(continued)

**Table 9.1** (continued)

Method	Feature	Classifier	Script type	Document type	Script identification mode	Identification accuracy
Busch et al. [7]	GLCM features	LDA + GMM	Latin, Chinese, Japanese, Cyrillic, Greek, Devanagari, Hebrew, Persian	Printed	Para-wise	90.90 %
	Gabor energy					95.10 %
	Wavelet energy					95.40 %
	Wavelet log mean dev.					94.80 %
	Wavelet co-occurrence					98 %
	Wavelet log co-occurrence					99 %
	Wavelet scale co-occurrence					96.80 %
	Gabor energy	KNN classifier	Devanagari, Latin, Gurmukhi, Kannada, Malayalam, Urdu, Tamil, Gujarati, Oriya, Bengali	Printed	Para-wise	97.11 %
	distribution, horizontal projection profile, energy ratios					
Ma et al. [59]	Gabor filter-based texture feature	KNN + SVM multiclassifier	Latin, Devanagari	Printed	Word-wise	98.08 %
Dhanya et al. [46]	Gabor filter-based directional feature	SVM	Latin, Arabic	Printed	Word-wise	92.66 %
			Tamil, Latin	Printed	Word-wise	96 %

Jaeger et al. [60]	Gabor feature	SVM	Arabic, Roman Chinese, Roman Korean, Roman Hindi, Roman	Printed	Word-wise	90.93 %
Chanda et al. [61]	Structural feature based on background and foreground information	SVM	Roman, Thai	Printed	Word-wise	99.62 %
Zhou et al. [62]	Connected component-based feature	Rule-based classifier	Bangla, Roman	Printed	Block-wise (postal address block)	99.00 %
Dhandra et al. [63]	Density feature after morphological operation	Rule-based classifier	Bangla, Roman	Handwritten		95.00 %
Hochberg et al. [9]	Hor./ver. centroids, aspect ratio, white holes	FLD	Arabic, Chinese, Cyrillic, Devanagari, Japanese, Latin	Handwritten	Page-wise	97.00 %
						85.46 %
						88 %

Keeping these crucial issues in view, an attempt is made in this section to refer to some of the methods and approaches developed for font and style recognition in written texts. In the following subsections, attention is concentrated on defining the term “font” within a general conceptual frame, methods of font generation, issues in variation of font formation, and recognition strategies for font and style.

## Font Terminology

In typography, a font is a set of printable or displayable text characters of a single size and style of a particular typeface [64]. The overall design of the character shapes is determined by the typeface, and the style refers to the average stroke width of characters [boldface versus lightface (normal)] and the posture of the body (italic versus Roman). Points or picas are used as units for font size. The size of a font is typically given in points (1 in. = 72 points, 1 in. = 6 picas). Pitch information is also used for size, where pitch means the number of characters displayed per inch. The set of all the characters for 9-point Times New Roman italics is an example of a font. Similarly, the set of all the characters for 10-point Times New Roman italics should be treated as a separate font, as the size is different here.

## Font Generation

In this age of computer technology, it is not a difficult task to design and develop a large number of fonts in digital form for many of the language scripts. According to [64], there are two basic ways to generate a font with a computer, as follows: In the first method, font artwork is scanned and quantized into bitmaps by a scanner, and then the font contours, represented by spline knots [64], are traced out from the high-resolution bitmaps and stored on a disk. The spline knot representation not only is an effective way of achieving data compression but also can serve as an effective master for later processing. In the second method, without using an artwork master, the spline knots are marked on a display screen and then entered with the desired stroke width. Construction of spline curves that pass through given spatial points to form the median or skeleton of the desired font bitmaps can be carried out using this procedure. In bitmap reconstruction, an imaginary electronic paintbrush with elliptical cross-section is swept along the spline curves, rotating along the way with angles complying with the stroke-width information. In this way, a font whose strokes are represented by the spline functions may be generated completely independently without any artwork [64].

## Font Variation

In the present computer age, the wide variety of fonts available may be organized into groups called “font families.” The fonts within a font family show typeface

similarities. As discussed in [75], the members of a font family may differ from each other in that one of them may be bold, another italic or use small caps, etc.; For example, the original Arial font, the Arial Black font, which is bold, and Arial Narrow, which is thinner than the rest, all come under the Arial font family. Some of the most popular font families include Times New Roman, Helvetica, Arial, etc. A font is fully specified by five attributes as noted in Hou [64]: typeface (Times, Courier, Helvetica), weight (light, regular, demibold, heavy), slope (Roman, italic), width (normal, expanded, condensed), and size. See Fig. 9.13 for examples of some fonts.

A very popular classification is that based on five generic font families, which is used for the purposes of cascading style sheets (CSSs) – a widely used mechanism for adding style (e.g., fonts, colors, spacing, etc.) to web documents [75]. The five generic font families consist of serif fonts (Times New Roman, MS Georgia, etc.), sans-serif fonts (MS Trebuchet, Univers, etc.), cursive fonts (Adobe Poetica, Sanvito, etc.), decorative fantasy fonts (Critter, Cottonwood, etc.), and fixed-width monotype fonts (Courier, MS Courier New, etc.).

## Recognition Strategies for Font and Style

Optical font recognition (OFR) is an important issue in the area of document analysis. It aims at recovering typographical attributes with which texts have been printed. Although this is a difficult and time-consuming task, it is an important factor for both character recognition and script identification for multilingual text documents. The number of alternative shapes for each class can be reduced by font classification, leading to essentially single-font character recognition [65]. This helps to improve OCR accuracy. To achieve automatic typesetting, the output of a document processing system should include not only the content of the document but also the font used to print it, and font recognition helps to do this.

Due to its usefulness for both document and character recognition, OFR can be considered to be a challenging problem to reduce the search space of word-to-word matching in keyword spotting. Besides the improvement in OCR and document indexing and retrieval, OFR has many other valuable applications; For example, font identification could be used as a preprocessing step in an automated questioned document analysis process. For crime detection, a forensic expert might be interested in analyzing only documents that are typeset with a particular font. The preprocessing step will help the forensic expert to narrow down the search space to find relevant evidence. Moreover, identifying the font of a document might help an investigator to determine its source of origin (since fonts are sometime country specific).

Font is also used in recognition of logical document structures. Here, knowledge of the font in a word, line, or text block may be useful for defining its logical label, such as chapter title, section title, paragraph, etc. Font information is also required for document reproduction, where knowledge of the font is primarily used in order to reprint a similar document.

<b>A</b>	<b>Wide latin</b>	<b>A</b>	Harrington
<i>A</i>	Vladimir	<i>A</i>	<i>Harlow Solid Italics</i>
<i>A</i>	Vivaldi	<i>A</i>	<i>Gigi</i>
<i>A</i>	Viner Hand ITC	<i>A</i>	<i>Freestyle Script</i>
<b>A</b>	<b>STENCIL</b>	<b>A</b>	Arial Narrow
<b>A</b>	<b>Snap ITC</b>	<i>A</i>	<i>Blackadder ITC</i>
<b>A</b>	<b>Ravie</b>	<i>A</i>	Bradley Hand ITC
<i>A</i>	Pristina	<b>A</b>	<b>Broadway</b>
<b>A</b>	Times new Roman	<i>A</i>	<i>Brush Script MT</i>
<b>A</b>	Papyrus	<b>A</b>	Century Gothic
<i>A</i>	Old English Text ITC	<i>A</i>	Chiller
<i>A</i>	Monotype Corsiva	<b>A</b>	Colonna MT
<b>A</b>	<i>Mature MJ Script Capitals</i>	<b>A</b>	<b>Copper Black</b>
<i>A</i>	<i>Magneto</i>	<b>A</b>	Courier
<b>A</b>	Lucida Fax	<i>A</i>	<i>Edwardian Script ITC</i>
<b>A</b>	<i>Lucida Calligraphy</i>	<b>A</b>	<b>ALGERIAN</b>
<i>A</i>	<i>Kunstler Script</i>	<b>A</b>	Coronet
<b>A</b>	Kristen ITC	<b>A</b>	<b>EUCLID MATH TWO</b>
<b>A</b>	Jokerman	<b>A</b>	Helvetica Narrow
<i>A</i>	<i>Informal Roman</i>	<b>A</b>	<i>Comic Sans MS</i>

**Fig. 9.13** Some examples of different fonts

There are mainly two possible approaches for font recognition: global and local [67]. Global features are generally detected by nonexperts in typography (text density, size, orientation and spacing of letters, serifs, etc.). This approach is suitable for *a priori font recognition*, where the font is recognized without any knowledge of the letter classes. On the other hand, in the local approach, features are extracted

---

from individual letters. The features are based on particularities of letters such as the shapes of serifs (coved, squared, triangular, etc.) and the representation of particular letters at different angles, e.g., “g” and “g,” and “a” and “a.” This kind of approach may derive substantial benefit from knowledge of the letter classes.

In spite of the many applications of automatic font recognition, few pieces of work have addressed these issues. As mentioned earlier, local as well as global features are used by researchers for font recognition. Typological features such as font weight (reflected by the density of black pixels in a text line image), slopes, sizes, typefaces, etc. [66], texture features [68], connected component-based features, and clusters of word images [69] are used for font recognition. Most of the available work on font recognition has been done on English; using 32 English fonts, 99.1 % accuracy was obtained by Khoubayri and Hull [68]. Some pieces of work have been done for Arabic, Chinese, and Japanese [72]. Although pieces of work exist on non-Indian languages, work on Indian script font detection is very limited in spite of the fact that many scripts and languages are used in India [76]. Some pieces of work have been done towards detection of different styles such as bold, italics, etc. For boldface detection, density features are used, whereas for italics detection, slant estimation techniques are incorporated.

Morris [72] proposed a study on the problem of digital font recognition by analyzing Fourier amplitude spectra extracted from word images. This study was mainly performed in order to examine the applicability of human vision models to typeface discrimination and to investigate whether spectral features might be useful in typeface production. First, Fourier transformation is applied to the word image, then a vector of features is extracted by applying many filters to the resulting spectra. A quadratic Bayesian classifier is used for font classification and showed good results considering the many important simplifications. Chanda et al. [76] defined models for characters and placed them in a tree according to certain attributes (ascenders, descenders, holes, etc.). Some preprocessing was done on the text in order to select one sample for each kind of shape (letter). The selected shapes were placed in the tree according to their attributes. In another operation, a tree was generated for each font in the system (instantiation of a generic tree with the different letters of the given font). Next, the shape tree was compared with each font tree in order to compute a distance in order to obtain the associated font. The objective of the font identification was mainly to improve the performance of the OCR system by limiting the search space.

Nowadays, some commercial software is available for font recognition; CVISION (<http://www.cvisiontech.com/reference/ocr/font-recognition-software.html?lang=eng>) is one such piece of software.

---

## Conclusions

This chapter deals with issues of language identification, script identification, and font-cum-style recognition – three important domains in the document analysis area. These domains are so closely interlinked with each other that discussion on one domain without reference to the other two domains is bound to be skewed

and incomplete. Therefore, all three domains were adequately addressed within the frame of a single chapter. After an overview about language in general, the chapter discussed, in a step-by-step process, the origin of language, problems faced in language identification, and approaches used in language identification. The second section presented an overview of script, showed differences between single- and multiscript documents, discussed issues and challenges involved in script identification, defined strategies for printed script identification by machines, and focused on methods for handwritten script identification. The final section defined terminologies used in automatic font recognition, discussed problems of font generation, presented font variations, and highlighted font and style recognition strategies.

In essence, this chapter presents an insightful survey on the problems and issues relating to the development of technology for machine recognition of language, script, and font of text documents. It also provides glimpses on the present state of the art of the technology to draw the attention of readers towards new directions.

In the context of script identification, it may be mentioned that most works on non-Indian languages to date have been based on either line-wise or paragraph/block-wise methods (a block being nothing but a rectangular box containing portions of several consecutive text lines of the same script). This technology, however, does not exert much impact in the context of Indian languages, where multiscript text documents are the rule of the day. Since the lines of text documents in many of the Indian languages contain two or more language scripts, it becomes necessary to design a word-based script identification technique that will yield a higher level of accuracy in case of multiscript documents. Moreover, there is an urgent need for rigorous operation on poor and noisy images of Indian text documents, since most existing script identification techniques operating on Indian languages have considered only clear and noiseless text documents.

On the other hand, there are many handwritten text documents where there is a need for application of script identification techniques. A postal document produced in India, for instance, may be written by hand in any of the Indian languages. In this case, we need to design an automatic postal sorting machine to recognize pin codes and city names. There exists only one work on word-based identification of handwritten Indian documents containing Bangla and Devanagari scripts [49]. This technique may be customized for other Indian and foreign-language scripts, with wider application potential across the globe.

Finally, we assume that online script identification techniques may become highly useful with extensive use of pen-computing technology. However, since these techniques have not received much attention to date, more work is needed in this direction. A similar observation is valid for automatic script identification technology from video-based text documents.

---

## Cross-References

- [Asian Character Recognition](#)
- [Middle Eastern Character Recognition](#)

## Notes and Comments

- <sup>1</sup>Height refers to the vertical length of a character between the two imaginary lines drawn on its head and foot, respectively.
- <sup>2</sup>Weight signifies the thickness of the character outlines relative to its height.
- <sup>3</sup>Width refers to the horizontal stretch of a character within the specified space allotted to it for its formation and visual representation.
- <sup>4</sup>Size refers to the coverage of space of a character in accordance to its degree of readability.
- <sup>5</sup>Slope is connected with the angle at which a particular character is made to stand in a text made in linear order. It can be upright, straight or slanted in right or left direction.

---

## References

1. Greenberg J (1963) Universals of languages. MIT, Cambridge
2. Hockett CF (1960) The origin of speech. *Sci Am* 203:89–97
3. Skinner BF (1953) Science and human behavior. Macmillan, New York
4. Chomsky AN (1959) On certain formal properties of grammars. *Inf Control* 2:137–167
5. Pinker S, Bloom P (1990) Natural language and natural selection. *Behav Brain Sci* 13(4): 707–784
6. Pinker S (1995) The language instinct: the new science of language and mind. Penguin Books, Middlesex
7. Busch A, Boles WW, Sridharan S (2005) Texture for script identification. *IEEE Trans Pattern Anal Mach Intell* 27:1720–1732
8. Nakayama T, Spitz AL (1993) European language determination from image. In: Proceedings of the 2nd international conference on document analysis and recognition, Tsukuba, pp 159–162
9. Hochberg J, Bowers K, Cannon M, Kelly P (1999) Script and language identification for handwritten document images. *Int J Doc Anal Recognit* 2:45–52
10. Sibun P, Spitz AL (1994) Language determination: natural language processing from scanned document images. In: Proceedings of the applied natural language processing, Stuttgart, pp 15–21
11. Beesley KR (1988) Language identifier: a computer program for automatic natural-language identification of on-line text. In: Language at crossroads: proceedings of the 29th annual conference of the American Translators Association, Seattle, pp 47–54
12. Cavner WB, Trenkle JM (1994) N-gram based text categorization. In: Proceedings of the third annual symposium of document analysis and information retrieval, Las Vegas, pp 161–169
13. Cole RA, Mariani J, Uszkoreit H, Zaenen A, Zue V (eds) (1997) Survey of the state of the art in human language technology. Cambridge University Press, Cambridge
14. Hays J (1993) Language identification using two and three-letter cluster. Technical report, School of Computer Studies, University of Leeds
15. Ingle NC (1976) A language identification table. *Inc Linguist* 15:98–101
16. Mathusamy YK, Barnard E (1994) Reviewing automatic language identification. *IEEE Signal Process Mag* 11:33–41
17. Souter C, Churcher G, Hayes J, Hughes J, Johnson S (1994) Natural language identification using corpus-based models. In: Lauridsen K, Lauridsen O (guest eds) *Hermes J Linguist* 13:183–203
18. Majumder P, Mitra M, Chaudhuri BB (2002) N-gram: a language independent approach to IR and NLP. In: Proceedings of the international conference on universal knowledge and language, Goa, 25–29 Nov 2002
19. Dash NS (2011) A descriptive study of the modern Bengali script. Lambert Academic: Saarbrucken

20. Ghosh D, Dube T, Shivaprasad AP (2010) Script recognition-a review. *IEEE Trans PAMI* 32(12):2142–2161
21. Pal U, Roy PP, Tripathy N, Llados J (2010) Multi-oriented Bangla and Devnagari text recognition. *Pattern Recognit* 43:4124–4136
22. Spitz L (1990) Multilingual document recognition. In: Furuta R (ed) *Electronic publishing, document manipulation, and typography*. Cambridge University Press, Cambridge/New York/Melbourne, pp 193–206
23. Spitz AL (1994) Text characterization by connected component transformation. In: Proceedings of SPIE, document recognition, San Jose, vol 2181, pp 97–105
24. Spitz L (1997) Determination of the script and language content of document images. *IEEE Trans Pattern Anal Mach Intell* 19:235–245
25. Hochberg J, Kelly P, Thomas T, Kerns L (1997) Automatic script identification from document images using cluster-based templates. *IEEE Trans Pattern Anal Mach Intell* 19: 176–181
26. Hochberg J, Kerns L, Kelly P, Thomas T (1995) Automatic script identification from images using cluster-based templates. In: Proceedings of the 3rd international conference on document analysis and recognition, Montreal, pp 378–381
27. Tan TN (1998) Rotation invariant texture features and their use in automatic script identification. *IEEE Trans Pattern Anal Mach Intell* 20:751–756
28. Peake GS, Tan TN (1997) Script and language identification from document images. In: Proc. Eighth British Mach. Vision Conf., Essex, UK, vol 2, pp 230–233
29. Pam WM, Suen CY, Bui T (2005) Script identification using steerable Gabor filters. In: Proceedings of the 8th international conference on document analysis and recognition, Seoul, pp 883–887
30. Ding J, Lam L, Suen CY (1997) Classification of oriental and European scripts by using characteristic features. In: Proceedings of the 4th international conference on document analysis and recognition, Ulm, pp 1023–1027
31. Zhang T, Ding X (1999) Cluster-based bilingual script-segmentation and language identification. In: Character recognition and intelligent information processing, Tsinghua University, China, vol 6, pp 137–148
32. Lee DS, Nohl CR, Baird HS (1996) Language identification in complex, un-oriented and degraded document images. In: Proceedings of the IAPR workshop on document analysis and systems, Malvern, pp 76–98
33. Ablavsky V, Stevens M (2003) Automatic feature selection with applications to script identification of degraded documents. In: Proceedings of the 7th international conference on document analysis and recognition, Edinburgh, pp 750–754
34. Elgammal M, Ismail MA (2001) Techniques for language identification for hybrid Arabic–English document images. In: Proceedings of the 6th international conference on document analysis and recognition, Seattle, pp. 1100–1104
35. Wood S, Yao X, Krishnamurthi K, Dang L (1995) Language identification for printed text independent of segmentation. In: Proceedings of the international conference on image processing, Washington, DC, pp 428–431
36. Tao Y, Tang YY (2001) Discrimination of oriental and Euramerican scripts using fractal feature. In: Proceedings of the 6th international conference on document analysis and recognition, Seattle, pp 1115–1119
37. Pal U, Chaudhuri BB (1999) Script line separation from Indian multi-script documents. In: Proceedings of the 5th international conference on document analysis and recognition, Bangalore, pp 406–409
38. Pal U, Sinha S, Chaudhuri BB (2003) Multi-script line identification from Indian documents. In: Proceedings of the 7th international conference on document analysis and recognition, Edinburgh, pp 880–884
39. Chaudhuri S, Sheth R (1999) Trainable script identification strategies for Indian languages. In: Proceedings of the fifth international conference on document analysis and recognition, Bangalore, pp 657–660

40. Pal U, Chaudhuri BB (2001) Automatic identification Of English, Chinese, Arabic, Devnagari and Bangla script line. In: Proceedings of the sixth international conference on document analysis and recognition, Seattle, pp 790–794
41. Wang KY, Casey RG, Wahl FM (1982) Document analysis system. IBM J Res Dev 26:647–656
42. Pal U, Roy PP (2004) Multi-oriented and curved text lines extraction from Indian documents. IEEE Trans Syst Man Cybern B 34:1676–1684
43. Pal U, Chaudhuri BB (1997) Automatic separation of words in Indian multi-lingual multi-script documents. In: Proceedings of the fourth international conference on document analysis and recognition, Ulm, pp 576–579
44. Patil SB, Subbareddy NV (2002) Neural network based system for script identification in Indian documents. Sadhana 27:83–97
45. Dhanya D, Ramakrishna AG, Pati PB (2002) Script identification in printed bilingual documents. Sadhana 27:73–82
46. Dhanya D, Ramakrishna AG (2002) Script identification in printed bilingual documents. In: Proceedings of the document analysis and systems, Princeton, pp 13–24
47. Manthalkar R, Biswas PK, An automatic script identification scheme for Indian languages. [www.ee.iitb.ac.in/~uma/~ncc2002/proc/NCC-2002/pdf/n028.pdf](http://www.ee.iitb.ac.in/~uma/~ncc2002/proc/NCC-2002/pdf/n028.pdf).
48. Mantas J (1986) An overview of character recognition methodologies. Pattern Recognit 19:425–430
49. Roy K, Pal U, Chaudhuri BB (2005) A system for neural network based word-wise handwritten script identification for Indian postal automation. In: Second international conference on intelligent sensing and information processing and control, Mysore, pp 581–586
50. Lam L, Ding J, Suen CY (1998) Differentiating between oriental and European scripts by statistical features. Int J Pattern Recognit Artif Intell 12(1):63–79
51. Hochberg J, Cannon M, Kelly P, White J (1997) Page segmentation using script identification vectors: a first look. In: Proceedings of the 1997 symposium on document image understanding technology, Annapolis, pp 258–264
52. Pal U, Chaudhuri BB (2002) Identification of different script lines from multi-script documents. Image Vis Comput 20(13–14):945–954
53. Padma MC, Nagabhushan P (2003) Identification and separation of text words of Kannada, Hindi and English languages through discriminating features. In: Proceedings of the 2nd Indian conference on document analysis and recognition, Mysore, India, pp 252–260
54. Jawahar CV, Pavan Kumar MNSSK, Ravi Kiran SS (2003) A bilingual OCR for Hindi–Telugu documents and its applications. In: Proceedings of the international conference on document analysis and recognition, Edinburgh, Aug 2003 pp 408–412
55. Chanda S, Sinha S, Pal U (2004) Word-wise English Devnagari and Oriya script identification. In: Sinha RMK, Shukla VN (eds) Speech and language systems for human communication. Tata McGraw-Hill, New Delhi, pp. 244–248
56. Jain AK, Zhong Y (1996) Page segmentation using texture analysis. Pattern Recognit 29(5):743–770
57. Singhal V, Navin N, Ghosh D (2003) Script-based classification of handwritten text documents in a multilingual environment. In: Proceedings of the 13th international workshop on research issues in data engineering–multilingual information management, Hyderabad, pp 47–53
58. Joshi GD, Garg S, Sivaswamy J (2006) Script identification from Indian documents. In: Proceedings of the IAPR international workshop document analysis systems, Feb 2006, Nelson, New Zealand, pp 255–267
59. Ma H, Doermann D (2003) Gabor filter based multi-class classifier for scanned document images. In: Proceedings of the international conference on document analysis and recognition, Aug 2003, Edinburgh, Scotland, pp 968–972
60. Jaeger S, Ma H, Doermann D (2005) Identifying script on word-level with informational confidence. In: Proceedings of the international conference on document analysis and recognition, Aug/Sept 2005, vol 1, pp 416–420
61. Chanda S, Pal U, Terrades OR (2009) Word-wise Thai and Roman script identification. ACM Trans Asian Lang Inf Process 8(11):1–21

62. Zhou L, Lu Y, Tan CL (2006) Bangla/English script identification based on analysis of connected component profiles. In: Proceedings of the 7th international workshop on document analysis and systems, Nelson, pp 243–254
63. Dhanda BV, Nagabhushan P, Hangarge M, Hegadi R, Malemath VS (2006) Script identification based on morphological reconstruction in document images. In: Proceedings of the international conference on pattern recognition (ICPR'06), Hong Kong, pp 950–953
64. Hou HS (1983) Digital document processing. Wiley, New York
65. Zramdini A, Ingold R (1993) Optical font recognition from projection profiles. Electron Publ 6(3):249–260
66. Zhu Y, Tan T, Wang Y (2001) Font recognition based on global texture analysis. IEEE Trans PAMI 23(10):1192–1200
67. Zramdini A, Ingold R (1998) Optical font recognition using typographical features. IEEE Trans PAMI 20(8):887–882
68. Khoubyari S, Hull JJ (1996) Font and function word identification in document recognition. Comput Vis Image Underst 63(1):66–74
69. Jeong CB, Kwag HK, Kim SH, Kim JS, Park SC (2003) Identification of font styles and typefaces in printed Korean documents. In: Sembok TMT et al (eds) ICADL 2003, Kuala Lumpur. LNCS 2911, pp 666–669
70. [http://en.wikipedia.org/wiki/Languages\\_of\\_India](http://en.wikipedia.org/wiki/Languages_of_India)
71. Sharma N., Chanda S, Pal U, Blumenstein U (2013) Word-wise script identification from video frames, 12th International conference on document analysis and recognition, Washington DC, USA pp 867–871
72. Morris RA (1988) Classification of digital typefaces using spectral signatures. Pattern Recognit 25(8):869–876
73. Anigbogu JCh (1992) Reconnaissance de Textes Imprimés Multifontes à l'Aide de Modèles Stochastiques et Métriques. Ph.D. dissertation, Université de Nancy I
74. Abuhaiba ISI (2003) Arabic font recognition based on templates. Int Arab J Inf Technol 1:33–39
75. <http://www.ntchosting.com/multimedia/font.html>
76. Chanda S, Pal U, Franke K (2012) Font identification: in context of an Indic script. In: Proceedings of the 21st international conference on pattern recognition (ICPR), Tsukuba, pp 1655–1658

## Further Reading

Greenberg [1] presents a classification of world languages based on their typological features. Skinner [3] provides ideas about how human behaviors may be modeled after animal behavior controlled by some operant conditioning (this has been criticized by Chomsky in his early works). Reading Chomsky is mandatory for understanding the generative aspects of language. Pinker [6] presents an excellent narration about the growth and development of natural language as a part of natural selection – a theory based on the Darwinian model of natural selection. Cole, Mariani, Uszkoreit, Zaenen, and Zue [13] present an informative sketch on the state of the art in human language technology from the perspective of machine learning and knowledge representation. Pal and Chaudhuri [52] propose an effective strategy for identification of different script lines from multiscript documents. For more details about script identification, we refer to the paper [20]. The book by Dash [19] presents a complete picture about the form and function of characters and other orthographic symbols used in the modern Bengali script. Jain and Zhong [56] present a good model for page segmentation using a texture analysis system. The book by Hou [64] is a basic text on digital document processing and may be used for initial reading. Zhu, Tan, and Wang [66] propose a good method for font recognition based on global texture analysis.

---

# Machine-Printed Character Recognition

# 10

Huagu Cao and Prem Natarajan

## Contents

Introduction.....	332
Overview.....	332
History of Machine-Printed OCR.....	332
Technological Evolution.....	335
Summary of the State of the Art.....	336
Segmentation and Preprocessing.....	338
Binarization.....	338
Page Segmentation.....	339
Line, Word, and Character Segmentation.....	339
Deskew.....	339
Normalization.....	341
Isolated Character Recognition.....	341
Feature Extraction.....	342
Character Recognition.....	345
Word Recognition.....	348
Character Segmentation.....	348
Hidden Markov Model OCR.....	350
n-Gram Language Model.....	351
Systems and Applications.....	353
Applications.....	353
Commercial Software.....	354
Well-Known Evaluations and Contests.....	355
Conclusion.....	355
Cross-References.....	356
Notes.....	356
References.....	356
Further Reading.....	358

---

H. Cao

Raytheon BBN Technologies, Cambridge, MA, USA

e-mail: [hcao@bbn.com](mailto:hcao@bbn.com)

P. Natarajan

Information Sciences Institute, University of Southern California, Marina Del Rey, CA, USA

e-mail: [pnataraj@isi.edu](mailto:pnataraj@isi.edu)

**Abstract**

This chapter reviews salient advances in techniques for machine-printed character recognition. Section “Overview” provides a historical perspective (The description of the historical evolution of OCR is based upon the Wikipedia entry for this topic: [http://en.wikipedia.org/wiki/Optical\\_character\\_recognition](http://en.wikipedia.org/wiki/Optical_character_recognition). The reader is referred to that page for a more detailed review) on how OCR techniques have evolved from the earliest stage (mechanical device) to special-purpose reading machines and to personal computer software. Section “Summary of the State-of-the-Art” summarizes the state of the art in machine-printed character recognition. Sections “Segmentation and Preprocessing”, “Isolated Character Recognition”, and “Word Recognition” describe core technologies including binarization, document image preprocessing, page segmentation, feature extraction, character classification, and language modeling that have been developed for modern character recognition systems. Section “Systems and Applications” introduces available machine-printed OCR systems and applications.

**Keywords**

Artificial intelligence • Binarization • Document analysis • Hidden Markov model • Image processing • Language model • Optical character recognition • Page segmentation • Pattern recognition • Reading machine

---

## **Introduction**

Optical character recognition (OCR) is one of the earliest applications of artificial intelligence and pattern recognition. From the early fixed-font reading machines to today’s omni-font OCR software, the research and development of machine-printed OCR have led to mature techniques and their successful commercialization.

---

## **Overview**

### **History of Machine-Printed OCR**

The history<sup>1</sup> of machine-printed OCR can be divided into distinct evolutionary phases: mechanical reading devices, special-purpose reading machines, and software applications for personal computers. The development of the computer industry played an important role in the evolution of the history of OCR, including the emergence of feature extraction and recognition techniques, diversification of applications, new market demands, and massive reduction in average retail prices.

## Mechanical Reading Devices

Since the first reading machine for the blind was patented in 1809 [1], OCR has been applied to applications as diverse as converting characters to telegraph code and creating reading machines for the blind. In 1912, Emanuel Goldberg invented a reading machine for automatic telegraphic message transmission. He later applied OCR techniques to the task of data entry. Soon thereafter, in 1913, Fournier d'Albe of Birmingham University invented the optophone that scanned text in printed documents and produced time-varying tones to identify different letters. Visually impaired people could “read” the printed material after learning the correspondence between the tones and letters.

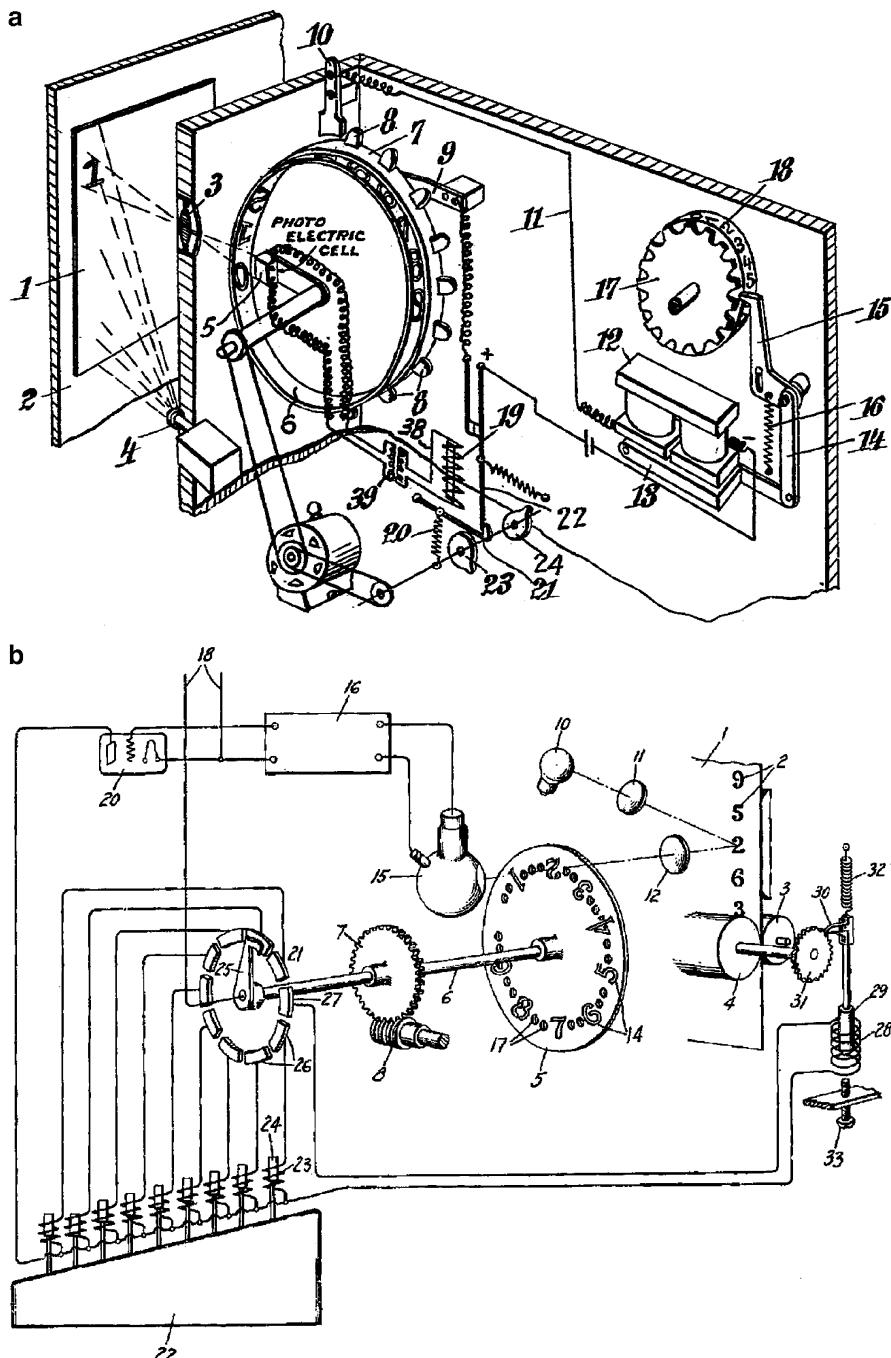
Developments in OCR continued throughout the 1930s. Two systems developed during that time are shown in Fig. 10.1 and were invented by Gustav Tauschek and Paul W. Handel. Despite the differences in the design of the templates (roll vs. plate), both systems used light sources to illuminate the print. The light passed through the print and the templates and activated photoelectrical selector circuits which were designed to “recognize” the right character according to the amount of received light. These early techniques used optical devices with rotating templates of hollow letters that were used to match letters in the print – a fact that likely led to the technology itself being called “optical character recognition.”

## Special-Purpose Reading Machines

OCR techniques became more important when computers were invented in the 1940s. In 1949, RCA developed the first computer-based prototype OCR system funded by the US Veterans Administration. An early text-to-speech technique was also developed to speak the recognized words and help blind people read the text. In 1950, David Shepard was awarded a patent for his OCR machine “Gismo.” He later founded Intelligent Machines Research Corporation (IMR) and developed the first computer-based commercial OCR systems. IMR’s systems were sold to clients such as Reader’s Digest, the Standard Oil Company, and the Ohio Telephony Company. OCR technology truly came into prime time in the 1960s when the United States Postal Service (USPS) began to use OCR machines for mail sorting based on technology developed by Jacob Rainbow [2].

## Software Applications for Personal Computers

Fast-forwarding to the 1990s, the market for OCR technology expanded rapidly due to two factors. The first was the emergence of OCR software developed for personal computers. For example, the first version of the OmniPage OCR software for Apple Macintosh and IBM PC released by Caere in 1988 featured full functionalities of page layout analysis for text and graphics and omni-font character recognition and debuted at a retail price of \$800. While \$800 might not seem so inexpensive in light of the price of desktop OCR software today, at the time it represented a 50-time reduction over the price of OCR machines which sold for over \$40,000.



**Fig. 10.1** Reading machines invented in the 1930s. (a) Reading machine by Gustav Tauschek (US Patent 2026329). (b) Reading machine by Paul W. Handel (US Patent 1915993)

The second factor was the development of a new generation of low-priced yet highly capable scanners that allowed users of personal computers to efficiently scan paper documents into digital images.

Today, OCR technologies are used for many tasks including scanner-based data entry, handheld price label scanners, recognition of documents with personal information such as bank checks and business cards, automatic mail sorting, and in business intelligence systems that support indexing and retrieval of large, heterogeneous document archives.

## Technological Evolution

In early years, research in OCR techniques focused on hardware-only solutions that used mechanical implementations of template matching and selection circuits. One result of the mechanical approach was that the recognition techniques had limited to no generalization ability. The development of the computer revolutionized the field.

Since the invention of computers, sophisticated image processing and statistical pattern recognition techniques have been developed and applied to the tasks of document image analysis and recognition. Over time, independent research at several laboratories across the world has resulted in significant advances in OCR technology. These disparate efforts all share certain methodological similarities. For example, binarization is now considered a necessary step in OCR systems, and extensive research has been dedicated to the development of robust, adaptive techniques such as the Niblack [3] and the Sauvola and Pietikainen [4] algorithms. Page segmentation and layout analysis became hot topics in OCR research with the emergence of principled approaches to the analysis of complex page layout including text, graphics, and tables. Word recognition algorithms using character over-segmentation and dynamic programming have been well studied for cursive handwriting recognition and have also benefitted machine-printed OCR, especially for cursive machine-printed scripts such as Arabic. The Modified Quadratic Discriminant Function (MQDF) classification approach [5,6] has become the approach of choice for ideographic scripts such as Chinese, Japanese, and Korean that have large alphabets.

The hidden Markov model (HMM) approach was applied to character recognition [7] during the 1990s. The ability of HMM-based systems to automatically learn from unsegmented training data coupled with the script-independent modeling methodology originally developed by BBN [7] makes the HMM-based OCR technique an attractive choice for developing retargetable OCR systems that can be rapidly and economically configured for new scripts. In a manner reminiscent of its effectiveness in speech recognition, the HMM-based approach has shown remarkable flexibility and generalizability as evidenced by its highly successful application to the task of offline handwriting recognition [7].

Not surprisingly, the advent of smartphones and the concomitant ubiquitous availability of digital cameras have sparked widespread research interest in

developing algorithms and software capable of operating on the smartphone and with handheld camera-captured text images. Starting in 2005, a new workshop, camera-based document analysis recognition (CBDAR), has been held every other year to address the increasing amount of attention drawn to using camera phones and handheld cameras as the primary capturing device for OCR and present the advances in restoration, segmentation, and recognition of camera-captured documents. Images captured with handheld cameras manifest several new challenges for OCR; salient challenges include out-of-focus images with blurry text, illumination variations, and variable resolution. These challenges make binarization a particularly difficult task. As a result, there is new emphasis on using features that are directly computed from grayscale images to overcome challenges in binarization of camera-captured document images. Text detection has also become important as a research topic since locating text from video or camera-captured scene pictures where there are relatively few constraints on the layout of the text is more challenging than analysis of document images. In addition to the traditional areas of research in machine-printed character recognition, branches of the research, such as music note recognition and mathematical expression understanding, are now increasingly drawing the attention of people in the community.

From a system engineering perspective, major OCR vendors now provide distributed OCR services including web-based OCR and server-based OCR. Web-based OCR or online OCR services are typically targeted to the retail market by allowing Internet users to upload scanned document images and convert them into searchable text in prevailing formats such as HTML, PDF, and MS Word. Server-based OCR is typically deployed for distributed OCR services within the private network of an organization.

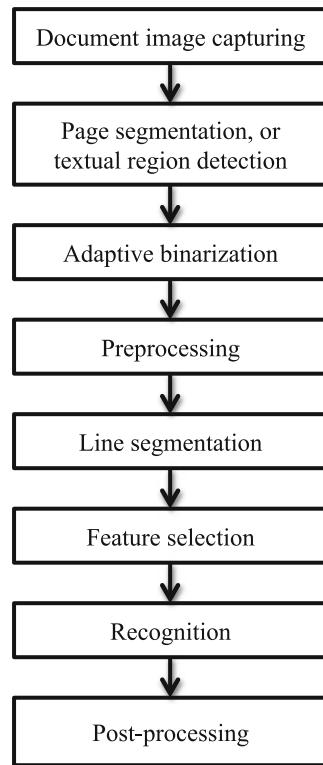
---

## Summary of the State of the Art

The processing pipeline of a typical OCR system is shown in Fig. 10.2. First, the captured image goes through the page segmentation or text detection module and text regions are located. Often times, knowledge of the genre of the text image, e.g., whether it is a newspaper image, a magazine, a technical journal, or a business card, is used to improve the performance of the text detection module. Adaptive binarization and preprocessing steps such as noise filtering and deskewing are then applied to textual regions. Features are computed from binarized and preprocessed textual regions and sent to the word recognizer for recognition. Post-processing involves steps such as language model-based error correction and automatic highlighting of suspicious recognition errors for interactive human proofreading. A stochastic language model has become part of the recognition algorithm in many more modern systems, especially the HMM-based ones.

The core OCR performance on perfectly segmented machine-printed character images in most nonstylized fonts of major languages has reached an error rate

**Fig. 10.2** Block diagram of a typical OCR system



of less than 1 % and, therefore, has very little room to improve. Segmentation often leads to more OCR errors than isolated character recognition. Most OCR systems have adopted recognition-driven character segmentation approaches to reduce segmentation errors. Page segmentation on complex-layout documents is now a mature technology, but text detection in camera-captured images and videos is still an active, challenging research subject.

Challenges also remain in recognition of noisy data (camera-captured/natural scene), infrequent words (named entities), offline handwritten documents, mixed printed/handwritten characters, and complex recognition-plus-parsing problems (music notes, mathematical expressions). Table 10.1 provides a summary view of the key areas of work.

As mentioned earlier, a variety of commercial OCR systems are available today for machine-printed character recognition applications including data entry, digital library, automatic mail sorting, form reader, aid for the blind, and vehicle plate reading.

In the following sections, each of the important steps in the pipeline of Fig. 10.2 is considered in detail.

**Table 10.1** Maturity of areas in machine-printed character recognition

Area	Typical data	Status
Isolated character feature selection and extraction	Scanned document images	Mature techniques available
Isolated character classification	Scanned document images, camera-captured scene	Mature techniques available
Word and character segmentation	Scanned document images	Mature techniques available
Page segmentation on well-formed layout (magazine, newspaper)	Scanned document images	Mature techniques available
Character feature selection and extraction	Scanned document images	Mature techniques available Some room to improve remains
Text detection and segmentation	Camera-captured scene	Significant improvements possible
Text recognition	Scanned document images	Mature techniques available for office quality documents Significant room for improvements in “degraded” or “real-world” documents such as faxed hardcopy, forms, and mixed machine-print + handwritten content
Text recognition	Camera-captured images and videos	Active area of research; limited capabilities available

## Segmentation and Preprocessing

Before the document image reaches the core OCR engine (character recognizer), it needs to go through a few preparatory steps which include binarization, page segmentation, automatic deskewining, line segmentation, and, sometimes, word/character segmentation.

### Binarization

The goal of binarization is not only to compress the size of document images but also to separate the text from the background to enable the computation of useful features for character recognition. While many scanning devices often provide binarization with an adjustable constant threshold as an option when the user scans the document, the constant threshold can produce suboptimal binarization in the presence of noisy or textured backgrounds. The most commonly used automatic constant-threshold binarization algorithm is the Otsu algorithm [8]. The algorithm aims to minimize the combined spread (intra-class variance) of the intensity of the two classes separated by the threshold. Adaptive thresholding algorithms such as the Niblack

algorithm [3] and the Sauvola algorithm [4] are typically more effective at binarizing noisy, textured document images using locally selected thresholds. A survey of the literature shows that researchers have investigated a variety of other binarization approaches (►Chap. 4 (Imaging Techniques in Document Analysis Process)).

## Page Segmentation

Page segmentation refers to the process of segmenting the document image into homogeneous regions such as single-column textual regions, graphical regions, and tables. The X–Y cut is a top–down approach that divides the page recursively using the vertical and horizontal projection profiles of the page. Bottom–up approaches such as the Voronoi diagram-based segmentation [9] and the docstrum [10] have also been applied to page segmentation. Mao and Kanungo [11] (►Chap. 5 (Page Segmentation Techniques in Document Analysis)) provides the reader with a comprehensive survey on different page segmentation approaches (Fig. 10.3).

## Line, Word, and Character Segmentation

Line segmentation is the problem of dividing a textual region into images of lines of text. Word segmentation is the problem of dividing a line image into word images. Character segmentation is the problem of dividing a word or line image into character images. While HMM-based approaches jointly perform segmentation and recognition and do not require a separate pre-segmentation of the line into words or characters, most other approaches require such pre-segmentation.

Among segmentation problems for Latin-script machine-printed document images, line segmentation, and word segmentation are easier to perform than page segmentation and character segmentation. They can be performed using the horizontal and vertical projection profiles, respectively, and can be refined using connected component analysis. In general, character segmentation is a very challenging problem and requires feedback from recognition to improve the accuracy. Typically, approaches that require pre-segmentation will segment the script into lengths that are much shorter than the average length of a character and use recombination rules to stitch the atomic segments into character hypotheses. For non-HMM-based approaches, heuristics such as the average character width can be used as feedback to separate character images recursively. In the case of HMM-based approaches character duration models can be automatically trained and naturally integrated into the recognition process [7] (►Chap. 8 (Text Segmentation for Document Recognition)).

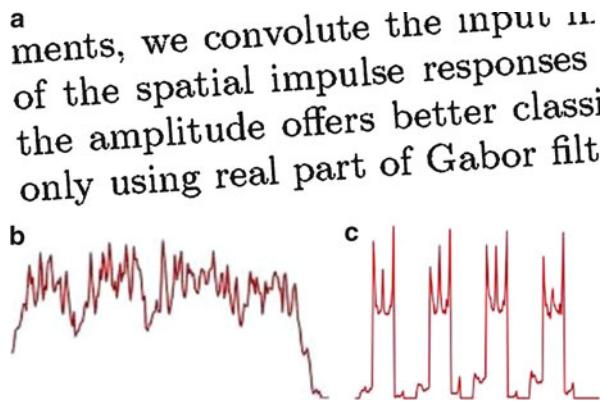
## Deskew

In most cases, the goal of deskewing is to make the text boundaries parallel to the image boundaries. Typically, it is a two-step process. The first step uses



Fig. 10.3 Page segmentation results on a newspaper image

**Fig. 10.4** Deskew using projection profile. (a) Skewed textual region. (b) Horizontal projection profile. (c) Projection profile at optimal angle



one of several projection-profile techniques to estimate the skew angle across the entire document. The second step is simply the application of an appropriate transformation to rotate the image by the inverse of the skew angle. There are several techniques described in the literature for deskewing scanned document images [12–15].

A more complex problem related to deskewing is the problem of straightening out the warping of a text line that happens when a thick book is scanned using a scanner. Again, researchers have developed effective techniques for dewarping the text lines in these cases [16–20] ([► Chap. 4 \(Imaging Techniques in Document Analysis Process\)](#)).

## Normalization

Differences in the placement of bounding boxes and variations of fonts and sizes can increase the within-class character shape variance. This affects zoning-based features in particular. Thus, character image normalization is an important step in most of zoning-based feature selection algorithms. Classic character image normalization methods aim to equalize stroke density with the character using estimated moment features of the character image [21]. Learning-based classification approaches can effectively combine features extracted from the raw image as well as the normalized image to improve performance over the use of either set of features by themselves.

---

## Isolated Character Recognition

Isolated character recognition was an important research area in machine-printed character recognition. Although it is no longer an active area, explorations in this area have provided us indispensable technical progress, the results of which are embodied in today's segmentation-based OCR techniques.

There was an early recognition of the need for discriminative features and feature transformations that can help a recognition engine to distinguish character glyphs from one another effectively. Therefore, in addition to classifier design and training, design and implementation of feature extraction techniques was an important focus of the research in isolated character recognition.

In the following, some of the salient feature extraction and classification techniques that were developed for the task of isolated character recognition are considered. Of course, immense research activity in the area of machine learning has resulted in general pattern recognition approaches like the support vector machine (SVM) [22] that can also be applied to the problem and used within an OCR system.

## Feature Extraction

Despite the many differences between existing languages in the world, at a fundamental level character and word glyphs in the vast majority of them are rendered as line drawings. Except for very few symbols, none of the scripts of modern languages use shading to express any meaning. This lack of creativity in scripts has allowed more rapid progress in the development of recognition approaches than would otherwise have been possible.

Histogram features are the most commonly used family of features for character recognition. Histogram features at the pixel level of the character image are obtained by decomposing the image signal around the pixel into several orientations using directed transformations (gradient operator, concavity, Gabor filter, etc.). The features of the whole image are usually obtained by sampling the character image both horizontally and vertically at certain intervals and computing histogram features at those selected locations. Character images may be normalized to distribute the mass evenly in the frame of the character before sampling. Histogram features are an effective tool to describe the structure of line drawings. These types of features are discussed in more detail later in this section.

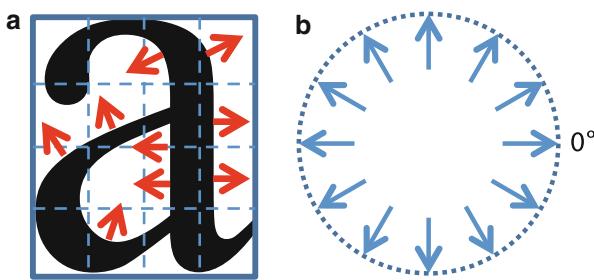
## GSC Features

The gradient, structural, and concavity (GSC) features [23] are an excellent example of histogram features for binarized character images. A character image is partitioned into four by four patches. From each patch, three types of features (gradient, structural, and concavity) are computed. The gradient features are computed by applying the Sobel gradient operators in Eq. (10.1):

$$G_x = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix}, G_y = \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix} \quad (10.1)$$

to the whole character image. The gradient at each pixel is quantized into 12 orientations:  $30 \times i^\circ$  ( $i = 0, 1, \dots, 11$ ). A total of 12 features are defined as

**Fig. 10.5** Gradient histogram features. (a) Gradient orientations in a character image. (b) Twelve quantized orientations



$$f_{gi} = \frac{\text{\#black pixels in the patch with gradient orientation } i}{\text{\# pixels in the patch}} \quad (10.2)$$

The numerator in Eq.(10.2) does not include pixels of zero gradients as their orientations are undefined (Fig. 10.5).

Structural features are derived from gradient features. The  $12^{3 \times 3}$  possible assignments of orientations of all 3 by 3 image blocks are clustered into 12 classes, representing the local structure of the central pixel. A total of 12 structural features are defined for each patch. Each structural feature is defined as

$$f_{si} = \frac{\text{\#black pixels of structural class } i \text{ in the patch}}{\text{\# pixels in the patch}} \quad (10.3)$$

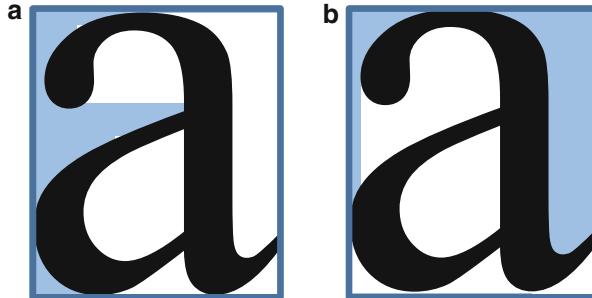
Concavity features<sup>2</sup> decompose the image background signal (white pixels) into different orientations. These orientations describe the concavity of text strokes. The steps to compute concavity features are as follows. First, scan the character image from one side to the opposite side row by row if from left to right or from right to left and column by column if from top to bottom and from bottom to top. While scanning, keep track of all scanned white pixels until the first black pixel is encountered and stop. Figure 10.6 shows labeled white pixels while scanning from left to right and from top to bottom.

In the next step, histogram features are taken in the same way as in the gradient and structural features using the following computation:

$$f_{ci} = \frac{\text{\# white pixels in the patch that can only be reached when scanning at orientation } i}{\text{\# pixels in the patch}} \quad (10.4)$$

Similarly, the hold feature is defined as

$$f_{hole} = \frac{\text{\#white pixels in the patch that cannot be reached from any orientation}}{\text{\# pixels in the patch}} \quad (10.5)$$



**Fig. 10.6** Labeled *white pixels* while scanning from left to right and from top to bottom. Right-to-left and bottom-to-top scans are omitted. (a) left-to-right scan. (b) top-to-bottom scan

### Directional Element Features

The directional element features [6] are another type of histogram features. First, the size of the character image is normalized to 64 by 64 pixels. The contour is then extracted from the character image. Next, all pairs of neighboring pixels on the contour are categorized into four orientations:  $0^\circ$ ,  $90^\circ$ ,  $45^\circ$  and  $135^\circ$ . Next, the character image is divided into 49 (7 by 7) overlapping patches, and each patch is further divided into four areas A, B, C, and D. A is a 4 by 4 area in the center of the patch. B is the 8 by 8 area in the center of the character image minus area A. C is a 12 by 12 area in the center of the character image minus areas A and B. D is the patch minus areas A, B, and C. The numbers of neighboring pixel pairs at each orientation  $i$  are computed for areas A, B, C, and D, denoted by  $x_i^{(A)}$ ,  $x_i^{(B)}$ ,  $x_i^{(C)}$ , and  $x_i^{(D)}$ , respectively. The directional element feature at orientation  $i$  is defined as

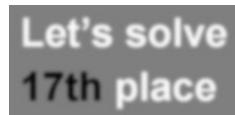
$$x_i = 4x_i^{(A)} + 3x_i^{(B)} + 2x_i^{(C)} + x_i^{(D)} \quad (10.6)$$

The resemblance between gradient features and directional element features is obvious. Gradient features decompose image signal using directions of normal lines on the contour, whereas directional element features decompose image signal using directions of tangent lines on the contour. On the one hand, normal line directions are more informative than tangent line directions since the latter can be derived from the former, but not vice versa. On the other hand, tangent line directions are more appropriate when it is known beforehand that the foreground is not always darker (or lighter) than the background in the text (Fig. 10.7).

### Percentile Features

Percentile features [7] are an interesting variant of histogram features. These features are typically computed on narrow, overlapping slices of character glyphs by considering the distribution of pixel intensities over the vertical extent of the slice. Typically, first and second differences in the percentile features across adjacent

**Fig. 10.7** Video text showing the foreground is not always darker than the background



slices are also used in addition to the raw percentiles themselves. The reader is referred to [7] for a detailed description of percentile features and their computation.

### Gabor Features

The Gabor filter is also one way to decompose the image signal into arbitrary orientations. The Gabor filter is a 2-D linear transform defined as a directed sine wave modulated by a 2-D Gaussian low-pass filter. Gabor features are less sensitive to noise and suitable for degraded, hard-to-binarize document images:

$$h(x, y; \lambda, \phi, \sigma_x, \sigma_y) = \frac{1}{2\pi\sigma_x\sigma_y} \exp \left\{ -\frac{1}{2} \left( \frac{R_1^2}{\sigma_x^2} + \frac{R_2^2}{\sigma_y^2} \right) \right\} \exp \left\{ i \frac{2\pi R_1}{\lambda} \right\} \quad (10.7)$$

where

$$\begin{aligned} R_1 &= x \cos \phi + y \sin \phi \\ R_2 &= y \cos \phi - x \sin \phi \end{aligned} \quad (10.8)$$

$\phi$  and  $1/\lambda$  are the Gabor filter's orientation of interest and frequency of interest, respectively. Usually,  $0^\circ$ ,  $90^\circ$ ,  $45^\circ$ , and  $135^\circ$  are chosen for  $\phi$  and twice the average stroke width in the image is chosen for  $\lambda$  (Fig. 10.8).

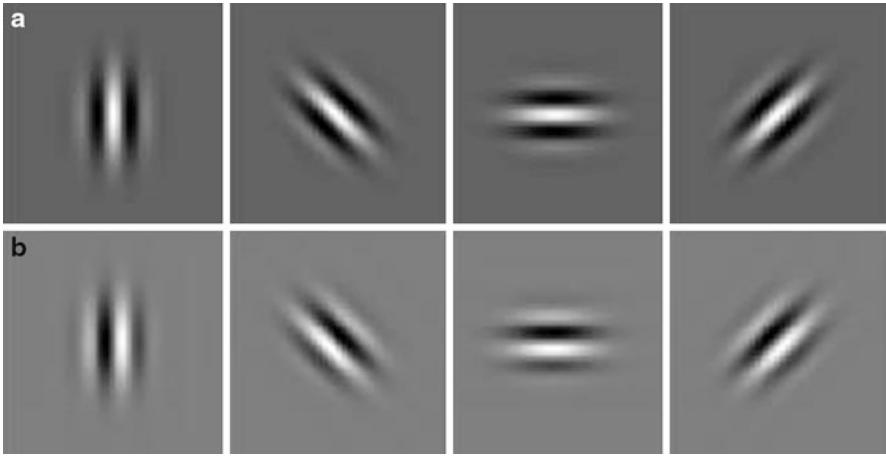
Gabor features for OCR in the literature [24–26] differ in whether the real part, imaginary part, or magnitude is computed, whether averages of positive responses and negative responses of the filter are accumulated separately for each patch, and whether the responses are binarized automatically.

### Character Recognition

Perhaps the simplest character recognizer is one that uses a simple set of features in combination with a distance-based classifier. After dimension reduction using the Principal Component Analysis or Linear Discriminant Analysis, the distance from the projected feature vector of a character image to the trained centroid of each character class is computed. The label of the class that gives the shortest distance is assigned to the character image as the character recognition result.

The MQDF is a modified version of the Quadratic Discriminant Function (QDF) classifier. The Quadratic Discriminant Function (QDF) for each class is defined as

$$f_0^{(l)} = \left( x - \mu^{(l)} \right)^T \left\{ \Sigma^{(l)} \right\}^{-1} \left( x - \mu^{(l)} \right) + \log \left| \Sigma^{(l)} \right| - 2 \log \Pr(\omega^{(l)}) \quad (10.9)$$



**Fig. 10.8** Spatial impulse response of a Gabor filter. The filter is visualized such that *black* represents negative values, the *gray* background represents zero, and *white* represents positive values. **(a)** Real part. **(b)** Imaginary part

where  $x$  is an  $n$ -dimensional feature vector,  $l$  is the class label,  $\mu^{(l)}$  and  $\Sigma^{(l)}$  are the mean vector and covariance matrix of class  $l$ , and  $\log \Pr(\omega^{(l)})$  is the prior probability of class  $l$ .  $\mu^{(l)}$  and  $\Sigma^{(l)}$  are obtained using the maximum likelihood estimation. The QDF can also be rewritten as

$$f_0^{(l)} = \sum_{i=1}^n \frac{1}{\lambda_i^{(l)}} \left\{ \varphi_i^{(l)T} (x - \mu^{(l)}) \right\}^2 + \log \prod_{i=1}^n \lambda_i^{(l)} - 2 \log \Pr(\omega^{(l)}) \quad (10.10)$$

where  $\lambda_i^{(l)}$  ( $\lambda_i^{(l)} \geq \lambda_{i+1}^{(l)}$ ) and  $\varphi_i^{(l)}$  are the  $i$ th eigenvalue and eigenvector of  $\Sigma^{(l)}$ . Several modifications to the QDF in Eq. (10.10) have been proposed in the literature. They differ in their definition depending on whether the objective is to reduce the estimation errors in eigenvectors corresponding to small eigenvalues or to generalize the normal distribution.

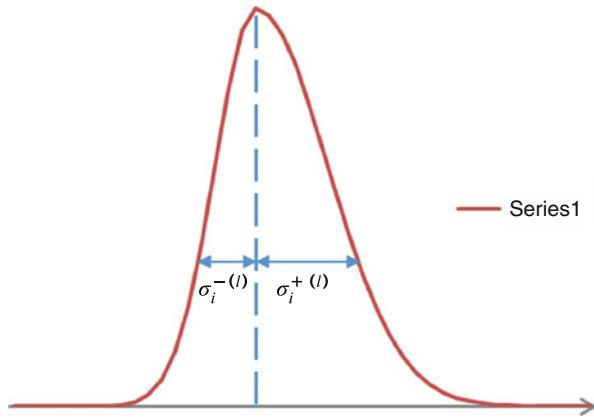
- MQDF for reducing the estimation errors in eigenvectors [5]

$$f^{(l)} = \sum_{i=1}^n \frac{1}{\lambda_i^{(l)} + h^2} \left\{ \varphi_i^{(l)T} (x - \mu^{(l)}) \right\}^2 + \log \prod_{i=1}^n (\lambda_i^{(l)} + h^2) - 2 \log \Pr(\omega^{(l)}) \quad (10.11)$$

where  $h$  is an empirically selected constant or

$$f^{(l)} = \sum_{i=1}^n \frac{1}{\lambda_i'^{(l)}} \left\{ \varphi_i^{(l)T} (x - \mu^{(l)}) \right\}^2 + \log \prod_{i=1}^n (\lambda_i'^{(l)}) - 2 \log \Pr(\omega^{(l)}) \quad (10.12)$$

**Fig. 10.9** Asymmetric distribution assumed by the MQDF defined in Eq. (10.14)



where

$$\lambda_i^{(l)} = \begin{cases} \lambda_i^{(l)}, & \text{if } i \leq k \\ h^2, & \text{if } i > k \end{cases} \quad (10.13)$$

and  $h$  and  $k$  are empirically selected parameters.

- MQDF that generalizes the normal distribution [6]

$$\bullet \quad f^{(l)} = \sum_{i=1}^n \frac{1}{\hat{\lambda}_i^{(l)}} \left\{ \varphi_i^{(l)T} (x - \mu^{(l)}) \right\}^2 + \log \prod_{i=1}^n \hat{\lambda}_i^{(l)} - 2 \log \Pr(\omega^{(l)}) \quad (10.14)$$

where

$$\hat{\lambda}_i^{(l)} = \begin{cases} \left( \sigma_i^{+(l)} \right)^2, & \text{if } \varphi_i^{(l)T} (x - \mu^{(l)}) \geq 0 \\ \left( \sigma_i^{-(l)} \right)^2, & \text{if } \varphi_i^{(l)T} (x - \mu^{(l)}) < 0 \end{cases} \quad (10.15)$$

and  $\left( \sigma_i^{+(l)} \right)^2$  and  $\left( \sigma_i^{-(l)} \right)^2$  are quasi-variances estimated separately for two conditions in Eq. (10.14). Equation (10.14) assumes that the de-correlated signal  $\varphi_i^{(l)T} (x - \mu^{(l)})$  follows the asymmetric, quasi-normal distribution shown in Fig. 10.9. The asymmetric distribution can better model the distribution of the signal which is usually different from the normal distribution.

While the MQDF was reviewed in detail because of the attention it had received in isolated character recognition work, classifiers such as the Gaussian mixture model (GMM), SVM, and neural networks that have achieved advances in later research on handwriting recognition can and have been applied to machine-printed character recognition for finer modeling and better performance at the expense of longer training and decoding time. Indeed, if one were to build an isolated character recognition system today, Bayesian or kernel-based classification techniques would top the list of viable candidates for the classification task.

## Word Recognition

Historically, with the exception of HMM-based approaches, character segmentation and recognition has provided a basis for word recognition. Even in the case of HMM-based techniques, character recognition forms the basis of word recognition. Character segmentation can be improved using the matching score produced during character recognition – of course, improving the matching score requires the design of better features, more effective feature transformations, and more accurate classification approaches. Furthermore, contextual information such as the word lexicon and language model can also be used to further improve word recognition. These techniques will be discussed in this section.

## Character Segmentation

### Features for Finding Segmentation Points

Character images can be identified using white space and connected component analysis when they are well separated. When touching characters exist, features such as local minima in the vertical projection profile [27] can be used to locate optimal locations of character boundaries (Fig. 10.10) ([►Chap. 8](#) (Text Segmentation for Document Recognition)).

### Recognition-Based Segmentation

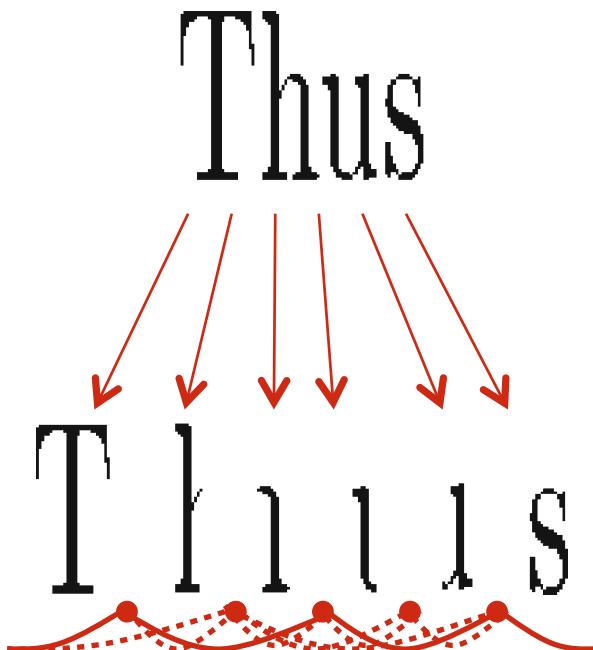
Sequential segmentation of character images without the feedback from recognition is vulnerable to touching characters and ambiguous character boundaries. It contributes a significant fraction to OCR errors when it is applied to document images of omni-font, variable aspect ratio characters.

Recognition-based segmentation creates multiple segmentation hypotheses for a word image and selects the optimal hypothesis using matching scores. Kovalevsky's algorithm [28] is based on a global search on the lattice of all possible segmentation hypotheses for the optimal path where the mean-squared matching error is minimized. Casey's algorithm [29] is based on similar idea. Starting from an initial guess of the boundary of the current character, it gradually moves the right boundary to the left until a meaningful character is found. And then, the same process is repeated for the next character until it either exhausts the set of cut points or every segmented character image matches a character class within a predefined threshold.

In [28] and [29], experimental results on machine-printed documents are reported. In these earliest recognition-based segmentation algorithms, sliding windows are used to implicitly provide sequences of tentative segmentation points. Nearly all later research works [30–32] address segmentation for handwriting recognition rather than machine-printed OCR owing to the fact that handwriting recognition is a more challenging problem and has remained an active area. In most of these works, complex dissection methods are developed to provide

- a c separation of foreground pixels from background. Thus  
 b

**Fig. 10.10** Features for character segmentation. (a) White space between characters. (b) Connected components of a word image. (c) Vertical projection profile of a line image



**Fig. 10.11** Word segmentation using over-segmentation. The optimal segmentation is denoted by the path of solid lines

over-segmentation of word images, and word lexicons are used to limit the size of the search space. The basic steps are summarized as follows (Fig. 10.11):

1. Find sufficiently many segmentation points to include all correct segmentation points. In addition to white space, connected components, and projection profile analysis, concaves in ligatures between characters are also used to locate segmentation points.
2. Select an optimal subsequence out of the segmentation points that minimizes the character sequence matching error using dynamic programming.

Complex dissection methods are not always superior to the implicit, sliding window-based segmentation owing to the fact that the latter does not rely on heuristic rules and can be generalized to text in different languages. The HMM-based OCR can also be thought of as an approach to generate implicit segmentation. Using the HMM, one can automate the process of whole-sentence data labeling and increase the scalability of the OCR system.

## Hidden Markov Model OCR

In the HMM OCR system [7], a text line image is modeled as a left-to-right signal and represented as a sequence of feature vectors sampled at a fixed frame rate (Fig. 10.12). Each frame is further divided horizontally into several patches and image features are computed from each patch. Remember a similar way was adopted to divide a character image into patches in section “GSC Features”. Here, the number of horizontal cuts is still fixed, but the number of vertical cuts is variable.

The HMM-based speech recognition algorithms can be adapted to solve the OCR problem by replacing speech features with the multivariate sequential OCR features. The HMM of a character is a first-order stationary Markov chain of  $n$  emitting states with left-to-right state transition shown in Fig. 10.13. Arrows in Fig. 10.13 indicate nonzero state transition probabilities. The whole sequence of feature vectors is treated as a time series, conforming to the convention in speech recognition. Each vector  $O_t$  is called an observation of the time series at time  $t$ . All feature vectors of each state follow a continuous observational probability distribution modeled as a GMM.

One can build word HMMs in a manner such that the HMM for a word is the linearly connected model of HMMs for all characters in the word. Thus, Fig. 10.13 can also represent a word HMM except that the number of states in the word HMM is  $n$  times the number of characters in the word.

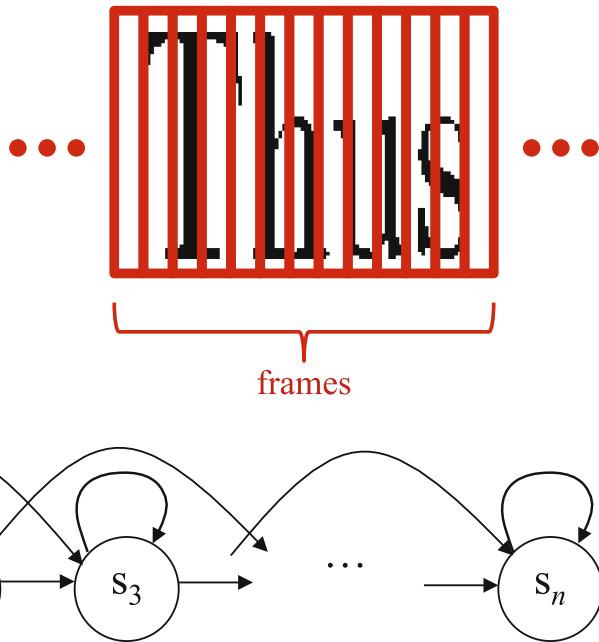
Isolated word recognition can be performed by evaluating word posterior probabilities using word HMMs. One can also use the complex HMM state transition map shown in Fig. 10.9 for continuous word recognition. Each row in Fig. 10.14 denotes a word HMM. The rectangular terminals are non-emitting states that denote the start and end of a character. No observational distribution is associated with non-emitting states.

For a description of how candidate segmentations from an HMM system can be used to combine scores from an HMM system with scores from complex 2-D classification approaches reminiscent of isolated character recognition in a tractable manner, the reader is referred to [33].

## Parameter Learning

Parameters of HMM including Gaussian means, covariances, mixture weights, and transition probabilities are estimated from the training data using the expectation–maximization algorithm known as the Baum–Welch algorithm. In the E-step, the probability of transitioning from state  $i$  to  $j$  at time  $t$   $\Pr(S_t = i, S_{t+1=j} | O, \lambda)$  and the probability for selecting the  $k$ th mixture component in state  $j$  for the observation at time  $t$   $\Pr(S_t = j, M_{t=k} | O, \lambda)$  are estimated for each  $t$ . In the M-step, model parameters  $\lambda$  are reestimated using estimated  $\Pr(S_t = i, S_{t+1=j} | O, \lambda)$  and  $\Pr(S_t = j, M_{t=k} | O, \lambda)$  and the observational sequence  $O$ .

**Fig. 10.12** Frames sampled at constant sample rate using a sliding window



**Fig. 10.13** Left-to-right state transitions in the character HMM

### Recognition

Recognition is the task of finding the optimal hidden state sequence  $s^*$  that produces the observation with maximal posterior probability

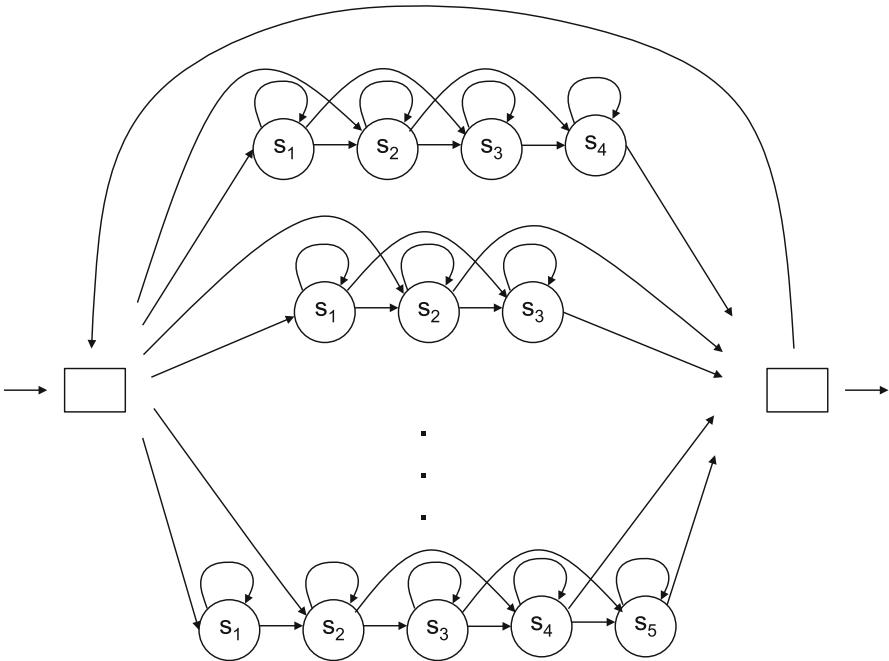
$$s^* = \underset{s}{\operatorname{argmax}} \Pr(s|O, \lambda) \quad (10.16)$$

using the Viterbi algorithm. Beam search can be used to reduce the search space in the Viterbi algorithm and accelerate the decoding process.

### ***n*-Gram Language Model**

The  $n$ -gram language model describes the joint probability that a series of words appear in the language in order  $w_1 \rightarrow w_2 \rightarrow \dots \rightarrow w_N$ . Assuming the probability is a Markov chain, the  $n$ -gram language model can be written as

$$\begin{aligned} \Pr(w_1, w_2, \dots, w_n) &= \Pr(w_1) \Pr(w_2|w_1) \dots \Pr(w_T|w_{T-n+1}, \dots, w_{T-1}) \\ &= \prod_{i=1}^T \Pr(w_i|w_{i-n+1}, \dots, w_{i-1}) \end{aligned} \quad (10.17)$$



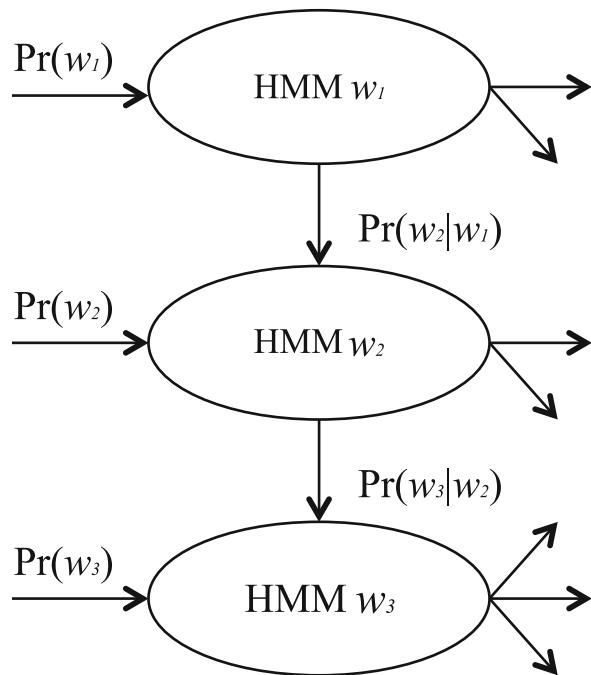
**Fig. 10.14** State transitions of the HMM for continuous word recognition

When the context in the word sequence is considered, the objective of OCR can be formulated as finding the optimal word sequence that produces the maximum posterior probability on the observational sequence:

$$w^* = \underset{w}{\operatorname{argmax}} \Pr(w|O) = \underset{w}{\operatorname{argmax}} \Pr(w) \Pr(O|w) \quad (10.18)$$

$\Pr(w)$  and  $\Pr(O|w)$  are provided by the word model and word recognizer, respectively. As the simplest case, a bigram language model can be integrated into the decoding algorithm using word networks [34] such that the bigram probabilities are treated as between-word transition probabilities shown in Fig. 10.15. Note that this is only applicable to bigram. Integration of trigram and higher-order language models requires complex algorithms that memorize the search history. An alternative is to generate multiple candidate word sequences using bigram in the first pass of decoding and reevaluate these word sequences using a high-order language model in the second pass of decoding.

**Fig. 10.15** Bigram modeled as between-word transition in an HMM word network



## Systems and Applications

### Applications

#### Desktop OCR

Desktop OCR features the use of scanners as the input device, automatic page segmentation, form analysis, multi-language omni-font character recognition, and support of common output file formats. The applications of desktop OCR include but are not limited to automated data entry, digital library, and Internet publishing.

#### Web Services

More and more OCR vendors such as Free Online OCR, OnlineOCR.net, and Google Docs now provide online OCR services as web applications. Users can upload their document images to the vendor's website and receive the OCR results.

#### Camera OCR

The applications of OCR using digital cameras as the input device can be divided into two categories. The first category of OCR applications is based on handheld

cameras or smartphones. These applications include business card, invoice, and bank check recognition. The second category is based on closed-circuit television and is mostly applied to video surveillance including license plate number and freight container number recognition.

## **Commercial Software**

Commercially available software products are available both in the form of end-user applications and as software development kits (SDKs) that allow users to customize the capability to fit their needs. This section presents a brief survey of commercially available OCR capabilities.

### **Omni-Font OCR Software**

The latest commercial OCR software applications are all typically equipped with the capability of reading multiple languages and fonts. For example, FineReader v11 by ABBYY now supports any combination of 189 languages, although dictionary support is only provided for 45 languages. OmniPage v17 by Nuance now supports over 120 languages.

To optimize the user experience, many productivity features are supported by major OCR systems to render the OCR result. These features include retention of original layout and fonts and conversion to prevalent formats including MS Office, WordPerfect, searchable PDF, HTML, and SharePoint.

Below is a list of some of the major OCR vendors in alphabetical order:

- ABBYY
- Nuance
- Presto! OCR
- Presto! PageManager
- Raytheon BBN Technologies Corp.
- Readiris
- Sakhr Inc.

### **SDK and Open-Source OCR**

Many vendors provide SDKs to provide users the ability to customize the OCR software for their proprietary tasks and documents. Here is a list of OCR software applications for which the SDK is available [35], again sorted in alphabetical order:

- ABBYY FineReader
- Aquaforest OCR SDK
- CuneiForm/OpenOCR
- Digital Syphon's Sonic Imagen
- ExperVision TypeReader & RTK
- Indian Scripts OCR
- LEADTOOLS
- NSOCR

- Ocrad
- OmniPage
- PrimeOCR
- Puma.NET
- Raytheon BBN Technologies Corp.
- Readiris
- Sakhr Inc.
- Transym OCR

Tesseract is an open-source library for machine-printed recognition featuring line, word, and character segmentation and character recognition for Latin scripts. It has demonstrated competitive OCR accuracy according to the Annual Test of OCR Accuracy [36].

## Well-Known Evaluations and Contests

The Annual Test of OCR Accuracy was held by the Information Science Research Institute (ISRI) in 1992–1996 to evaluate the performance of participants' OCR systems on text zones from English and Spanish magazines and newspapers. According to their results, the lowest character recognition error rates of all teams were from 0.5 to 5 %, depending on the image quality of the dataset.

Contests aiming at evaluating methods in subareas complementary to character recognition were initiated along with the International Conference on Document Analysis and Recognition (ICDAR). The contests of those subareas include Robust Reading and Text Locating (initiated in 2003) for text detection and recognition of camera-captured scenes, Page Segmentation Competition (initiated in 2003), and Document Image Binarization Contest (DIBCO) (initiated in 2009).

---

## Conclusion

This chapter provides a brief review of some of the salient aspects of the history of machine-printed character recognition techniques, described basic feature extraction and classification techniques for machine-printed character recognition, and presented applications, systems, and evaluations of OCR.

Although there are mature techniques for features and classification of isolated characters, major sources of character recognition errors including text detection and segmentation are still important research topics in the community. OCR of documents that are not compiled in any natural language, e.g., music notes and mathematical equations, are particularly fertile areas for the intrepid researcher as are tasks such as inferring the logical reading order of multicolumn text zones and even data presented in a tabular manner.

Increasing popularity of multimedia and the concomitant rise of smartphones brings opportunity for recognition of text in videos and camera-captured images. It also offers new challenges in the design and extraction of features in

low-resolution, colored images, in the adaptation of models that reuse large volumes of binarized document image training data and in continuous online learning that leverages crowd-sourced information such as opportunistic or targeted annotations and corrections provided by users.

---

## Cross-References

- ▶ [Document Analysis in Postal Applications and Check Processing](#)
  - ▶ [Page Segmentation Techniques in Document Analysis](#)
  - ▶ [Text Segmentation for Document Recognition](#)
- 

## Notes

<sup>1</sup>The description of the historical evolution of OCR is based upon the Wikipedia entry for this topic: [http://en.wikipedia.org/wiki/Optical\\_character\\_recognition](http://en.wikipedia.org/wiki/Optical_character_recognition). The reader is referred to that page for a more detailed review.

<sup>2</sup>Some of the features categorized as concavity features in [8] have nothing to do with stroke concavity. They are derived from the local intensity and whether or not pixels belong to horizontal and vertical strokes.

---

## References

1. Schantz HF (1982) The history of OCR, optical character recognition. Recognition Technologies Users Association, Manchester Center
2. Shahi M, Ahlawat AK, Pandey BN (2012) Literature survey on offline recognition of handwritten Hindi curve script using ANN approach. *Int J Sci Res Publ* 2(5):362–367
3. Niblack W (1986) An introduction to digital image processing. Prentice Hall, Englewood Cliffs
4. Sauvola J, Pietikainen M (2000) Adaptive document image binarization. *Pattern Recognit* 33(2):225–236
5. Kimura F, Takashina K, Tsuruoka S, Miyake Y (1987) Modified quadratic discriminant functions and the application to Chinese character recognition. *IEEE Trans Pattern Anal Mach Intell* 9(1):149–153
6. Kato N, Suzuki M, Omachi S, Aso H, Nemoto Y (1999) A handwritten character recognition system using directional element feature and asymmetric Mahalanobis distance. *IEEE Trans Pattern Anal Mach Intell* 21(3):258–262
7. Natarajan P, Lu Z, Schwartz R, Bazzi I, Makhoul J (2001) Multilingual machine printed OCR. In: Bunke H, Caelli T (eds) Hidden Markov models – applications in computer vision. Series in machine perception and artificial intelligence, vol 45. World Scientific Publishing Company, River Edge, NJ, USA
8. Otsu N (1979) A threshold selection method from Gray-level histograms. *IEEE Trans Syst Man Cybern* 9(1):62–66
9. Kise K, Sato A, Iwata M (1998) Segmentation of page images using the area Voronoi diagram. *Comput Vis Image Underst* 70:370–382
10. O’Gorman L (1993) Document spectrum for page layout analysis. *IEEE Trans Pattern Anal Mach Intell* 15:1162–1173

11. Mao S, Kanungo T (2001) Empirical performance evaluation methodology and its application to page segmentation algorithms. *IEEE Trans Pattern Anal Mach Intell* 23(3): 242–256
12. Lu Y, Tan CL (2003) A nearest neighbor chain based approach to skew estimation in document images. *Pattern Recognit Lett* 24:2315–2323
13. Kapoor R, Bagai D, Kamal TS (2004) A new algorithm for skew detection and correction. *Pattern Recognit Lett* 25:1215–1229
14. Li S, Shen Q, Sun J (2007) Skew detection using wavelet decomposition and projection profile analysis. *Pattern Recognit Lett* 28:555–562
15. Singh C, Bhatia N, Kaur A (2008) Hough transform based fast skew detection and accurate skew correction methods. *Pattern Recognit Lett* 41:3528–3546
16. Zhang Z, Tan CL (2001) Recovery of distorted document images from bound volumes. In: Proceedings of the 6th international conference on document analysis and recognition, Seattle, pp 429–433
17. Cao H, Ding X, Liu C (2003) A cylindrical surface model to rectify the bound document image. In: Proceedings of the 9th IEEE international conference on computer vision, Nice, vol 1, pp 228–233
18. Brown MS, Tsui Y-C (2006) Geometric and shading correction for images of printed materials using boundary. *IEEE Trans Image Process* 15(7):1544–1554
19. Liang J, DeMenthon D, Doermann DS (2008) Geometric rectification of camera-captured document images. *IEEE Trans Pattern Anal Mach Intell* 30(4):591–605
20. Zhang L, Yip AM, Brown MS, Tan CL (2009) A unified framework for document restoration using inpainting and shape-from-shading. *Pattern Recognit* 42(12):2961–2978
21. Miyoshi T, Nagasaki T, Shinjo H (2009) Character normalization methods using moments of gradient features and normalization cooperated feature extraction. In: Proceedings of the Chinese conference on pattern recognition, Nanjing
22. Cristianini N, Shawe-Taylor J (2000) An introduction to support vector machines and other Kernel-based learning methods. Cambridge University Press, Cambridge/New York
23. Favata JT, Srikanth G, Srihari SN (1994) Handprinted character/digit recognition using a multiple feature/resolution philosophy. In: Proceedings of the fourth international workshop frontiers in handwriting recognition, Taipei, pp 57–66
24. Huo Q, Ge Y, Feng Z-D (2001) High performance Chinese OCR based on Gabor features, discriminative feature extraction and model training. *Proc Int Conf Acoust Speech Signal Process* 3:1517–1520
25. Wang X, Ding X, Liu C (2005) Gabor filters-based feature extraction for character recognition. *Pattern Recognit* 38(3):369–379
26. Chen J, Cao H, Prasad R, Bhardwaj A, Natarajan P (2010) Gabor features for offline Arabic handwriting recognition. In: Proceedings of the document analysis systems, Boston, pp 53–58
27. Lu Y (1993) On the segmentation of touching characters. In: Proceedings of the international conference on document analysis and recognition, Tsukuba, pp 440–443
28. Kovalevsky VA (1968) Character readers and pattern recognition. Spartan Books, Washington, DC
29. Casey RG, Nagy G (1982) Recursive segmentation and classification of composite patterns. In: Proceedings of the 6th international conference on pattern recognition, Munich
30. Fujisawa H, Nakano Y, Kurino K (1992) Segmentation methods for character recognition: from segmentation to document structure analysis. *Proc IEEE* 80(8):1079–1092
31. Favata JT, Srihari SN (1992) Recognition of general handwritten words using a hypothesis generation and reduction methodology. In: Proceedings of the fifth USPS advanced technology conference, Washington, DC
32. Sinha RMK, Prasada B, Houle G, Sabourin M (1993) Hybrid recognition with string matching. *IEEE Trans Pattern Anal Mach Intell* 15(10):915–925
33. Natarajan P, Subramanian K, Bhardwaj A, Prasad R (2009) Stochastic segment modeling for offline handwriting recognition. In: Proceedings of the international conference on document analysis and recognition, Barcelona, pp 971–975

34. Fink GA (2007) Markov models for pattern recognition: from theory to applications. Springer, Berlin/Germany, Heidelberg/Germany, New York/USA
35. Comparison of optical character recognition software, Wikipedia page. [http://en.wikipedia.org/  
wiki/Comparison\\_of\\_optical\\_character\\_recognition\\_software](http://en.wikipedia.org/wiki/Comparison_of_optical_character_recognition_software)
36. Rice SV, Jenkins FR, Nartker TA (1995) The fourth annual test of OCR accuracy. In: Proceedings of the annual symposium on document analysis and information retrieval, Las Vegas, Nevada, USA

## **Further Reading**

- Natarajan P, Lu Z, Schwartz R, Bazzi I, Makhoul J (2001) Multilingual machine printed OCR. *Int J Pattern Recognit Artif Intell* 15(1):43–63
- Natarajan P, Subramanian K, Bhardwaj A, Prasad R (2009) Stochastic segment modeling for offline handwriting recognition. In: Proceedings of the international conference on document analysis and recognition, Barcelona, pp 971–975

---

# Handprinted Character and Word Recognition

11

Sergey Tulyakov and Venu Govindaraju

## Contents

Introduction.....	360
History and Importance.....	361
Evolution of the Problem.....	362
Applications.....	363
Main Difficulties.....	364
Summary of the State of the Art.....	366
Preprocessing of Handprinted Character and Word Recognition Texts.....	367
Properties of Handprinted Text.....	367
Binarization.....	368
Representations of Handprinted Text.....	369
Segmentation.....	371
Feature Extraction.....	372
Global Image Features.....	372
Localized Features.....	376
Structural Features.....	377
Recognition Approaches.....	378
Character Recognition.....	378
Character-Based Word Recognition.....	380
Background Modeling.....	382
Combinations of Character and Word Recognizers.....	383
Conclusion.....	384
Cross-References.....	385
References.....	385
Further Reading.....	389

---

S. Tulyakov

Center for Unified Biometrics and Sensors, University at Buffalo, Amherst, NY, USA  
e-mail: [tulyakov@buffalo.edu](mailto:tulyakov@buffalo.edu)

V. Govindaraju

Department of Computer Science & Engineering, Center for Unified Biometrics and Sensors,  
University at Buffalo, Amherst, NY, USA  
e-mail: [govind@buffalo.edu](mailto:govind@buffalo.edu)

**Abstract**

The handprinted texts are produced when the writer tries to emulate some standard printed representation of the characters with the goal to make the written texts legible. Postal address blocks, different fillable forms, or other documents are among the examples of handprinting. Current chapter reviews the main techniques in recognizing handprinted characters and words. It outlines the characteristics of the handprinted texts, such as the stroke structure, distribution of strokes inside character bounding box, and frequently separated characters. Then it investigates how the reviewed techniques address such characteristics of the handprinted texts.

**Keywords**

Binarization • Contour • Dynamic programming • Fillable forms • Handprinted character • Handprinted word • Image histograms • K-nearest neighbor recognizer • Moments • Postal address • Segmentation • Skeleton • Zoning

---

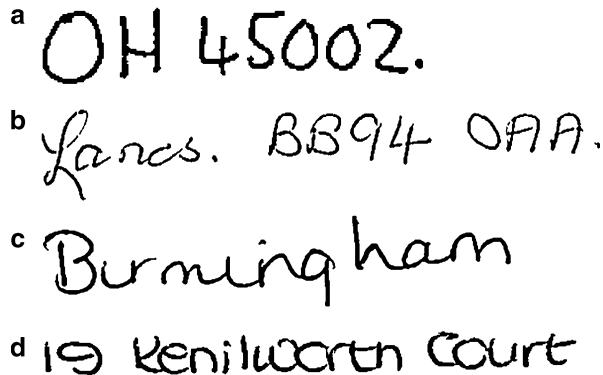
## Introduction

Certain situations might require a person to write some information in a most legible form, easily readable by other people. Such situations include the filling of legal forms, financial and tax documents, postal address information, etc. The person-specific learned cursive writing, while usually easily readable by the writer, might not always be well understood by other persons. Instead, in such situations, a person tries to write the characters resembling some standard printed characters. The handprinted text can be characterized by the characters consisting of a series of separate strokes or arc elements, the frequent separation of characters in a word and the absence of ligatures, and frequent use of uppercase style characters instead of lowercase. Handprinting also differs from machine-printed text by the bigger variability in the shape of written characters, variability in character positions and sizes, the absence of some specific font, and smaller proportion of stroke widths to stroke lengths. The recognition of handprinted characters and words thus represents an intermediate problem between machine-printed text recognition ([►Chap. 10](#) (Machine-Printed Character Recognition)) and handwritten text recognition ([►Chap. 12](#) (Continuous Handwritten Script Recognition)) and generally is more difficult than the first problem and easier than the second.

Figure 11.1 contains few examples of handprinted text images extracted from the address blocks of postal mail pieces. The examples show that the distinction between handprinted and handwritten words is subjective, and same address block can contain both separated and clearly handprinted characters, as well as connected, cursive, and rather handwritten characters.

The terms “handwritten” and “handprinted” sometimes have the same meaning in the research literature, and same recognition methods could be employed for both types of written texts. In this chapter it will be assumed that the recognition methods attempting to perform explicit segmentation of text into characters and

**Fig. 11.1** Handprinted samples from postal address images: The United States zip code (a), The United Kingdom postcode (b), city name (c), and street address (d)



perform recognition of such segmented characters are specific for handprinted text recognition. In contrast, the recognition methods which rely on implicit segmentation of words into characters, or not performing any segmentation at all, are specific for handwritten text recognition problem.

A significant number of works have been published on the topic of handprinted text processing, including extensive reviews [3, 12, 17, 80, 81, 89, 93]. Some of the methods presented in the published literature have universal nature and might be applicable to other types of texts or imagery type. As a result, they might not be very efficient in processing handprinted text. This chapter outlines the main developed approaches and discusses their strength or weaknesses specifically with respect to the handprinted texts.

## History and Importance

The handprinted character and word recognition represent one of the earliest research problems in the image processing and pattern classification fields. The problem development was linked to the appearance of computers, image scanners, and digital tablets, as well as the desire to utilize these new devices for automating certain record-keeping tasks. Due to relatively modest computer hardware requirements and the sufficient diversity of the handprinted character shapes, the area of handprinted character recognition became possibly the most studied area of artificial intelligence field in the years 1980–2000.

The handprinted character recognition problem served as a test case for many pattern recognition and pattern classification algorithms which have been developed in the 1960s–1980s. For example, the early works investigated both syntactical [2, 92] and statistical [5] methods of handprinted character recognition. The syntactical approaches were popular at the beginning due to visual and well-defined nature of features and the sequences of features representing characters. The recognition was performed according to well-formulated syntactical rules and was fairly efficient on a limited hardware of the days. But, with the increased capability of computers,

it was becoming clear that syntactical representation of handprinted characters does not allow the description of all possible character shapes and robust estimation of observation likelihoods. Consequently, the majority of the methods followed the statistical methods for character recognition [5, 91].

The research in this area also underscored the importance of publicly available datasets for training and testing pattern recognition algorithms. The handprinted character datasets collected at NIST [32, 36] and CEDAR [43], as well as handwritten word IAM [69] database, provide the researchers the ability to develop algorithms and to compare their performance to the performance of other researcher's algorithms. ►[Chapter 29](#) (Datasets and Annotations for Document Analysis and Recognition) provides additional references to existing databases.

Finally, the handprinted text recognition algorithms have great value to today's industry. Many applications processing handprinted character and word inputs have been automated, recognizing the handprinted texts in different forms, such as tax forms, postal addresses, and financial documents. Undoubtedly, they will continue to be widely used as long as people transmit the information by means of handprinted text.

## Evolution of the Problem

From the beginning, the development of the handprinted text recognition algorithms depended on the availability of hardware. The early research (1960–1970) was mostly performed only in big companies who could afford some computer time and had access to scanners and touchpad devices. As a consequence, many early algorithms [5, 27, 92] were concerned with hardware limitations and designed algorithms accordingly to address low storage and computing power requirements. Some of the important techniques, such as contour [26] and skeleton representations [10] of character images, feature vector extraction and zoning [5] have originated at that time. Online handwritten character recognition [8] took roots at this time as well (►[Chap. 26](#) (Online Handwriting Recognition)).

The proliferation of less expensive personal computers in 1980–1990 spurred the burst of research activity by universities and other organizations. This time coincided with the apparent peak in handprinted text recognition research. As it was already mentioned, the research has gradually shifted from syntactical matching [77, 92] to statistical approaches [37, 86, 91, 93]. In contrast to syntactical methods, the more advanced type of approaches allowing the descriptive representation of characters, structural methods have appeared. The structural methods could be based on graph matching [62, 98] or variable length structural feature vectors with HMM-based matching [103].

Currently, the interest in the area of handprinted recognition is somewhat diminished, probably due to the increased computing power and the ability to perform other image processing tasks: general object recognition and scene analysis, face and gesture recognition, biometric applications, etc. Still, there is an active research that continues in the processing of non-Latin-based scripts

(see ►Chaps. 13 (Middle Eastern Character Recognition) and ►14 (Asian Character Recognition)), which were previously ignored, as well as the application of machine learning methods developed in other areas for handprinted texts and the development of new applications (medical form processing, digital libraries, etc.).

## Applications

One of the frequent applications of handprinted text recognition is the automated reading of handprinted input fields on different fillable forms [33]. One of the properties of most of these forms helping the recognizers is some limitation on the possible word and character sets. Also, the forms themselves are typically designed so that their reading by the automated algorithms is easy. Some research addresses the topic of designing the forms most easily read by the computers [31].

The examples of form reading applications include the processing of handprinted names, addresses, and other possible information on tax forms [85] or census forms [66]. Recently, some research has been conducted into processing of medical forms [13, 70]. Note that this recent application is significantly more challenging due to less accurate handprinting, low quality of the images (carbon copies), and increased lexicon size. ►Chapter 19 (Recognition of Tables and Forms) contains the examples of different forms and describes specific algorithms for form processing in detail.

Reading of the bank checks is another application which can be easily automated [47, 50, 54, 88]. Two fields in the bank checks, the numerical amount of the check (courtesy) and the handwritten amount (legal), could both be read by handprinted text recognition algorithms, and two read amounts could be verified with each other.

The more challenging application for handprinted text recognition is the processing of postal mail. In contrast to fillable forms, the postal envelopes might not have well-defined address blocks with particular fields, but instead allow free-form writing of the address. Still, due to the specific order of address information and the limited lexicon involved, automated reading of postal addresses has been successfully implemented. As an example, the system for reading the United States postal addresses first tries to recognize numerical ZIP codes and house street numbers, retrieves from the database the street names and city names which have such ZIP codes and house numbers, and verifies the presence of such names in the address block [19, 20, 83]. ►Chapter 21 (Document Analysis in Postal Applications and Check Processing) discusses the postal address and bank check recognition applications in more detail.

The above applications for reading handprinted text have been successfully developed and used in different countries. But, due to the increased use of Internet communications and the replacement of paper documents by their digital versions, it is possible that such applications will have limited use in the future. On the other hand, it is possible that the widespread use of tablet computers will result in some need to process the handprinted text entered on it. In contrast to text written on paper and later scanned as digital image (*off-line handwriting*),

the digital handwriting devices capture the trajectory of pen movements (analogous to skeleton representation, see section “[Representations of Handprinted Text](#)”) along with timestamps of each drawn point (*online handwriting*). Such information is inherently more precise, and specifically designed online handwriting recognition methods (refer to ▶ [Chap. 26](#) (Online Handwriting Recognition)) typically deliver better performance than off-line handwriting recognition methods considered in this chapter.

## Main Difficulties

Given high-quality images of a well-separated characters or character strings, the current recognizers are able to produce nearly perfect recognition results. Most of the errors appear when the input consists of low-quality, poorly segmented, or just ambiguous images. Therefore, the main problem of handprinted text recognition research might be outside the scope of character and word recognizers, but with the preprocessing and noise removing algorithms. Still, the efficient recognizer should be able to overcome some of the difficulties and possibly verify the results of preprocessing algorithms.

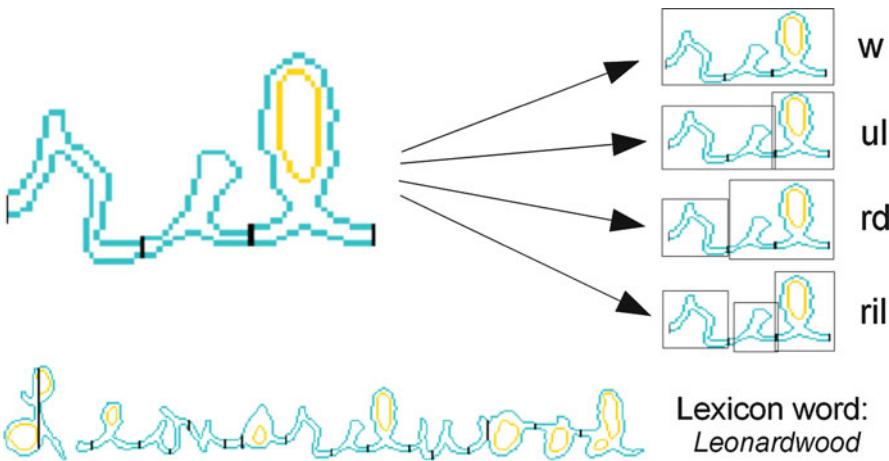
### 1. *Touching or Intersecting Characters or Extraneous Strokes*

Humans can relatively easily read the texts containing, for example, intersecting characters or crossed-out characters. Intuitively, the extraneous information from the character image is getting automatically discarded during the reading process. Most of the current character recognizers do not attempt to discard any such information, and the features are extracted from the whole image. Although the presence of true character’s features mitigates the effect, and the character could be recognized correctly in such situations, the inability to account for non-character information introduces an uncertainty in the recognition results and leads to further errors in document analysis.

### 2. *Low-Quality Images*

The low image quality expresses itself in broken components of binarized characters, missing structural elements due to too light or too dark image parts, extraneous noise elements, etc. Some of the recognition algorithms might try to reconstruct the strokes, merge together broken components, and so on. The reconstruction of the strokes might make sense if, for example, it was known beforehand that the large percentage of images contains broken strokes. Usually, researchers will look at the samples of the datasets and decide whether such reconstruction is needed. But given other datasets with no broken strokes, such modifications might introduce errors in the images and result in poorer recognition rate.

The difficulty in processing low-quality images and the images containing extraneous information seems to be inherent in current recognition algorithms. Instead of trying to connect together the low-level features, build a structural representation of characters, and match these structural representations with stored prototypes, a simpler approach based on feature vectors and discriminating



**Fig. 11.2** Ambiguous appearance of the segmented subimage might lead to four possible recognition results. Using the lexicon word and the original image to match all the characters in the word, the recognizer deduces that the subimage should be matched to “rd”

these vectors in the feature space dominates the research field. The current methods based on structural feature extraction [103] might not have the desired precision to find and represent structural features sufficiently well, and as a result, they are generally outperformed by the statistical methods.

### 3. Ambiguous Characters

Although the goal of handprinting is to make the text as clear as possible, different factors, such as haste, inattention, and physical difficulties, could influence the legibility of written images. Some characters could also be more similar to each other than to other characters due to the presence of common structural elements and the ways of writing them. The segmentation choices can also add to the ambiguities during reading. Figure 11.2 has an example of segmented image which could be recognized in different ways.

This problem is difficult to solve since the characters seem to not have a uniform distribution in the image space, and the confusion among recognition results is common. One of the practical solutions is to use some external knowledge about the image. Figure 11.2 presents an example of how the segment corresponding to two letters “rd” is recognized using the knowledge about lexicon word “leonardwood” and the matches between its characters and other segments of the image. The ambiguity problem is so common that most word recognizers rely on either the word lexicon or some grammatical model during recognition. Since the general texts can have very large lexicons, the recognition of such texts still presents a large problem.

### 4. Performance Evaluation

The research into handprinting recognition is somewhat hampered by the difficulty of properly evaluating the algorithms and comparing their performance

to the algorithms of other researchers. The reasons for this difficulty include the absence of common benchmarks (the training and testing sets are not defined even for publicly available datasets), the easiness of available datasets (it might be difficult to compare algorithms if they have the recognition rate of 99 % or more), and the lack of detailed description of the published algorithms [81]. It might be even difficult to compare the recognition algorithms operating on the same benchmarks [65], since the results of recognition could be greatly influenced by the preprocessing techniques, feature extraction, or postprocessing additions. As an example, the k-nearest method has been shown to have better performance than multilayer perceptron on the features obtained by Karhunen-Loëve transform [37] but worse performance on other features [63]. ►Chapter 30 (Tools and Metrics for Document Analysis Systems Evaluation) has additional discussion on the possible benchmarks and algorithm performance evaluation methods.

## Summary of the State of the Art

It is reasonable to say that the area of handprinted text recognition research has mostly matured to the application stage, and a significant portion of the development is done by the commercial companies, such as BBN and Siemens. The current systems could implement the recognition of the scripts from different languages and with large lexicons or with the grammar models defined by the recognized language. The reported error rates could be sufficiently small for the successful reading of postal addresses or government forms. The extensive review of some current commercial and university recognizers can be found in [81].

Most current algorithms for handprinted and handwritten word recognition seems to be based on *Hidden Markov Models* (HMM). The HMMs offer more transparent derivation of the matching scores than dynamic programming-based approaches (see section “[Character Based Word Recognition](#)”) and could be more easily trained with the word images. HMM-based word recognition could be *segmentation-free* or it could incorporate some type of presegmentation of the image into candidate characters [22]. It is not quite clear whether segmentation free or segmentation-based algorithms have better performance; the reviews argue for both first [81] and second [99] kinds.

The segmentation-free HMMs can incorporate the HMM-based character recognizers which are trained as parts of word HMM. Most other methods incorporate explicitly trained character recognizers. The methods for extracting features and recognizing characters vary significantly. Although discriminative methods, such as support vector machines (SVM), might show better performance for separate character recognition [65], the use of character recognizers inside HMM might require a proper probabilistic measures, e.g., likelihood, and generative classification methods, such as *Gaussian Mixture Models* (GMM).

## Preprocessing of Handprinted Character and Word Recognition Texts

► [Chapters 4](#) (Imaging Techniques in Document Analysis Processes), ► [5](#) (Page Segmentation Techniques in Document Analysis), and ► [8](#) (Text Segmentation for Document Recognition) of this book have already addressed general methods for preprocessing the document images, including binarization, noise removal, and segmentation of lines and words. The current section underlines some of the important methods which specifically deal with handprinted texts, in contrast to other text types such as machine-printed and cursive handwriting.

### Properties of Handprinted Text

The existing methods dealing with handprinted texts can be evaluated on how well they utilize the inherent properties of such texts. Therefore, let us define the properties of handprinted texts, which could be useful for recognition algorithms.

1. The image pixels come from two separate classes: background (paper) and foreground (ink). The colors of background and foreground pixels are generally different.
2. The areas of background and foreground pixels, as well as the boundary between them, have some smoothness constraints.
3. The width of the foreground areas, which can be defined as an average distance from the foreground pixel to the closest background pixel, is determined by the pen width and should remain relatively constant for a handwritten document. The stroke width is usually small relative to the character size for handprinted documents.
4. The characters are composed of a small number of structural elements: strokes, arcs, and dots. Each character class has its own typical set of such elements arranged in a specific order.
5. Each character occupies a limited region of space, and the structural elements of a character are somewhat uniformly distributed inside its bounding box.
6. The characters can be separate from each other or have some stroke elements, *ligatures*, connecting them.
7. The characters in a handprinted word have similar sizes, with some variations possible depending on the character class.

The above properties of handprinted texts are similar to the properties of handwritten texts. The possible difference lies in the better separation of characters, more prominent nature of structural elements inside each character, and more uniform placement of these elements inside character's bounding box. The handprinted texts are also less writer specific, and while handwritten text recognizers might need to be adjusted for a particular writer's style, the recognizers of handprinted texts

could be trained in writer-independent manner. These properties also differ from the properties of machine-printed texts, which could have greater similarity between characters of the same class, the presence of vertical and horizontal strokes, greater separability between classes, and particular font characteristics, e.g., serif.

The coarticulation effect is another property of the cursive writing, which could be possibly accounted for during the recognition [80]. The effect consists in the appearance changes of characters with respect to the neighboring characters. For example, the upper or lower ligature from a previous character will influence the appearance of current character. It could be assumed that the coarticulation effect is less pronounced in handprinted texts than in the cursive handwritten ones and that handprinted text recognizers could omit it from consideration.

If not all properties are utilized by the particular approach, then it might not be optimal and performance will suffer. For example, the character recognition algorithms which operate on grayscale character images, such as image moment methods, do not assume usually clear boundary between foreground and background. As a result, their performance is typically worse than the performance of algorithms explicitly locating this boundary and extracting features based on it [86].

## Binarization

Due to the first property of handprinted text, foreground, and background pixels, the binarization, that is the assignment of pixels to one of these two classes, could rely on the difference in distributions of the grayscale pixel values of these two classes. By counting the numbers of pixels in the image with different intensity, a histogram of pixel intensities could be constructed. The foreground pixels should contribute to the peak of the histogram near black pixel intensity, and the foreground should form a peak near white pixel intensity. Simple thresholding of the histogram, e.g., at the lowest point between two peaks, will separate the image pixels into two classes. For better performance, some prior distributions of pixel intensities can be assumed along with prior class probabilities as it was done by Otsu [75]. One of the possible modification of thresholding technique is to perform multiple candidate binarizations with different thresholds and then estimate, whether the obtained binary images satisfy some properties of handprinted text [104]. Sezgin and Sankur [84] provides a review of different thresholding methods, including adaptive methods which restrict histograms to the local neighborhood of a currently thresholded pixel.

The traditional thresholding methods do not account for the smoothness of foreground and background regions or their boundary; consequently they can be applied in many research areas with the same success as in handprinted character binarization. The methods, which are more suitable for processing handprinted characters, should take into account the assumption of the smoothness of these regions and their boundary. The application of smoothing filters during binarization

will result in smoother boundaries between binarized regions and more closely match the “true” boundaries of characters.

Further developments in binarization techniques for handprinted characters include the use of filters accounting for the edges between foreground and background areas [76], the filters depending on the width of the strokes and thus well suitable to preserve the stroke structure [48], and, finally, the filters which assume the possible changes in stroke directions [73] and which preserve the connections between different strokes. As one can see, these techniques increasingly take into account the properties of handprinted texts and are bound to improve the performance of the text processing systems.

The binarization algorithm should preferably learn the correspondence between the true or prototype binary character and the grayscale image from training data and utilize this correspondence during image processing. Statistical modeling methods try to create the models of prototype binary images and their mappings to grayscale images. Markov Random Fields (MRF) could be used for such modeling. In [13] the MRF consists of rectangular grid of prototype binary image patches,  $x_{i,j}$ . Two types of dependencies are learned using the training data: the probabilities of having some particular patch given a set of neighboring patches,  $P(x_{i,j}, \{x_{i+\delta_i, j+\delta_j}\}_{\delta_i, \delta_j \in \{-1, 0, 1\}})$ , and the probabilities of observing a grayscale patch  $y_{i,j}$  for the underlying prototype patch,  $P(y_{i,j}, x_{i,j})$ . The binarization of grayscale image composed of the patches  $\{y_{i,j}\}$  consists in finding the set of prototype patches  $\{y_{i,j}\}$  which maximize both sets of probabilities. The presented algorithm has the desired properties of being able to learn from the training samples the possible local prototype shapes as well as the appearance of these shapes in the images. Thus, it implicitly takes into consideration the first four properties of the handprinted texts proposed before. The experiments confirm the superior performance of this method over non-trainable thresholding techniques, especially for low-quality handwritten text images.

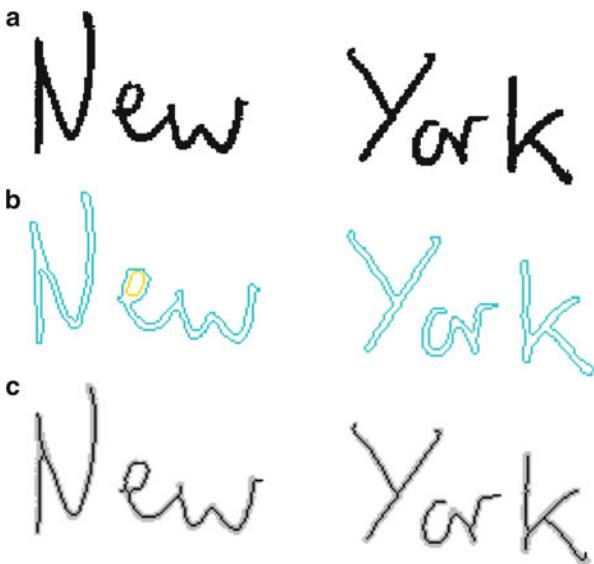
## Representations of Handprinted Text

The binary character image is one possible representation of the handprinted character, and many character recognition algorithms use it to extract features. But there are other representations which could be useful for feature extraction, specifically contours and skeletons.

Freeman [26, 27] proposed using the chain codes for representing arbitrary curves. The curves on the image are the sequences of connected pixels, and in order to store these sequences in the computer storage, only the direction of next pixel with respect to previous one can be recorded. Each of 8 directions is encoded by 3 bits, and the total curve has around  $3^n$  bits. Such encoding will be more effective than the traditional run-length encoding of the binary images for images containing few connected components.

Toussaint and Donaldson [92] traced the outer contours of the characters and used the positions of the local extrema points to represent a character as a

**Fig. 11.3** Possible representations of handprinted texts: binary image (a), contour set (b), and skeleton (c)



short sequence of local extrema positions. The distance between two characters can be calculated as minimum edit distance between these sequences. As this work exemplifies, the contour representation of characters has a sequential or one-dimensional structure and allows different methods for extracting features or matching than two dimensional binary representation. For example, the works cited in section “[Contour Moments](#)” construct moments using one-dimensional contour functions, which arguably are more effective in encoding character shapes than the two-dimensional moments discussed in section “[Image Moments](#).”

The third possible representation of the handprinted character shapes is based on the notion that they were created by a pen movements, and it is possible to derive the trajectory of the center of the pen from it. There are different definitions and names for this central trajectory of the pen, with possibly one of the first one being the *medial axis* [10]. The most popular and simple methods of obtaining the medial axis, or *skeleton*, of the character are based on the *thinning* process, in which the boundary pixels of the shape are iteratively removed following some predetermined rules [1, 7, 38]. Figure 11.3 gives an example of the three possible representations discussed; the skeleton extraction method is based on [7].

Both contour and skeleton representations of characters allow the extraction of directional features or curvature features in form of slope of contour or skeleton interval. Although it is possible to extract such features from binary images or even original grayscale images, contour and skeletons make the process simpler. They also make simple the detection of the possible strokes or arcs or the end points. The skeletons also allow easy segmentation methods and the detection of the stroke intersections.

## Segmentation

► [Chapter 8](#) (Text Segmentation for Document Recognition) provided an overview of general segmentation methods applicable to different types of texts. As it was mentioned in this chapter, one of the properties of the handprinted texts is the frequent separation of characters, and possibly an easier approach to segmentation can be used. Still, as Fig. 11.1 illustrates, it is typical for handprinted texts to have at least some of the characters to be connected or touched. The segmentation of handprinted texts, therefore, is a necessary step during recognition. Although, it would be desirable to have a recognizer, which would spot or recognize characters or words directly from the unsegmented texts, apparently such recognizers do not exist or do not have satisfactory performance yet.

General two approaches to character segmentation exist. One approach performs an explicit segmentation procedure and tries to understand the structure of the character elements near the segmentation point. The other approach is performing segmentation implicitly during word recognition; possible segmentation points or separation boundaries could have a large variety, and a particular segmentation point is determined by the best scores of recognized characters surrounding it. Cursive word recognition algorithms, such as HMMs, typically rely on this method. The set of possible segmentation boundaries could be simply represented by vertical lines, sets of two points in upper and lower contours, or by a single point in a skeleton representation of the word. Casey and Lecolinet [14] contains a review of segmentation methods along with their more detailed categorization.

The explicit segmentation algorithms are probably more powerful, since they try to utilize the knowledge about specific structures of handprinted texts. Fujisawa et al. [28] considered different possibilities on how two digits could touch each other, e.g., the stroke end of one digit is touching the middle of a stroke or arc of another digit and similar possible scenarios. Depending on the contour profiles, these scenarios could be separated, and specific segmentation could be performed for each scenario. Madhvanath et al. [67] considered another set of features, such as the closeness of upper and lower contours, the closeness of both of them to the baseline, and the separation of ascenders and descenders. The segmentation points obtained with this method are shown in Fig. 11.6.

It might be also possible to utilize some machine learning techniques to separate the true segmentation points from false ones. For example, [58] uses a neural network accepting as input set features extracted from a small region around candidate segmentation point and verifies whether this is valid segmentation point.

There are also some works which do not perform the segmentation of touching characters but try to classify the sets of touching characters. For example, Ciresan [18] is training a convolutional neural network to recognize the pairs of digits. The digit string recognition system tries to match a particular component to a single digit or to the pair of touching digits and selects the better match as recognition result. The technique is clearly limited since it would not allow the recognition of three touching digits. Also, it is doubtful that this algorithm has

better performance than explicit segmentation algorithms utilizing the patterns of connections in touching characters.

The words in handprinted texts are usually well separated, and one could face the problem of determining the boundaries of the words and distinguishing them from the possibly separated characters inside the words. This problem is somewhat easier than the segmentation of the word into characters. The algorithms performing word segmentation could use such information as typical sizes of the characters, intervals between characters, or intervals and lengths of strokes or ligatures [79].

---

## Feature Extraction

Current section presents some techniques in extracting features from character images. The literature on this topic is numerous and additional analysis can be found in some survey works [3, 12, 17, 72, 93]. The discussion of this section emphasizes two major directions for feature extraction: global and local. Global features are extracted from the whole character images by some unified approach, and they are usually complementary to each other. For example, the moments are extracted by the same approach and might reflect different frequencies or directions of moment functions. Whereas each moment taken separately has little meaning, the full set of moments can be used to reconstruct the original character image. Local features are typically extracted in some local neighborhood, or a zone, of the character image; the full feature vector for a character is a concatenation of feature vectors extracted for each zone. The features themselves are usually more simple in this case. They might be an average stroke direction, a histogram of gradients, the bits accounting for the presence of a stroke end or a corner in the zone, etc. Overall, one can hypothesize that the approaches extracting local features are more powerful since they exploit the characteristics of handprinted text to a greater degree.

## Global Image Features

### Projection Histograms

One of the simplest and possibly earliest methods of extracting the features includes the counting of foreground pixels at different horizontal or vertical pixel rows of the character image [55]. These numbers form the projection graphs, or histograms, which can be used as features themselves or could serve for extracting some secondary features. Although it might be proved that the set of projection histograms taken for different angles is sufficient to reconstructing original character image similar to the reconstruction method of computer tomography (CT) scans, the applications in character recognition do not go so far and might use only a limited projection set for features.

The projections are typically used for segmentation methods rather than for recognition. As an example of using projections for recognition, Wang et al. [100] construct the projections originating from the center of moments of the character for different angles. The Fourier moments are then extracted from the projection

histogram and used as features for character recognition. In general, the set of projections does not seem to reflect many properties of the handprinted characters, and therefore this type of features might not have good performance, and as a result, was used rarely for recognition.

### Image Moments

The image of character can be represented as a function in two-dimensional space:  $I(x, y)$ , where  $x$  and  $y$  are the coordinates of the pixels and  $I$  represents the value of the intensity of the pixel (e.g., it can be assumed that the value 0 represents white pixel and 1 represents black). The set of functions defined on the same image area  $A$ , e.g.,  $L^2(A)$ , forms a vector space with inner product:

$$\langle f_1, f_2 \rangle = \iint_{x,y \in A} f_1(x, y) f_2(x, y) dx dy.$$

For any orthonormal basis  $\{v_1, \dots\}$  in this vector space, the image function can be expressed as a linear sum of basis functions:

$$I(x, y) = \sum_i a_i v_i(x, y), \text{ where } a_i = \iint_{x,y \in A} I(x, y) v_i(x, y) dx dy.$$

Then, the original image function  $I(x, y)$  can be approximated by the sum of first  $N$  terms:

$$\|I(x, y) - \hat{I}_N(x, y)\| \xrightarrow{N \rightarrow \infty} 0, \text{ where } \hat{I}_N(x, y) = \sum_{i=1}^N a_i v_i(x, y).$$

Depending on the choice of the orthonormal basis, different approximations of the image function  $I(x, y)$  are possible. The multiple proposed methods of this type usually choose the basis functions of some predetermined form. Possibly one of the first works on applying the methods of basis function decomposition for character images by Hu [41] considered polynomial basis functions of increasing orders, and extracted coefficients were called *moments*.

$$M_{ij} = \iint_{x,y \in A} x^i y^j I(x, y) dx dy$$

The polynomials  $x^i y^j$  in the above formula do not represent an orthonormal basis, and thus, the reconstruction of the original image by means of moments  $M_{ij}$  is not straightforward. Nevertheless, they can be used as features for character classifiers as any other mapping of image pixel values into single values. As an example of the usefulness of these features, the centroid of the image is expressed by first-order moments  $M_{10}$  and  $M_{01}$ , and the direction of the principal axis of the image can be expressed by means of second-order moments,  $M_{20}$ ,  $M_{11}$ , and  $M_{02}$  [41]. The Gram-Schmidt orthonormalization procedure can convert the basis of moments

into the orthonormal basis of Legendre polynomials, and the reconstruction of the image function will be simpler in this case [90]. But it is not certain if the coefficients obtained with Legendre polynomials will have any advantage over moments when used as features for character recognition. Effectively, the new coefficients are obtained by the algebraic transformation of moments and thus have similar discriminating power as moments. Also, the formulas for calculating moments are simpler, and this might be the reason why moments are used more frequently than Legendre polynomial coefficients.

The other type of orthonormal functions which were extensively studied for character recognition are the Zernike polynomials, and the corresponding coefficients were named *Zernike moments* [4, 6, 51, 52, 90]. The Zernike polynomials are defined on the circle area and can be expressed with the help of polar coordinates [51]:

$$V_{nm}(x, y) = V_{nm}(\rho, \theta) = R_{nm}(\rho)e^{im\theta},$$

where  $n$  and  $m$  are integers,  $n \geq 0$ ,  $|m| \leq n$ ,  $n - |m|$  is even and

$$R_{nm}(\rho) = \sum_{s=0}^{(n-|m|)/2} (-1)^s \frac{(n-s)!}{s! \left(\frac{n+|m|}{2} - s\right)! \left(\frac{n-|m|}{2} - s\right)!} \rho^{n-2s}.$$

Intuitively,  $R_{nm}(\rho)$  represents an oscillating function along radial direction with the number of oscillations increasing with  $n$ , and term  $e^{im\theta}$  represents angular oscillations. Polynomials with larger  $n$  and  $m$  reflect finer details of the image. Figure 11.4 illustrates the approximation of character images by the Zernike moments of different orders (the order is the largest  $n$  of participating Zernike polynomials). The polynomials of orders 10–15 are able to approximate the original character images to a sufficient degree, and thus, the corresponding coefficients can be used as features for character recognition. Note that the number of coefficients is bigger than the order of moments, e.g., polynomials of order 10 have 36 coefficients.

Note that other constructions of orthonormal basis functions are possible, which might be well applicable to character recognition. For example, [49] investigates similarly defined on circle area, orthogonal Fourier-Mellin moments and asserts that they outperform Zernike moments.

One of the frequently cited advantages for traditional moments and Zernike moments is that they could be used to construct the scale and rotation invariants [51, 90]. Such property would be useful if character images had random orientations, as, for example, characters on topographic maps. But for many available datasets and applications, the character orientation is given. For such well-oriented characters the performance of moment invariants is worse than the performance of original moments [4, 51].

The important consideration in the choice of basis functions for character recognition is their ability to approximate the character images with possibly fewer terms. The visual inspection of moment polynomials suggests that these functions

**Fig. 11.4** Original character images and their representation using Zernike moments of orders 5, 10, 15, and 20



can account for the variations in different character images, and the experiments on character recognition confirm that moments, especially Zernike moments, can have sufficiently good performance. But, the construction of the moment polynomials does not depend on the training character sets and might not use the particular handprinted character properties which have been outlined before. For example, the moments do not consider the possibly small stroke width relation to the character size; the character “e” in Fig. 11.4 arguably has better approximation than character “A” due to bigger stroke width. The moments also do not pay attention to structural elements of the characters, strokes, and arcs. At the same time, due to smoothness of polynomial functions, they could represent the similarly smooth foreground regions of the characters quite well. Note that the moments can be used not only for recognizing character images but also any arbitrary smooth shapes and have been used well in these other applications.

### Karhunen-Loëve (KL) Transform

Instead of using predefined functions for image decomposition, one might want to learn the orthonormal basis from the training data. Karhunen-Loëve (KL) transform considers the training images as points in  $N = n_1 * n_2$  dimensional space ( $n_1$  and  $n_2$  are the dimensions of the images) and finds a basis  $\{v_1, \dots, v_N\}$  which minimizes the approximation errors of representing training images by their projections into  $k$ -dimensional subspace spanned by the first  $k$  basis vectors:

$$\min \left| \left| \sum_i I_i - \hat{I}_i^k \right| \right|, \text{ where } \hat{I}_i^k = \sum_{j=1}^k a_{ij} v_j \text{ and } a_{ij} = \langle I_i, v_j \rangle.$$

The solution to this problem is given by the eigenvectors, or principal components, of covariance matrix constructed with the help of training images  $I_i$ . The KL transform has been proposed [97] and used extensively for face detection and face recognition.

The use of KL transforms for handprinted character recognition has also been investigated. Grother [35] and Grother and Candela [37] report satisfactory results

on using it for handprinted digit classification. They also suggest that around 50,000 training samples are needed to overcome the difference in the performance on training and testing sets and thus to achieve a full potential of this method of feature extraction. The algorithm has been subsequently released in NIST form processing software [33].

But, when compared to other methods of feature extraction, the KL transform is not performing as well, and consequently it was used rarely for character recognition. The reason for this might be the great variety of character shapes, which KL transform fails to model properly. Indeed, it assumes that the character images occupy some linear subspace in the space of all possible two-dimensional images. Whereas for face images this assumption seems to hold, this is not the case for character images. For example, the average or mean face, used during KL transform, is usually well-defined image representing somewhat average person's face. But the mean image of digits [35] is just an obscure blob with no structure.

### Contour Moments

As it was mentioned in section “[Representations of Handprinted Text](#),” the contour representation of the character is equivalent to the binary representation but has a one-dimensional nature instead of two dimensional. Consequently, instead of constructing two-dimensional moments of the image, the one-dimensional moments of the contours could be used. In one of the works utilizing this method, Cheng and Yan [16] used one-dimensional function of the distance of contour points from the centroid of the image and extracted Fourier coefficients of this function. Clearly, it is possible to use other one-dimensional functions of contours for deriving features. Contour profile technique looks at either  $x$  or  $y$  coordinate function of the contour points, and extracts moments from these functions. These and similar shape representation methods are reviewed in [72].

Although the contour moment techniques could be more efficient in representing characters and deriving features than image moment methods, essentially they have the same strengths and drawbacks with respect to utilizing the properties of handwritten characters – they account for smoothness of foreground regions and boundaries, but not for other properties. Therefore, their performance most probably will be worse than the performance of other methods.

### Localized Features

One of the main ideas in character recognition is splitting the area of the character image into some subimages and determining the presence of particular features in each of these subimages. Trier et al. [93] uses the term *zoning* for this technique and assumes that most feature extraction methods have zoning varieties. The extraction of features from image zones begins from earliest works on character recognition. Bakis et al. [5] searched for the presence of 4-pixel configurations, roughly representing edges and angles of the characters, in 6 zones of the character images.

Tou and Gonzalez [91] looked for the structural elements, such as strokes arcs or loops, in eight segments of character image.

The technique continues to be used widely in for handprinted character and word recognition. For example, different combinations of gradient and structural binary or float number features are extracted in either 9 or 16 zones of contour representation of character in [86] and used for neural network classification algorithm. The performance of this and similar algorithms is very close to the optimal, and misclassified cases cause the difficulties for human experts as well. The character and character-based word recognizers utilizing this technique have been successfully used for real-life applications [83]. The hierarchical zoning technique is investigated in [46], where the zones possessing foreground regions are expanded to a greater degree and serve to calculate more features. There is some research addressing the problem of finding optimal zoning configurations from the training data [45].

The technique utilizes the property of the uniform distribution of structural elements of characters inside their bounding box and the usual separation of such features. For example, when a  $3 \times 3$  zoning is used for digit “0,” one might get 8 zones on the boundary detecting a presence of some structural elements, say arcs, and each zone will have only one such element detected. Thus, there is little confusion between detected features, and the overall structure of “0” is well represented by the extracted zone features. Without zoning technique, it is difficult to account for the fifth property of handprinted texts proposed in section “[Properties of Handprinted Text](#).”

## Structural Features

The structural features reflect the elements constituting the handwriting: strokes, arcs, end points, intersections and corners of them, loops, ascenders and descenders, etc. The use of these elements provides possibly a shortest description of a character. Earlier approaches utilized the strings of structural elements for syntactical matching [91]. The assumption for these approaches is that the sequences of elements follow a particular well-defined grammar, and each character follows its own grammar rules. It might also be possible to define the structural elements through the composition of more primitive elements, such as chain code elements or the sides of a polygon approximating the boundary, thus expanding the grammar.

The weaknesses of approaches relying on structural features include the possibly complex structure of characters resulting in complex grammars, the imprecision in extracting the structural elements, and the difficulty in accounting for variations in the positions of the elements. Some of these difficulties were addressed in the later approaches representing the structural elements in the graph and performing elastic matching of the graphs [62, 98].

Elastic matching accounts for the variations in the positions of the elements. But the detection of structural elements can contain errors and should be addressed for better performance as well. Xue and Govindaraju [103] perform the matching of

structural elements with the help of HMMs, which cast the problem of imprecise matching into statistical framework. HMMs have two possible ways to account for such variations: by the variations in the explored paths of the hidden states and by the variability of the observed elements for the same hidden state.

Still, the most effective way to utilize the structural elements is by recording their presence in different zones. For example, Favata and Srikantan [25] construct a vector of bits which are set to 1 if a particular structural element (line, corner, or concavity) is present in 1 of 16 zones of the character image. In this case, it is possible to obtain a fixed length feature vector and perform matching by calculating the distance between two vectors or by applying classification algorithm.

---

## Recognition Approaches

### Character Recognition

The character recognition presents a sufficiently difficult problem for machine learning and pattern classification algorithms to continue being in the focus of interest for a research community. The number of the classes is relatively large: 10 classes for digit recognition, 26 or 52 for alphabetic characters, and 36 or 62 for alphanumeric. Note that the English alphabet is implied here; some classes can be grouped together due to little difference in their appearance (“0” and “O,” “1,” “i,” and “l”). Each character might exhibit a significant difference in how it is written by different people at different times, and, at the same time, different characters might have very similar appearance which is frequent source of confusion even for human readers. These properties lead to relatively big intra-class and small interclass distances between extracted feature vectors and hinders the correct classification. Some learning algorithms can also experience difficulties due to unequal numbers of training samples, which is caused by different frequencies of characters in the texts.

The choice of the classification method depends on the nature of extracted features. The set of structural features could have variable size and discrete values which reflect important connections between more primitive features. The classification approaches which are suitable for such features include syntactical matching [77], decision trees [87], elastic matching of graphs [98], and HMMs [103]. The methods in this category could have complex implementations with many parameters requiring heuristic adjustments [87] in addition to using the training samples.

Other types of features, including moments and localized features, take a form of a fixed-length feature vector. For such features, a variety of classification methods operating in N-dimensional feature space could be applied. The methods in this category are generally simpler than methods for matching structural features, and generic implementations of classification algorithms could be utilized with no changes. The parametric classification methods represent the classification decision boundaries or the class densities by some functions depending on parameters and search for the parameters best satisfying training data. As an example, Naive

Bayes classification assumes a normal distribution of class densities with diagonal covariance matrix and the variances determined by the strengths of the features; the classification is done according to Bayes decision rule. Parametric methods typically require a small amount of training data and have been used in earlier handprinted character classification works [5]. The disadvantage of parametric methods is the limited form of decision boundary and, consequently, the possible inability to correctly classify complex patterns.

Whereas the decision surface functions for parametric methods have a predetermined and limited form, the nonparametric methods are more flexible and could approximate the arbitrary decision surface with the help of the large number of training samples. The frequently used nonparametric methods include nearest neighbor classifiers [25], Bayesian classification by modeling the densities of the character classes [39], neural networks [29, 60], and support vector machines (SVM) [61, 65]. Given the improved performance of computers, the nonparametric classifiers became dominant for handprinted character recognition.

Nearest neighbor methods [21] present one of the simplest and possibly most frequently used classification methods for handprinted character recognition. Grother and Candela [37] compare the performance of few types of nearest neighbor, neural network, linear, and quadratic classifiers on the handprinted digit classification problem, where the features are extracted by KL transform. Overall, the nearest neighbor classifiers, as well as probabilistic neural networks operating on similar principle, showed better performance than all other classifiers on this task. As an example of such recognizer, consider the character recognizer based on k-nearest neighbor matching [25]. It records the presence of gradient, structural, and concavity features in  $4 \times 4$  zones of the character images as a binary 512-dimensional feature vector. The calculation of the distance between tabulated training samples and unknown sample gives bigger weights to set bits in both feature vectors. The variety of distance calculation methods and the methods to increase the matching have been proposed as well [9].

The k-nearest neighbor classification method has obvious drawbacks – large training template storage space and long matching time – but these drawbacks are becoming less important with increased computing hardware capabilities. Its advantages include the ability to perform well with large dimensional feature vectors and non-floating point features [25]. But its main advantage is probably the ability to account well for the diversity of handprinted character shapes: if a particular character shape is not yet tabulated and incorrectly recognized by current training samples, it could be added to the training set, thus allowing future correct classification of similar shapes. The error rate of k-nearest neighbor classifier approaches the error rate of the optimal Bayes classifier with the increased number of training samples. In general, it might be more effective to increase the size of the training set, say by 50%, than to search for other classification methods in order to achieve the desired error rate reduction. Note that the proper feature extraction is necessary for well performing k-nearest neighbor classifier; if trained directly on the image pixels or low level features, neural networks and SVMs might achieve a better performance [61, 64, 65].

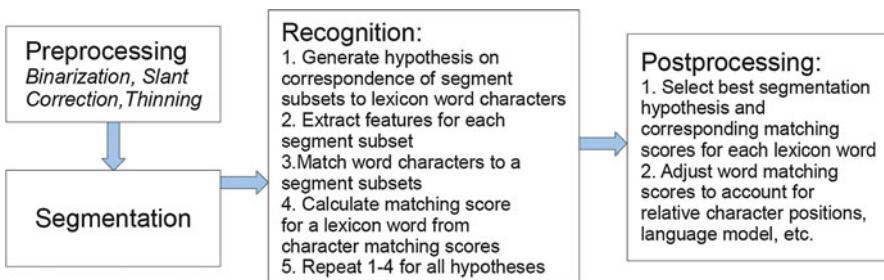
## Character-Based Word Recognition

The words are composed from characters, and it is natural to build the word recognition system on the basis of character recognizers. Although it might be possible to perform word recognition based on some global features [68], such as the number of detected strokes, loops, ascenders, or descenders, the performance of such recognizers would be limited. One of the possible approaches to word recognition is presented on Figure 11.5. It is the approach based on the oversegmentation of word image – few consecutive segments are combined together to generate a hypothetical character image which is matched by included character recognizer. All possible combinations of segments are considered and matched to appropriate characters from lexicon words.

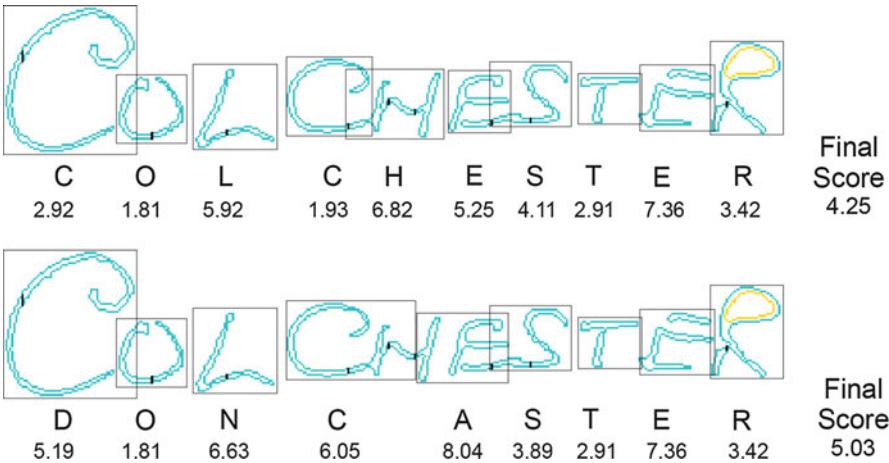
Many word recognition algorithms follow this workflow [11, 15, 22, 24, 53, 64, 71, 78, 99]. The numerical strings can also be processed by the similar method [74]. Even if the algorithm does not have an explicit segmentation step, it might perform an implicit segmentation. Some of the HMM-based word recognition approaches [71, 82] use vertical lines to separate the observation frames, which directly corresponds to the segmentation by the same lines.

One of the techniques, which most word recognizers employ, is the use of dynamic programming matching methods resulting in the speedup in the search for best fitting correspondence of segments and characters [53]. The technique basically keeps in memory the results of matching the consecutive subsequences of segments to the subsequences of characters in the lexicon words. Then, the matching of longer subsequences of segments will utilize the results of previous matchings of shorter subsequences. If an HMM is used in word recognizer, a forward-backward algorithm of the same complexity is employed for matching.

In general, there is little difference in the explicit dynamic programming word recognizers [53] and HMM-based recognizers [22]. Both of the approaches could be trained to recognize separate characters and combine the results of character matches into word recognition result. It might be possible to utilize similar features in both approaches, and the time and memory requirements are the same. Probably,



**Fig. 11.5** General workflow of a segmentation-based handprinted word recognition algorithm



**Fig. 11.6** Two results of matching the word image to lexicon words “Colchester” and “Doncaster” by segmentation-based word recognizer [53]. The character matching scores represent the distances from the feature vectors extracted from character subimages to the nearest prototype feature vectors of corresponding training characters. The final matching score is the root mean square of character scores

the main difference appears during the training procedure – an HMM training implicitly accounts for finding best segmentation points separating characters with the objective that segmented characters are most easily recognized, but the segmentation points in the dynamic programming algorithms are determined by the external segmentation algorithm. The characteristics of the segmentation point are automatically included in the matching scores for HMM, but dynamic programming algorithms should incorporate this information explicitly. It is not well ascertained if one type of recognizer performs better than another, and the possible comparison results [71] might be greatly influenced by the different choices of preprocessing steps, features, or matching score calculations in two types of recognizers.

Figure 11.6 presents the results of matching the same image to the two lexicon words by the word recognizer of dynamic programming type [53]. The figure also provides the matching scores for the characters in these words and the segment assignments to the characters. The interesting theoretical question arises with regard to these results: what is the optimal way to combine character scores into word recognition scores? The Bayesian approach suggests that the matching score for a lexicon word should be constructed in form of the geometric mean

$$S \cong \sqrt[k]{\prod_{m=1,\dots,k} s_m}$$

if the character matching scores  $s_m$  are proportional to the likelihood of matching character  $c_m$  to the corresponding subimage:  $s_m \cong p(\text{subimage}_m | c_m)$  [94].

HMM approaches usually follow the above formula for matching score calculation, but the dynamic programming algorithms might not. As in example of Fig. 11.6, the word matching score is the root mean square of character matching scores. It turns out that taking root mean square for this word recognizer is equivalent to constructing geometric mean of character likelihoods, but even better results could be achieved by training character recognizer to deliver scores proportional to  $p(\text{subimage}_m | c_m)$  [94].

## Background Modeling

The last two properties of handprinted texts outlined in section “[Properties of Handprinted Text](#)” describe the properties of character subimages inside the word image: there is frequently a space or connecting stroke from one character to the next, and the sizes and positions of characters in the same word should be proportional. Most of the word recognizers do not consider the relative sizes and positions of characters, and the final matching score is constructed only from the matching results of each character [53]. The relative sizes and positions of characters can be represented as additional matching scores accounting for the connections between neighboring characters, e.g., for two consecutive letters “a” and “b” of the lexicon word, and for the currently matched subimages of these letters,  $\text{sub}_a$  and  $\text{sub}_b$ , one might derive the connection probability  $P(\text{"ab"} | \text{sub}_a \cup \text{sub}_b)$  and use this probability in word recognizer’s score [30].

The distances between characters are partially accounted for during the segmentation procedure for segmentation-based approaches and during the matching for HMM-based approaches. But, more information could be obtained from, for example, the size of the gap between characters or some other distance measures between them. The gap information might be essential during recognition of phrases, where the gaps between words are more pronounced than the gaps between characters. El-Yacoubi et al. [23] introduce *space HMM* to account for the spaces between the words in an HMM-based phrase recognizer. Park and Govindaraju [79] consider a more detailed and explicit gap treatment in a segmentation-based phrase recognizer, accounting for both inter-character and interword gaps and modifying the matching scores depending on the relationships between gaps and average distances between characters.

The property of character geometry relative to other characters in a handprinted word has also got the attention of researchers. The simple bounding box relationships such as “the average height of character ‘f’ should be around 1.5 of the average character height” have been considered in [102]. Gader et al. [30] investigated a more complicated method of extracting special *transition* and *bar* features and using a neural network to find the likelihood of two neighboring characters of particular types, such as “ascender, ascender” and “ascender, descender.” But the reported performance was similar to a simpler approach relying only on the information about character bounding boxes.

## Combinations of Character and Word Recognizers

One of the usual techniques to improve the performance of handprinted character and word recognition system is to incorporate the matching results of few recognizers. Preferably, the combined recognizers would utilize different features extracted from the images and thus will have complementary information useful for fusion. Usually, the character and word recognizers produce match confidences for a finite number of characters or lexicon words, and, therefore, they could be considered as classifiers. Thus, the problem of combining character and word recognition results can be cast as a general problem of classifier combinations [57]. Some of the frequently used combinations methods include voting techniques [59], Dempster-Shafer theory [101], or a variety of combination rules, such as sum, product, min, and max rules [56].

It is possible to distinguish three types of recognition outputs: a single choice of matching class, a ranking order of classes, or a matching score or confidence assigned to each class [101]. The combination methods could deal only with one type of such outputs, e.g., voting methods use single choices of classifiers, Borda count method accepts rank information, and combination rules use matching scores. Since the matching scores deliver the most information about the recognition results, the use of combination rules [56] is generally favored by the researchers. The choice of a particular combination rule, e.g., sum or product, depends on the assumptions on what the matching scores really mean. For example, if the product rule was optimal for the scores of some recognizer, then the sum rule would have been optimal on the logarithms of those scores. As a result, the use of combination methods typically involves some procedure to normalize the matching scores, i.e., to convert them to some prespecified form such as the posterior probability of the class [44]. The efficient normalization procedure might require some analysis on the strength of particular recognizer on a given class set [34].

Despite the large number of published works in the classifier combination field, there is still no consensus on the optimal or, at least, the best performing approach for combining the results of character and word recognizers. For example, it was noted [40] that the rank-based combination methods can have better performance than the matching score combination methods, which apparently utilize more available information. Thus, the conversion of the matching scores to the ranks and performing some rank combinations, e.g., by Behavior Knowledge Space method [42], could potentially give better results than trying to utilize matching scores with combination rules [56].

The recent results in classifier combination research pinpoint the problem to the dependencies of matching scores assigned to different classes [96]. For example, a character recognizer might produce high matching scores for a good quality image and character classes “a,” “d,” and “o,” but low matching scores for a poor quality image and same character set. If the score for “a” is higher than the scores for other characters in both situations, then the rank information will correctly classify image as “a.” But the high matching score for incorrect character “d” in the first

case and low matching score for correct character “a” in the second case might confuse the combination algorithm and result in misclassification. Thus, the optimal combination algorithm should include the model of how the matching scores were generated, but it seems that it is hard to construct such model for each combined recognizer. Instead, it is possible to use some statistical measures derived from the sets of matching scores assigned to different classes during combination in order to account for such dependencies and to achieve a better performance than either rank or traditional rule-based approaches [95].

---

## Conclusion

The reviewed methods of feature extraction and character and word recognition are among the most typical and frequently used, but the list is surely not complete. The written character and word recognition research area is one of the most popular and a great number of works have been published on the topic. The different techniques of preprocessing, feature extraction, and recognition could be combined in many ways, and performance of existing algorithms could be usually improved in multiple ways.

Among this variety, it should be important to understand what makes a particular algorithm weak or strong. In order to do it, this chapter tried to relate the reviewed algorithms to the properties of the handprinted texts. The processing of handprinted texts seems to favor the algorithms which extract the stroke information, use structural features, and utilize the locality of features. The handprinted word recognition could be facilitated by having some character segmentation algorithm, which learns the typical connection patterns between characters. It is also important to have some method for verifying the relative sizes of found characters and their positions.

Many published algorithms fail to take into account all the properties of handprinted texts and thus probably do not have the optimal performance. One of the reasons might be the complexity of implementation and not guaranteed performance improvements. For example, in order to extract some structural features, like a presence of strokes, the algorithm has to binarize image, convert it to chain code representation or extract skeleton, and apply some dedicated algorithms determining if a sequence of points in the chain code or skeleton represents a stroke. Such process is more difficult than, for example, extracting moment features or performing KL transform. The construction of the segmentation algorithm, either heuristic or trainable, is even more complicated. But the utilization of simple properties, such as character size and position distributions, is easy to implement and should be utilized more frequently.

There seems to be an increasing tendency to use advances in machine learning algorithms instead of heuristic processing and feature extraction of handprinted images. The kernel methods, such as support vector machines, can be used for feature extraction or for classification, with the input consisting of original or downsampled pixel image. For word recognition, the use of HMMs seems to become a mainstream approach. The advantage of these methods is the precise

accounting for the variability in the training data, and it is possible that they would be able to construct proper internal representation of characters and account for the presented properties of handprinted texts.

---

## Cross-References

- [A Brief History of Documents and Writing Systems](#)
  - [Continuous Handwritten Script Recognition](#)
  - [Document Analysis in Postal Applications and Check Processing](#)
  - [Imaging Techniques in Document Analysis Processes](#)
  - [Page Segmentation Techniques in Document Analysis](#)
  - [Recognition of Tables and Forms](#)
  - [Text Segmentation for Document Recognition](#)
- 

## References

1. Ahmed M, Ward R (2002) A rotation invariant rule-based thinning algorithm for character recognition. *IEEE Trans Pattern Anal Mach Intell* 24(12):1672–1678
2. Anderson RG (1968) Syntax-directed recognition of hand-printed two-dimensional mathematics. PhD thesis, Harvard University
3. Arica N, Yarman-Vural FT (2001) An overview of character recognition focused on off-line handwriting. *IEEE Trans Syst Man Cybern Part C Appl Rev* 31(2):216–233
4. Bailey RR, Srinath M (1996) Orthogonal moment features for use with parametric and non-parametric classifiers. *IEEE Trans Pattern Anal Mach Intell* 18(4):389–399
5. Bakis R, Herbst NM, Nagy G (1968) An experimental study of machine recognition of hand-printed numerals. *IEEE Trans Syst Sci Cybern* 4(2):119–132
6. Belkasim SO, Shridhar M, Ahmadi M (1991) Pattern recognition with moment invariants: a comparative study and new results. *Pattern Recognit* 24(12):1117–1138
7. Bernard TM, Manzanera A (1999) Improved low complexity fully parallel thinning algorithm. In: Proceedings of the international conference on image analysis and processing, Venice, 1999, pp 215–220
8. Bernstein MI (1964) Computer recognition of on-line, hand-written characters. Technical report RM-3753-ARPA, The RAND Corporation
9. Bin Z, Srihari SN (2004) Fast k-nearest neighbor classification using cluster-based trees. *IEEE Trans Pattern Anal Mach Intell* 26(4):525–528
10. Blum H (1967) A transformation for extracting new descriptors of shape. In: Wathen-Dunn W (ed) Models for the perception of speech and visual form. MIT, Cambridge, pp 362–380
11. Bozinovic RM and Srihari SN (1989) Off-line cursive script word recognition. *IEEE Trans Pattern Anal Mach Intell* 11(1):68–83
12. Bunke H, Wang PS-P (1997) Handbook of character recognition and document image analysis. World Scientific, Singapore
13. Cao H, Govindaraju V (2009) Preprocessing of low-quality handwritten documents using Markov random fields. *IEEE Trans Pattern Anal Mach Intell* 31(7):1184–1194
14. Casey RG, Lecolinet E (1996) A survey of methods and strategies in character segmentation. *IEEE Trans Pattern Anal Mach Intell* 18(7):690–706
15. Chen MY, Kundu A, Srihari SN (1995) Variable duration hidden Markov model and morphological segmentation for handwritten word recognition. *IEEE Trans Image Process* 4(12):1675–1688

16. Cheng D, Yan H (1998) Recognition of handwritten digits based on contour information. *Pattern Recognit* 31(3):235–255
17. Cheriet M, Kharma N, Liu CL (2007) Character recognition systems: a guide for students and practitioners. Wiley-Interscience, Hoboken
18. Ciresan D (2008) Avoiding segmentation in multi-digit numeral string recognition by combining single and two-digit classifiers trained without negative examples. In: 10th international symposium on symbolic and numeric algorithms for scientific computing (SYNASC'08), Timisoara, pp 225–230
19. Cohen E, Hull JJ, Srihari SN (1991) Understanding text in a structured environment: determining zip codes from addresses. *Int J Pattern Recognit Artif Intell* 5(1–2):221–264
20. Cohen E, Hull JJ, Srihari SN (1994) Control structure for interpreting handwritten addresses. *IEEE Trans Pattern Anal Mach Intell* 16(10):1049–1055
21. Dasarathy BV (1991) Nearest neighbor (NN) norms : NN pattern classification techniques. IEEE Computer Society, Los Alamitos
22. El-Yacoubi A, Gilloux M, Sabourin R, Suen CY (1999) An HMM-based approach for off-line unconstrained handwritten word modeling and recognition. *IEEE Trans Pattern Anal Mach Intell* 21(8):752–760
23. El-Yacoubi MA, Gilloux M, Bertille JM (2002) A statistical approach for phrase location and recognition within a text line: an application to street name recognition. *IEEE Trans Pattern Anal Mach Intell* 24(2):172–188
24. Favata JT (1996) Character model word recognition. In: Fifth international workshop on frontiers in handwriting recognition, Essex, pp 437–440
25. Favata JT, Srikanth G (1996) A multiple feature/resolution approach to handprinted digit and character recognition. *Int J Imaging Syst Technol* 7(4):304–311
26. Freeman H (1961) On the encoding of arbitrary geometric configurations. *IRE Trans Electron Comput EC-10(2):260–268*
27. Freeman H (1974) Computer processing of line-drawing images. *ACM Comput Surv* 6(1):57–97. 356627
28. Fujisawa H, Nakano Y, Kurino K (1992) Segmentation methods for character recognition: from segmentation to document structure analysis. *Proc IEEE* 80(7):1079–1092
29. Fukushima K (2003) Neocognitron for handwritten digit recognition. *Neurocomputing* 51(0):161–180
30. Gader PD, Mohamed M, Chiang J-H (1997) Handwritten word recognition with character and inter-character neural networks. *IEEE Trans Syst Man Cybern Part B Cybern* 27(1): 158–164
31. Garris MD, Dimmick DL (1996) Form design for high accuracy optical character recognition. *IEEE Trans Pattern Anal Mach Intell* 18(6):653–656
32. Garris MD, Wilkinson RA (1992) Handwritten segmented characters NIST special database 3, National institute of standards and technology, Gaithersburg, Maryland, USA
33. Garris MD, Blue JL, Candela GT, Dimmick DL, Geist J, Grother PJ, Janet SA, Wilson CL (1994) NIST form-based handprint recognition system, Technical Report NISTIR 5469, National institute of standards and technology, Gaithersburg, Maryland, USA.
34. Govindaraju V, Slavik P, Xue H (2002) Use of lexicon density in evaluating word recognizers. *IEEE Trans Pattern Anal Mach Intell* 24(6):789–800
35. Grother P (1992) Karhunen Loëve feature extraction for neural handwritten character recognition. *Proc SPIE* 1709(1):155
36. Grother PJ (1995) NIST special database 19-handprinted forms and characters database, National institute of standards and technology, Gaithersburg, Maryland, USA
37. Grother PJ, Candela GT (1993) Comparison of handprinted digit classifiers. (U.S.), National institute of standards and technology, Gaithersburg, Maryland, USA
38. Guo Z, Hall RW (1992) Fast fully parallel thinning algorithms. *CVGIP Image Underst* 55(3):317–328
39. Hinton GE, Dayan P, Revow M (1997) Modeling the manifolds of images of handwritten digits. *IEEE Trans Neural Netw* 8(1):65–74

40. Ho TK, Hull JJ, Srihari SN (1994) Decision combination in multiple classifier systems. *IEEE Trans Pattern Anal Mach Intell* 16(1):66–75
41. Hu M-K (1962) Visual pattern recognition by moment invariants. *IRE Trans Inf Theory* 8(2):179–187
42. Huang YS, Suen CY (1995) A method of combining multiple experts for the recognition of unconstrained handwritten numerals. *IEEE Trans Pattern Anal Mach Intell* 17(1):90–94
43. Hull JJ (1994) A database for handwritten text recognition research. *IEEE Trans Pattern Anal Mach Intell* 16(5):550–554
44. Ianakiev K, Govindaraju V (2002) Deriving pseudo-probabilities of correctness given scores. In: Chen D, Cheng X (eds) *Pattern recognition and string matching*. Kluwer, Dordrecht/Boston
45. Impedovo S, Lucchese MG, Pirlo G (2006) Optimal zoning design by genetic algorithms. *IEEE Trans Syst Man Cybern Part A Syst Hum* 36(5):833–846
46. Park J, Govindaraju V, Srihari SN (2000) OCR in a hierarchical feature space. *IEEE Trans Pattern Anal Mach Intell* 22(4):400–407
47. Jayadevan R, Kolhe S, Patil P, Pal U (2011) Automatic processing of handwritten bank cheque images: a survey. *Int J Doc Anal Recognit* 15(4):1–30
48. Kamel M, Zhao A (1993) Extraction of binary character/graphics images from grayscale document images. *CVGIP Graph Models Image Process* 55(3):203–217. 167588
49. Kan C, Srinath MD (2002) Invariant character recognition with Zernike and orthogonal Fourier-Mellin moments. *Pattern Recognit* 35(1):143–154
50. Kaufmann G, Bunke H (2000) Automated reading of cheque amounts. *Pattern Anal Appl* 3(2):132–141
51. Khotanzad A, Hong YH (1990) Invariant image recognition by Zernike moments. *IEEE Trans Pattern Anal Mach Intell* 12(5):489–497
52. Khotanzad A, Hong YH (1990) Rotation invariant image recognition using features selected via a systematic method. *Pattern Recognit* 23(10):1089–1101
53. Kim G, Govindaraju V (1997) A lexicon driven approach to handwritten word recognition for real-time applications. *IEEE Trans Pattern Anal Mach Intell* 19(4):366–379
54. Kim G, Govindaraju V (1997) Bank check recognition using cross validation between legal and courtesy amounts. *Int J Pattern Recognit Artif Intell* 11(4):657–674
55. Kirsch RA, Cahn L, Ray C, Urban GH (1958) Experiments in processing pictorial information with a digital computer. In *Papers and discussions presented at the December 9–13, 1957, eastern joint computer conference: Computers with deadlines to meet*. ACM: Washington, D.C 221–229
56. Kittler J, Hatef M, Duin RPW, Matas J (1998) On combining classifiers. *IEEE Trans Pattern Anal Mach Intell* 20(3):226–239
57. Kuncheva LI (2004) *Combining pattern classifiers: methods and algorithms*. Wiley Inter-Science, Hoboken
58. Kurniawan F, Rehman A, Mohamad D (2009) From contours to characters segmentation of cursive handwritten words with neural assistance. In: *International conference on instrumentation, communications, information technology, and biomedical engineering (ICICI-BME)*, Bandung, 2009, pp 1–4
59. Lam L, Suen CY (1995) Optimal combinations of pattern classifiers. *Pattern Recognit Lett* 16(9):945–954
60. Le Cun Y, Jackel LD, Boser B, Denker JS, Graf HP, Guyon I, Henderson D, Howard RE, Hubbard W (1989) Handwritten digit recognition: applications of neural network chips and automatic learning. *IEEE Commun Mag* 27(11):41–46
61. LeCun Y, Jackel LD, Bottou L, Cortes C, Denker JS, Drucker H, Guyon I, Muller UA, Sackinger E, Simard P, Vapnik V (1995) Learning algorithms for classification: a comparison on handwritten digit recognition. In: Oh JH, Kwon C, Cho S (eds) *Neural networks: the statistical mechanics perspective*. World Scientific, Singapore, pp 261–276
62. Lee RST, Liu JNK (2003) *Invariant object recognition based on elastic graph matching: theory and applications*. IOS, Amsterdam/Washington, DC

63. Liu C-L, Nakashima K, Sako H, Fujisawa H (2003) Handwritten digit recognition: benchmarking of state-of-the-art techniques. *Pattern Recognit* 36(10):2271–2285
64. Liu C-L, Sako H, Fujisawa H (2004) Effects of classifier structures and training regimes on integrated segmentation and recognition of handwritten numeral strings. *IEEE Trans Pattern Anal Mach Intell* 26(11):1395–1407
65. Liu C-L, Fujisawa H, Marinai S (2008) Classification and learning methods for character recognition: advances and remaining problems. In: Marinai S, Fujisawa H (eds) *Machine learning in document analysis and recognition*. Volume 90 of studies in computational intelligence. Springer, Berlin/Heidelberg, pp 139–161
66. Madhvanath S, Govindaraju V, Srihari SN (1996) Reading handwritten phrases on U.S. census forms. *Int J Imaging Syst Technol* 7(4):312–319
67. Madhvanath S, Kim G, Govindaraju V (1999) Chaincode contour processing for handwritten word recognition. *IEEE Trans Pattern Anal Mach Intell* 21(9):928–932
68. Madhvanath S, Kleinberg E, Govindaraju V (1999) Holistic verification of handwritten phrases. *IEEE Trans Pattern Anal Mach Intell* 21(12):1344–1356
69. Marti UV, Bunke H (2002) The IAM-database: an English sentence database for offline handwriting recognition. *Int J Doc Anal Recognit* 5(1):39–46
70. Milewski R, Govindaraju V, Bhardwaj A (2009) Automatic recognition of handwritten medical forms for search engines. *Int J Doc Anal Recognit* 11(4):203–218
71. Mohamed M, Gader P (1996) Handwritten word recognition using segmentation-free hidden Markov modeling and segmentation-based dynamic programming techniques. *IEEE Trans Pattern Anal Mach Intell* 18(5):548–554
72. Loncaric S (1998) A survey of shape analysis techniques. *Pattern Recognit* 31(8):983–1001
73. Oh I-S (1995) Document image binarization preserving stroke connectivity. *Pattern Recognit Lett* 16(7):743–748
74. Oliveira LS, Sabourin R, Bortolozzi F, Suen CY (2002) Automatic recognition of handwritten numerical strings: a recognition and verification strategy. *IEEE Trans Pattern Anal Mach Intell* 24(11):1438–1454
75. Otsu N (1979) A threshold selection method from gray-level histograms. *IEEE Trans Syst Man Cybern* 9(1):62–66
76. Palumbo PW, Swaminathan P, Srihari SN (1986) Document image binarization: evaluation of algorithms. *Proc. SPIE* 697:278–285
77. Parizeau M, Plamondon R (1995) A fuzzy-syntactic approach to allograph modeling for cursive script recognition. *IEEE Trans Pattern Anal Mach Intell* 17(7):702–712
78. Park J (2002) An adaptive approach to offline handwritten word recognition. *IEEE Trans Pattern Anal Mach Intell* 24(7):920–931
79. Park J, Govindaraju V (2002) Use of adaptive segmentation in handwritten phrase recognition. *Pattern Recognit* 35(1):245–252
80. Plamondon R, Srihari SN (2000) Online and off-line handwriting recognition: a comprehensive survey. *IEEE Trans Pattern Anal Mach Intell* 22(1):63–84
81. Plötz T, Fink G (2009) Markov models for offline handwriting recognition: a survey. *Int J Doc Anal Recognit* 12(4):269–298
82. Senior AW, Robinson AJ (1998) An off-line cursive handwriting recognition system. *IEEE Trans Pattern Anal Mach Intell* 20(3):309–321
83. Setlur S, Lawson A, Govindaraju V, Srihari S (2002) Large scale address recognition systems truthing, testing, tools, and other evaluation issues. *Int J Doc Anal Recognit* 4(3):154–169
84. Sezgin M, Sankur B (2004) Survey over image thresholding techniques and quantitative performance evaluation. *J Electron Imaging* 13(1):146–168
85. Srihari SN, Shin Y-C, Ramanaprasad V, Lee D-S (1996) A system to read names and addresses on tax forms. *Proc IEEE* 84(7):1038–1049
86. Srikantan G, Lam SW, Srihari SN (1996) Gradient-based contour encoding for character recognition. *Pattern Recognit* 29(7):1147–1160
87. Suen CY, Nadal C, Legault R, Mai TA, Lam L (1992) Computer recognition of unconstrained handwritten numerals. *Proc IEEE* 80(7):1162–1180

88. Suen CY, Lam L, Guillevic D, Strathy NW, Cheriet M, Said JN, Fan R (1996) Bank check processing system. *Int J Imaging Syst Technol* 7(4):392–403
89. Tappert CC, Suen CY, Wakahara T (1990) The state of the art in online handwriting recognition. *IEEE Trans Pattern Anal Mach Intell* 12(8):787–808
90. Teague MR (1980) Image analysis via the general theory of moments. *J Opt Soc Am* 70(8):920–930
91. Tou JT, Gonzalez RC (1972) Recognition of handwritten characters by topological feature extraction and multilevel categorization. *IEEE Trans Comput C-21(7)*:776–785
92. Toussaint GT, Donaldson RW (1970) Algorithms for recognizing contour-traced handprinted characters. *IEEE Trans Comput C-19(6)*:541–546
93. Trier ØD, Jain AK, Taxt T (1996) Feature extraction methods for character recognition—a survey. *Pattern Recognit* 29(4):641–662
94. Tulyakov S, Govindaraju V (2001) Probabilistic model for segmentation based word recognition with lexicon. In: 6th international conference on document analysis and recognition (ICDAR 2001), pp 164–167, Seattle. IEEE Computer Society
95. Tulyakov S, Govindaraju V (2008) Use of identification trial statistics for combination of biometric matchers. *IEEE Trans Inf Forensics Secur* 3(4):719–733
96. Tulyakov S, Wu C, Govindaraju V (2010) On the difference between optimal combination functions for verification and identification systems. *Int J Pattern Recognit Artif Intell* 24(2):173–191
97. Turk M, Pentland A (1991) Eigenfaces for recognition. *J Cogn Neurosci* 3(1):71–86
98. Uchida S, Sakoe H (2005) A survey of elastic matching techniques for handwritten character recognition. *IEICE Trans Inf Syst E88-D(8)*:1781–1790
99. Verma B, Blumenstein M (2008) Fusion of segmentation strategies for off-line cursive handwriting recognition. In: Verma B, Blumenstein M (eds) *Pattern recognition technologies and applications: recent advances*. IGI Global, Hershey
100. Wang K, Yang YY, Suen CY (1988) Multi-layer projections for the classification of similar Chinese characters. In: 9th international conference on pattern recognition, Rome, 1988, vol.2, pp 842–844
101. Xu L, Krzyzak A, Suen CY (1992) Methods of combining multiple classifiers and their applications to handwriting recognition. *IEEE Trans Syst Man Cybern* 22(3):418–435
102. Xue H, Govindaraju V (2002) Incorporating contextual character geometry in word recognition. In: Proceedings of eighth international workshop on frontiers in handwriting recognition, Niagara-on-the-Lake, 2002, pp 123–127
103. Xue H, Govindaraju V (2006) Hidden Markov models combining discrete symbols and continuous attributes in handwriting recognition. *IEEE Trans Pattern Anal Mach Intell* 28(3):458–462
104. Ying L, Srihari SN (1997) Document image binarization based on texture features. *IEEE Trans Pattern Anal Mach Intell* 19(5):540–544

## Further Reading

Researchers typically do not separate the tasks of recognizing handprinted data from more cursive handwritten data. As a result, many published algorithms can be applied to both types of data, and it might be hard to say if a particular algorithm is more suitable to deal with handprinted or cursive handwritten characters and words. Thus, the image enhancement and binarization techniques described here and in ►Chap. 4 (Imaging Techniques in Document Analysis Processes) can be used with both types of text. The extensive overview of feature extraction methods mostly related to handprinted characters is presented in [93]. The character and word recognition methods reviewed in [3, 17, 80] are mostly applicable to handprinted data as well. The methods oriented more towards a cursive handwritten data are described in [12, 81].

---

# Continuous Handwritten Script Recognition **12**

Volkmar Frinken and Horst Bunke

## Contents

Introduction.....	392
Overview.....	393
History and Importance of the Problem.....	393
Evolution of the Problem.....	394
Applications.....	394
Main Difficulties.....	395
Summary of the State of the Art.....	396
Preprocessing.....	397
Overview.....	397
Normalization.....	397
Overview of Common Features.....	398
Pixel-Based Features.....	399
Statistical Features.....	399
Gradient and Filter Features.....	401
Shape-Based Features.....	402
Feature Transformation.....	402
Summary of the Features.....	403
Recognition Methods.....	404
Overview.....	404
Hidden Markov Models.....	405
BLSTM Neural Network with CTC Token Passing Algorithm.....	411
Summary of the Recognizer.....	416
Datasets, Software, and Evaluation Methodology.....	417
Overview.....	417
Databases.....	417

---

V. Frinken (✉)

Computer Vision Center, Autonomous University of Barcelona, Bellaterra, Spain  
e-mail: [vfrinken@cvc.uab.es](mailto:vfrinken@cvc.uab.es)

H. Bunke

Research Group on Computer Vision and Artificial Intelligence (FKI), Institute of Computer  
Science and Applied Mathematics, University of Bern, Bern, Switzerland  
e-mail: [bunke@iam.unibe.ch](mailto:bunke@iam.unibe.ch)

---

Software .....	419
Evaluation Methodology .....	419
Conclusion .....	420
Notes .....	421
References .....	421
Further Reading .....	424

---

### Abstract

The transcription of written text images is one of the most challenging tasks in document analysis since it has to cope with the variability and ambiguity encountered in handwritten data. Only in a very restricted setting, as encountered in postal addresses or bank checks, transcription works well enough for commercial applications. In the case of unconstrained modern handwritten text, recent advances have pushed the field towards becoming interesting for practical applications. For historic data, however, recognition accuracies are still far too low for automatic systems. Instead, recent efforts aim at interactive solutions in which the computer merely assists an expert creating a transcription. In this chapter, an overview of the field is given and the steps along the processing chain from the text line image to the final output are explained, starting with image normalization and feature representation. Two recognition approaches, based on hidden Markov models and neural networks, are introduced in more detail. Finally, databases and software toolkits are presented, and hints to further material are provided.

---

### Keywords

BLSTM neural networks • Feature extraction • Handwriting recognition • Hidden Markov models • Sequential approaches

---

## Introduction

Continuous handwritten script recognition (HWR) is the task of transforming handwritten text into a machine-readable format, for example, a string of ASCII characters. The way the text is entered into the computer can either be *online* or *off-line*. In the case of *online* recognition, information about the pen is sampled at well-defined points in time while the text is written. Therefore, the input is a sequence of values, such as the  $x$ - and  $y$ -coordinates of the pen tip. By contrast, *off-line* recognition is the task of reading text from an image, such as a scanned document. Early approaches tried to treat handwritten text one character or word at a time [6, 52]. However, a main problem became evident with this approach. While machine-printed text can easily be segmented into single words or characters to be recognized individually, this does not hold true for cursive handwritten text. In 1973, Robert Sayre summarized that knowledge in what is known as the Sayre's paradox: *To recognize a letter, one must know where it starts and where it ends, to isolate a letter, one must recognize it first* [50]. Nowadays it has become a widely accepted

position in *off-line* recognition that it is necessary to treat the entire text line as one sequence. This way, one does not need to separate the letter extraction from letter recognition task. In realizing the sequential nature of continuous handwritten script, its recognition has more similarities to speech recognition than machine-printed optical character recognition.

This chapter focuses on the off-line recognition of Latin scripts, but it has been shown that the same, or similar, approaches are also valid for numerous other scripts, such as Arabic or Devanagari [36, 53]. ►Chapter 13 (Middle Eastern Character Recognition) of this book takes a detailed look at recognition techniques for Middle Eastern scripts. The rest of this chapter is structured as follows. In section “[Overview](#),” an overview of the field of continuous recognition is given. Preprocessing steps specific to *off-line* handwriting recognition, such as the normalization of the writing slant, are reviewed in section – “[Preprocessing](#)” while common features used to convert a text line into a sequence of features vectors are discussed in section “[Overview of Common Features](#).“ Recognition techniques are presented in section “[Recognition Methods](#)” with a focus on Hidden Markov models and neural networks. Common databases, software toolkits, and remarks about evaluation methodology are given in section “[Datasets, Software, and Evaluation Methodology](#).“ Finally, the chapter closes with a summary of the key issues and promising approaches in section “[Conclusion](#).“

---

## Overview

### History and Importance of the Problem

Handwriting evolved as means of communication between humans and became a common way of exchanging information. The automatic recognition of handwritten text offers a natural way for human-computer interaction. Instead of requiring a person to adopt to technology, or to require a person to extract valuable information from a given handwritten text and type it manually into a computer, automatic handwriting recognition tries to directly decode the written information.

Depending on the context in which such a system is used, two fields with different prerequisites and requirements can be distinguished. In one field, handwriting recognition serves as a preprocessing step for further interpretation of the content. Later in this book, in ►Part D (Processing of Non-textual Information) and in ►Part E (Applications), several key applications based on handwriting recognition are presented. Examples of the case where handwriting recognition is used as a preprocessing step are address reading on pieces of mail (see ►Chap. 21 (Document Analysis in Postal Applications and Check Processing)), bank check processing, or analyzing handwritten letters in order to classify them. Typically, the vocabulary is limited, and robust post-processing methods might be able to deal with a set of recognition hypotheses instead of just one final recognition result. Sometimes, single-word recognition (see ►Chap. 11 (Handprinted Character and Word Recognition)) approaches are sufficient for that. The other case is the pure transcription of

handwritten text, for example, when digitizing historic documents (see ►Chap. 23 (Analysis of Documents Born Digital)) or personal notes. In this case, a restricted vocabulary cannot be expected, and only one recognition output is sought after.

Well-working commercial applications exist for bank check processing and address reading. As far as transcription is concerned, recognition engines for tablet computers and mobile devices exist that yield good results. For historic data, however, the variability of existing writing styles limits the applications in that domain to interactive systems that aid in generating a transcription.

## **Evolution of the Problem**

Interest in automatic recognition of handwritten text started in the middle of the last century when banks began to recognize the potential of automatic check processing [49]. Soon, automatic postal address processing followed [2]. These two tasks, as well as form reading, have the advantage of strict constraints. The underlying vocabulary is limited, and, due to some redundancy implicitly included in the task, error detection is possible. For example, on a bank check, the legal amount written in words has to match the one written in digits. Similarly, in a postal address, the name of the city has to match the zip code, and the street and house number have to exist.

With the advent of more powerful recognition systems, more and more constraints were lifted and the recognition of unconstrained continuous handwritten script became a focus of attention. State-of-the-art recognition systems are nowadays able to perform, e.g., at an accuracy of 75 % on writer-independent, unconstrained English texts [14]. However, this figure is still far away from what is needed in many practical applications. Therefore, after several decades of intensive research, serious efforts are still undertaken to improve the performance of current recognition technology.

Traditionally, humans were involved in the transcription process only towards the end of the recognition process, in order to correct errors committed by the system. Recently, more elaborate approaches have been proposed where a much tighter interaction between the recognition system and a human corrector is achieved. In this way, the human effort for error correction can be significantly reduced [58].

## **Applications**

The recognition of handwritten text can make human-machine interaction more natural. Automatically recognizing documents written in a way to be legible for humans and not specifically for machines reduces the overhead of digitalizing data. Hence, applications of continuous handwritten script recognition are present, wherever written information has to find a way into a computer system. In the previous paragraphs, bank check reading [22], address reading [7], and form

processing [34] have been mentioned. In addition, digitalization and classification of text in handwritten letters and forms are becoming more and more automatized. Companies such as ITESOFT<sup>®</sup>, VisionObjects<sup>®</sup>, and A2iA<sup>®</sup> offer such solutions to businesses for efficient communications with customers.

In the context of preserving the world's cultural heritage, the digitalization of historic documents emerges as an important task. Huge amounts of handwritten data are stored in libraries, either completely unavailable to the public or only in the form of scanned images [44]. Obviously, the usefulness of collecting those images increases dramatically with the possibility of searching, browsing, and reading the transcribed content.

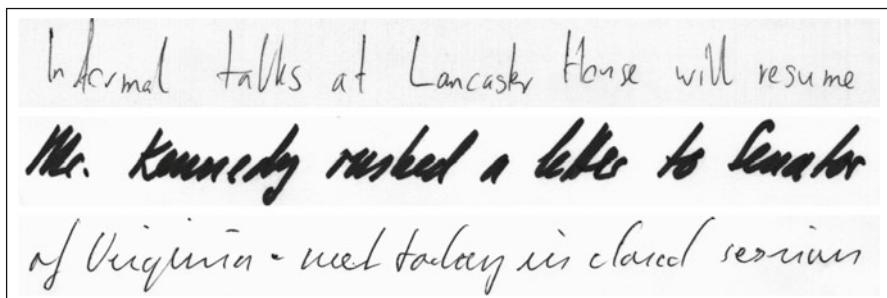
## Main Difficulties

Similarly to other applications dealing with data not specifically designed for automatic processing, handwriting recognition poses several challenges. Extreme variability can be encountered in the writing styles even within a group of writers with a similar cultural and educational background. Samples of English text lines written by Swiss students can be seen in Fig. 12.1a. In addition to that, handwriting is taught differently across different countries and generations. In contrast to speech, which humans use frequently in daily communication, writing text is not done as frequently and often only for personal use. As a consequence, handwriting styles can become sloppy to a degree where even the writer has difficulties reading his or her own handwriting after some time.

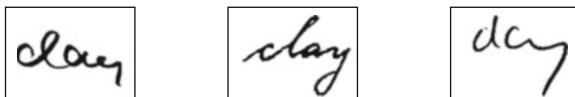
But not only in those extreme cases, reading handwritten text requires understanding the context. Latin cursive script is too ambiguous to be recognized one isolated word at a time. The example in Fig. 12.1b shows that the letters “cl” are often written very similarly to the letter “d.” Hence, the presented words could be “day” or “clay.” Obviously, as the meaning of the two possible transcriptions is so different, a human instantly recognizes the words as “day” if they are preceded by “Have a nice,” for example. However, when the surrounding text is about the history of water jugs, the images are recognized as “clay.”

Furthermore, the segmentation of text lines into individual words is not a straightforward task. It is easier than segmenting a word into characters, since interword gaps are usually larger than intra-word gaps. Nevertheless, many exceptions exist. In Fig. 12.1c, the gap in the first word is larger than the gap between the two words at the end of the text line.

In a general text, every possible sequence of characters may occur, especially when proper names are used or words of different languages. Nevertheless, allowing any possible character sequence as an output is not desirable since it may impede the robustness of the system. As a consequence, one can restrict the vocabulary of words that can be recognized, at the cost of not being able to recognize out-of-vocabulary words at all. The benefit of this approach is an increased average recognition rate, since most words in a single text originate from a limited vocabulary.

**a**

Handwritten text lines

**b**

“day” or “clay”?

**c**

includes the bare minimum text).

Intra-word gaps might be larger than inter-word gaps

**Fig. 12.1** (a) Examples of different writing styles, (b) ambiguous words, and (c) segmentation challenges

## Summary of the State of the Art

The state of the art in continuous handwriting recognition has led to a number of well-working applications for small vocabulary or single-writer tasks. But it was also realized that general, large vocabulary unconstrained handwriting recognition continues to be a hard task for which only research prototypes exist. Typically, high-performing recognition systems try to make use of as much external information as possible, such as specific dictionaries or language models in the form of word probabilities. Also, research has started to focus on particular tasks like interactive transcription systems or cases where training data is limited.

As far as recognition technology is concerned, nearly all modern approaches are segmentation-free. In such a setup, a text line is transformed into a sequence of feature vectors. Afterwards, methods specifically designed to process sequences are used to recognize handwritten text. Many of those methods have their origins in speech recognition.

## Preprocessing

### Overview

The transformation of text line images into sequences of features vectors requires several preprocessing steps. At first, the area of the scanned page containing the handwritten text needs to be localized and cleaned of artifacts like horizontal ruling on the page. Afterwards, individual text lines are extracted and normalized. Up to text line skew detection and normalization, the same methods as the ones reviewed in ▶Chap. 3 (The Evolution of Document Image Analysis) of this book can be applied. Nevertheless, succeeding normalization steps are needed to cope with the immense differences in writing styles. A sample of typical normalization steps are given in the following subsections.

### Normalization

A property specific to handwritten text lines in the angle by which the writing is inclined is the so-called writing slant. An example is given in Fig. 12.2. The slant angle can be detected in several ways (see below). Once it has been estimated, a sheer operation is used to normalize the slant angle without affecting the slope correction. Afterwards, the height of the text is normalized. The major challenge in this step is to treat ascenders, descenders, and the core region of the text differently.

#### Uniform Slant Correction

In [37], the contour of the script is computed by marking all pixels whose horizontal neighbors have a different brightness with the difference exceeding a certain threshold. Note that only horizontal neighbors are considered to put an emphasis on vertical strokes. Next, the pixels are approximated by straight lines. Finally, a weighted histogram of the inclinations of all lines is computed using the squared lines' lengths as weights. The angle where the maximum of the histogram occurs is then considered to be the writing slant.

#### Nonuniform Slant Correction

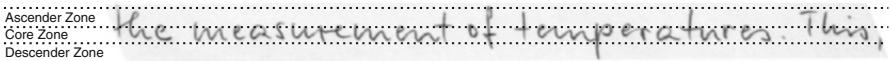
In [52] an edge detection operator is used to find the edges of the writing. Then the average angle of nearly vertical strokes is computed. In [55], a non-uniform approach is proposed that estimates the local slant angles by solving a global optimization problem on the sequence of local slant angles using dynamic programming. In [14] a neural network is used to correct the local slant.

#### Text Height Normalization

The height normalization is done by horizontally segmenting text into three writing zones: the core zone, the ascender zone, and the descender zone. The core zone is



**Fig. 12.2** Illustration of writing slant



**Fig. 12.3** The three different writing zones

the region in which all lowercase letters are written that do not have ascending or descending strokes. It is known as “x-height,” and examples of characters occupying only this zone are *a*, *c*, *m*, or *x* (see Fig. 12.3). The area above the core zone, which is filled by capital letters or ascenders, is the ascender zone. The zone below the core zone in which descending strokes are written is called descender zone.

By analyzing the horizontal histogram of the black pixels, the three different zones can be found since the number of black pixels in the ascender and descender zone is substantially lower than in the core zone. More work on height normalization with histogram analysis is described in [6]. A learning-based method is proposed in [51]. Similarly, in [14], the borders between the different zones are estimated using a neural network. Once the three different zones are detected, they can be normalized to uniform height in a nonlinear way.

### Text Width Normalization

In a next step, the writing width can also be normalized. Ideally, the text line image should be stretched or compressed horizontally so that each character has a predefined width. However, this is not possible since the actual text needs to be known for this. Instead, the frequency of strokes crossing the line that runs horizontally through the middle of the image is counted. This quantity may then be used to adjust the image’s width by setting the average distance between these crossings to a predefined value.

---

### Overview of Common Features

Features for continuous handwritten text can be separated into holistic features, extracted from an entire word for segmentation-based approaches, or sequential features for segmentation-free approaches. While ▶Chap. 11 (Handprinted Character and Word Recognition) of this book covers features for single-word recognition, this chapter focuses on the latter ones. Feature extraction processes are discussed that transform a text line into a sequence of vectors. These processes typically use a sliding window. In such an approach, a small window with a width of  $n$  pixels is swept across the text line image in steps of  $m$  pixels, usually in the direction of writing, i.e., left to right in Latin scripts. At each position, a feature vector is extracted from the window and the feature vectors are then concatenated to a

sequence. Thus, the two-dimensional information of the text image is transformed into a one-dimensional sequence.

In short, the goal of feature extraction is to represent arbitrarily complex shapes by a fixed set of numerical attributes as well as possible. A good feature set should contain discriminative information but should also be easy to compute. Here, we present popular pixel-based features that directly use the grayscale values; statistical features that focus on properties of the black pixel distribution; gradient-and filter-based features; and shape-based features that try to capture significant information of the shape around the sliding window.

## Pixel-Based Features

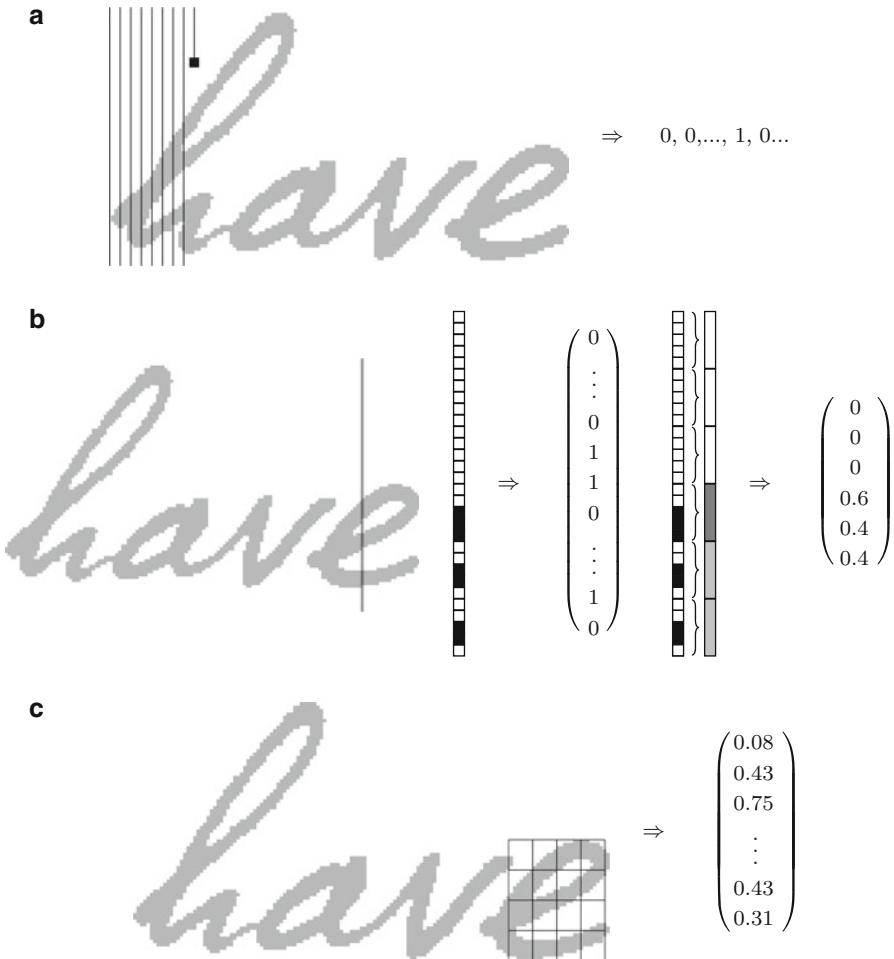
The most basic features than can be extracted from a text line image are the raw pixel intensities of the window, without any further processing. This can basically be done in two ways. The first way is to read the gray-level value of one pixel after another, e.g., top to bottom, one column after another, as depicted in Fig. 12.4a. This results in a one-dimensional sequence of the image that is easy to process. However, the two-dimensional information of the image is lost or must be transmitted separately. In [24], this approach is used in combination with a recurrent neural network, specifically designed for such sequences. At each time step  $t$ , not only the state of the hidden nodes at time  $t - 1$  but also the state at time  $t - H$  can be accessed, where  $H$  is the height of the image. This way, when a pixel is processed, the context to the top and to the left of that pixel is known. The other approach to use the gray-level values of an image is via a sliding window of width 1, i.e., one column of the image, which is directly interpreted as a column vector (see Fig. 12.4b). This approach is followed in [57].

Using directly the pixels, especially as a column vector, does not tend to be extremely robust when it comes to different writing styles. In fact, a vertical displacement of the same word by just one pixel changes the feature vector substantially. To counter this effect, the sliding window can be divided into several partitions, and only the average value of each partition is considered, as depicted in Fig. 12.4b. Those partitions can be the three different writing zones [32] or just arbitrary blocks [4].

To cope even better with variations in text size and vertical positions, a quite successful set of features has been proposed in [60]. The sliding window is subdivided into a regular  $4 \times 4$  grid, and its height is adjusted to only consider the area containing black pixels, as shown in Fig. 12.4c.

## Statistical Features

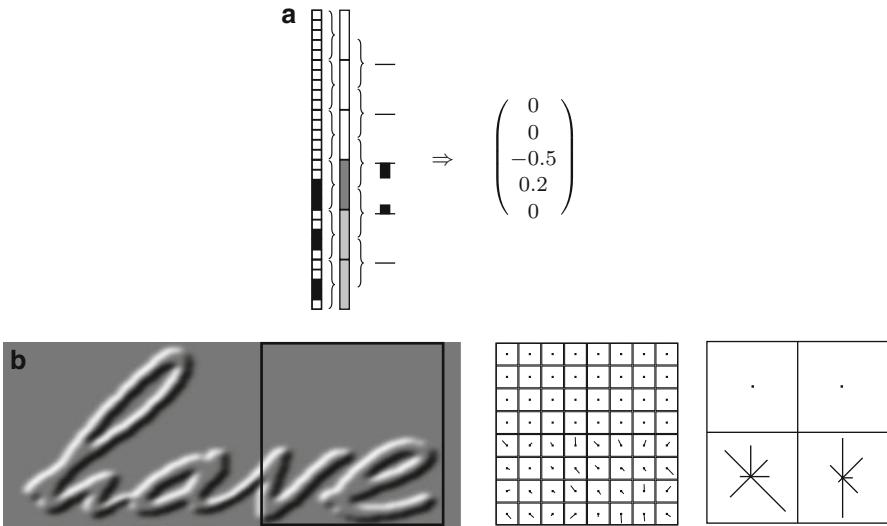
From a mathematical point of view, one can focus one the distribution of the black pixels within the sliding window and use statistical measures of that distribution as feature values. A common approach to do this is to compute the distribution's moments



**Fig. 12.4** Examples of pixel-based features. (a) A vertical scan line creates a one-dimensional feature sequence (b) Gray-level values from a small sliding window (c) Averages from a height-adjusted regular grid

$$m_W^{p_x, p_y} = \sum_{(x,y) \in W} D(x, y) \cdot x^{p_x} y^{p_y},$$

where  $m_W^{p_x, p_y}$  is the  $(p_x, p_y)$ th moment,  $p_x, p_y = 0, 1, \dots$ , of the pixels' distribution in window  $W$ ,  $(x, y) \in W$  are the individual pixels, and the darkness value  $D(x, y)$  returns 1 if the pixel is black and 0 otherwise [32, 38].



**Fig. 12.5** The derivative of the zones act as features. **(a)** Vertical derivatives between zones. **(b)** Histogram of quantized gradients of 4 zones, subdivided into 16 cells each

## Gradient and Filter Features

For the following class of features, assume that the image is not binarized but given as a continuous intensity function  $I(x, y) \in [0, 1]$ . In fact, most approaches that use gradient features blur the image using a Gaussian function prior to the feature extraction process. Using the intensity function  $I(x, y)$ , the horizontal and vertical gradient can be computed as its partial derivatives along the axes. Usually, however, the gradient is not taken between adjacent pixels but rather between larger zones as the difference of the average intensities in these zones. An example is given in Fig. 12.5a.

Consequently, the gradient within a rectangular cell can also be computed. The horizontal gradient is the difference between the average intensity on the left half and the right half of the cell. Similarly, the vertical gradient is given by the difference between the top and the bottom part of that cell. Hence, when dividing the sliding window into rectangular cells, the set of vertical and horizontal gradients from the different cell constitute a set of gradient features [4, 57].

Frequently, not the single gradients themselves are used, but a histogram of gradients. In this approach, the horizontal and vertical gradients ( $G_x, G_y$ ) from each cell are treated as a vector and represented by its length and direction. The direction is then quantized and weighted according to the length of the vector. Finally, the gradient vectors from different parts of the sliding window are accumulated in different histograms of gradients. An example of the process is depicted in Fig. 12.5b. On the left hand side, the magnitude of the vertical gradients of the

sample image is represented by different gray values (1 is white,  $-1$  is black), superimposed with the sliding window. To the right, the gradient vectors for each cell are indicated and finally combined to four different histograms. Notable works following a comparable approach are [47, 56].

Mathematically speaking, the computation of a gradient between adjacent areas can be described as a convolution of the image with a simple, Haar-like filter. Instead of such a rather simple filter, more sophisticated ones are also applicable. Gabor filters, e.g., are sensitive towards a sinusoidal stimulus of a specific direction and frequency. Hence, several Gabor filters with different directions and frequencies can be applied to areas of the sliding window to compute features [9]. There is evidence that the human eye processes information in a similar way [59].

## Shape-Based Features

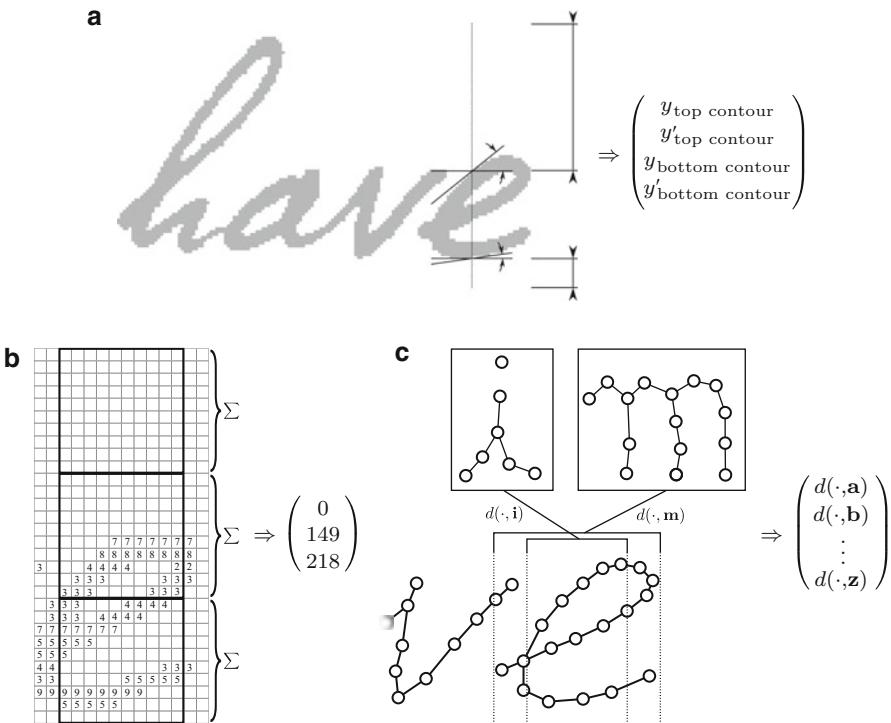
Shape-based features express attributes about the shape of the writing within the sliding window as numerical values. Frequently used, yet simple attributes are the frequency of vertical black-white transitions as well as the position and inclination of the contour [20, 38]. Contour features of this type are displayed in Fig. 12.6a.

A different way to represent a shape is by using run-length information, which is the number of similar, consecutive pixels along a given line. In that regard, the position of the contour can also be seen as a run-length, since it reflects the number of white pixels from the top, resp. the bottom, of the image until the first black pixel is reached. Likewise, the number of black pixels along a line can also be used. Figure 12.6b shows such an approach. For each pixel, its horizontal run-length is given. The sum of the run-lengths in different zones of the sliding window can then be used as features. Obviously, the direction of the run-length does not necessarily need to be horizontal, but can be any direction [19].

Still, the problem exists of how to represent complex, two-dimensional shape information using a vector. Upon a closer look, however, the absolute shape is not as important as its similarity to known shapes, such as characters. In the context of continuous text recognition, it was proposed to match a set of prototype shapes with the content of different sliding windows, each having the same size as the corresponding prototype shape (see Fig. 12.6c). Afterwards, the distances to the prototype shapes is used as feature values. In [16], the prototype shapes and the writing are represented as a skeleton graph. The distance between the sliding window subgraph and the prototype graph is computed via a graph edit distance.

## Feature Transformation

Often recognition systems do not make use of just one class of features but combine several types. Also, feature transformation methods have shown to be beneficial, such as using the Loeve-Karhunen transformation [42], a Kernel-principal



**Fig. 12.6** Some shape-based features. (a) The position  $y$  and inclination  $y'$  of the contour. (b) Horizontal run-length features. (c) Graph edit distance to prototypes

component analysis (PCA) [15], neural network-based non-linear PCA [61], or independent component analysis [15, 61].

## Summary of the Features

In List 12.1, a list of the feature types are given, together with a list of their advantages and disadvantages. Table 12.1 contains an overview of key references. Note that some features have only been proposed for word recognition or keyword spotting. However, it is obvious that they can be used for continuous text line recognition as well.

For a successful recognition system, selected features should not be too sensitive to noise and have discriminative power, i.e., different characters should produce different feature vector sequences. Selecting too few features might not be able to do that while too many features might lead to unstable or overtrained systems. Selecting a well-performing set of features can be difficult and may depend upon the script, the writers, the specific task, and even the amount of the available training data.

*Pixel-based features:*

- + Easy implementation
- + Script and style independent
- + Fast
- Careful height normalization required, sensitive against vertical variations
- Puts more burden on the recognition system

*Gradient-based features:*

- + The best performing features in the literature are gradient-based.
- Not independent of stroke thickness.
- The areas in which the gradients are computed are fixed, hence not extremely robust against vertical variations of the text.

*Shape-based features:*

- + Complex shape information is extracted.
- + Each feature carries meaningful information.
- Potentially slow to compute.
- Shape representation important.
- Not script independent.

*Feature transformations:*

- + Discriminative information can be enhanced.
- + Reduction of irrelevant information.
- A further step that adds complexity and a possible bias.
- Some feature transformations require learning, which requires more training data.

**List 12.1** A list of advantages and disadvantages for different types of features for continuous handwriting recognition

**Table 12.1** An overview of features for converting an image of handwritten text into a sequence of vectors

Feature description	References
Pixel value of a column as binary vector	[57]
1D sequence of pixels, top to bottom, left to right	[24]
Number of black/white transitions	[20, 38]
Partial derivatives along $x$ and $y$ axis	[4, 14, 57]
Histogram of derivatives in subregions	[56]
Histogram of gradients in subregions	[47]
Position and slope of the contour	[38]
Graph edit distances to prototype letters	[16]
Correlation of intensities between subregions	[4]
Post-processing of medium and low-level features	[15, 42, 59, 61]

---

## Recognition Methods

### Overview

In this section, methods are discussed to transform a sequence of feature vectors, extracted from a handwritten text line, into a character or word sequence.

An obvious approach is to segment the text line into individual words and individually classify the segments using one of the methods described in ►Chap. 11 (Handprinted Character and Word Recognition). However, similarly to classifying a word into individual letters, ambiguities might arise when segmenting a text line into words. Hence, the classification results obtained on the single segments have to undergo suitable post-processing procedures, using, e.g., dynamic programming. Further details on segmentation can be found in ►Chap. 11 (Handprinted Character and Word Recognition) and [31].

A handwritten text line can also be considered as a horizontal signal. From that point of view, similarities to speech recognition are apparent. Consequently, most, if not all, sequential approaches to continuous handwriting recognition have their roots in speech recognition. The most promising ones include statistical methods such as hidden Markov models. Due to the outstanding role these models play in this field, a detailed review is given in section “[Hidden Markov Models](#).” A well-performing neural network architecture that was recently developed and since then has been consistently outperforming traditional approaches is the so-called bidirectional long short-term memory neural network. Its emergence significantly changed the field and further improvements are to be expected. This network is presented in section “[BLSTM Neural Network with CTC Token Passing Algorithm](#).”

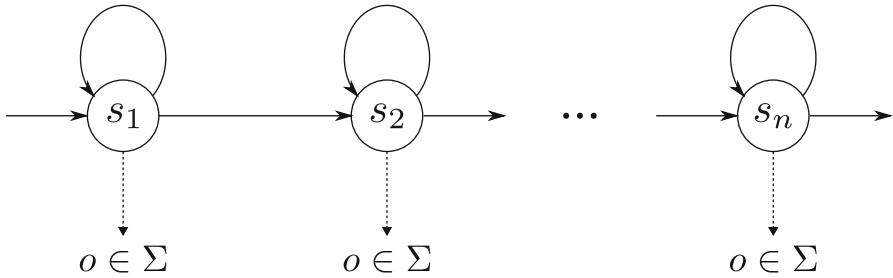
Other sequential recognition techniques [11] might theoretically also be applied to this task. Although Conditional Random Fields and their derivatives have occasionally been proposed for single-word recognition [12], they play practically no major role in continuous handwriting recognition.

## Hidden Markov Models

### General Definition

A hidden Markov model (HMM) is a model of a stochastic process which generates a sequence of output elements or emissions over a period of time. The system may have several internal and unobservable states which are assumed to resemble a first-order Markov chain, i.e., a *discrete-time* random process, which is *stationary*, *causal*, and *simple*. Discrete time means that at each time step, the system is in exactly one of several distinct states. Stationary means that the absolute time does not have an influence on the state the system is in. Causal means that only states the system has been in the past, but no future events, influence the next state. Finally, simple means that only the current state influences the next state. In short, the probability  $p(S(t) = s_i)$  of the system to be in state  $S = s_i$  at time  $t$  does only depend upon the state at time  $t - 1$  and can therefore be written in the form of a state transition probability matrix  $A = (a_{ij}) \in [0; 1]^{n \times n}$  with

$$a_{ij} = P(S(t) = s_j | S(t - 1) = s_i).$$



**Fig. 12.7** Linear HMM state transition topology for handwriting recognition

Formally, an HMM  $\lambda$  is a tuple  $\lambda = (S, A, \pi, B)$ , where  $S = \{s_1, s_2, \dots, s_n\}$  is a set of states,  $A = (a_{ij})$  the state transition probability matrix,  $\pi \in [0; 1]^n$  the start probability vector, and  $B = \{b_1, b_2, \dots, b_n\}$  a set of state-dependent emission probabilities. The system starts in one of the states, according to the start probabilities  $\pi$ . At each time step it changes its internal state, according to the state transition probability matrix  $A$  and emits an element according to the (state-dependent) emission probability function.

Although no restrictions are given for possible state transitions according to this definition, certain topologies are frequently used. The ergodic model is the most general case, allowing a transition from every state to every other. If  $A$  is an upper triangular matrix, the states form an ordering, since from one state, only transitions into following states are possible; hence, it is called left-right model. Common for handwriting recognition are even more restricted models, especially the Bakis model in which only the next two states can be changed into and the linear model that allows transitions into the next state only. A linear hidden Markov model is shown in Fig. 12.7.

### Training and Decoding Algorithms

The popularity of HMMs and their applicability to a diverse class of problems can be accounted to the fact that efficient training algorithms exist to automatically estimate the parameters of an HMM and to infer the most likely sequence of hidden states given an observed sequence.

The most common training method is the *Baum-Welch* algorithm, introduced in [3]. It considers the probability  $P(O|\lambda)$  that a given sequence  $O$  is produced by the hidden Markov model  $\lambda$  and computes a new model  $\lambda'$  which is more (or equally) likely to have produced the given sequence, so that  $P(O|\lambda') \geq P(O|\lambda)$ . It is an iterative optimization algorithm which converges towards a local maximum, but it cannot guarantee to find the globally optimal parameters. A detailed description of the Baum-Welch algorithm is given in [45].

If the emission probabilities are modeled by Gaussian distributions, every sequence of hidden states could have produced the results. Therefore, the most likely sequence

$$s_{\text{best}} = \arg \max_{s \in S^*} P(s|O, \lambda)$$

out of a set of sequences  $S^*$  is sought after.  $S^*$  does not contain every possible character sequence but only those that form words contained in the language model. Using the Bayes rule [13] and dropping the constant factor  $P(O|\lambda)$ , which does not influence the result, leads to

$$s_{\text{best}} = \arg \max_{s \in S^*} P(s|O, \lambda) = \arg \max_{s \in S^*} \frac{P(O, s|\lambda)}{P(O|\lambda)} = \arg \max_{s \in S^*} P(O, s|\lambda). \quad (12.1)$$

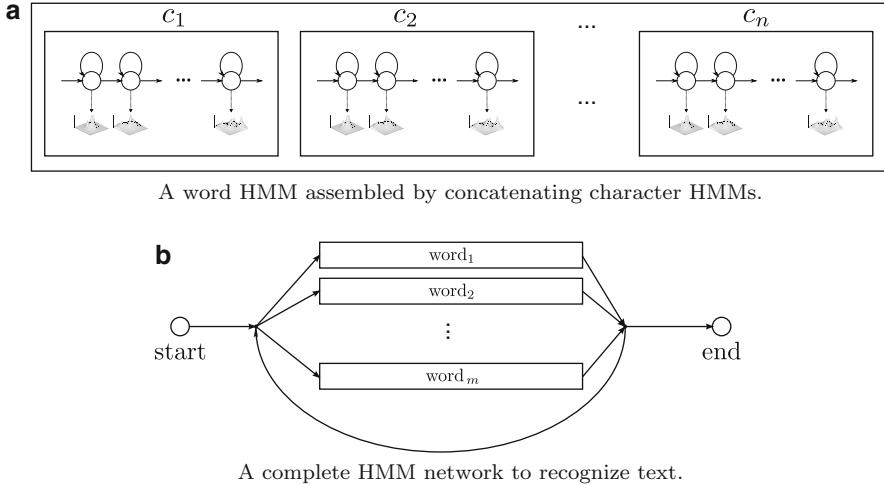
In the *Viterbi* algorithm [63], this equation is used to compute the most likely sequence. In the first step, the starting probabilities are computed. In each of the following iteration over all observed sequence elements, a state's probability can be inferred by exploiting the fact that next to the observed emission, only the system's predecessor state is important. The algorithm stores for all steps and each state the most likely preceding state and after the sequence elements have been processed, starting from the end, the most likely states can be efficiently inferred in a dynamic programming approach.

### **Hidden Markov Models for Continuous Handwriting Recognition**

With their ability to compute a likely sequence of hidden states given a sequence of elements, hidden Markov models are well suited for handwriting recognition. For continuous handwriting recognition, a text line is represented as a sequence of elements. It is assumed that the writing process that created the letters and finally the sequence can be modeled by a hidden Markov model. In this setup, the emissions of the system are the feature vectors of the sequence. The internal states correspond to words, characters, or even smaller units. Character HMMs have advantages and disadvantages over word HMMs. On the one hand, small, common words are often written in one stroke and do not resemble the simple concatenation of the individual letters very well. On the other hand, in each text, the number of training elements is much higher when only characters are to be trained as opposed to whole words. In most cases, the advantages gained by the increase in training data when using character models outweigh the disadvantage of having to model, in one HMM, very diverse ways a character can be written.

In the training phase, text line HMMs are assembled using character HMMs as well as HMMs representing punctuation marks and spaces. To decode more than just a single word, a modified version of the Viterbi algorithm, called *Token Passing Model*, is applied [66].

Usually a separate hidden Markov model is constructed for each text unit. Hence, character HMMs or whole-word HMMs are used frequently. Afterwards, these are in turn aggregated into a larger network. In each HMM, the states follow a clear ordering, mostly in a linear or a Bakis topology. The number of states in such HMMs depends upon the size and complexity of the object and has to be validated or set according to prior knowledge. In Fig. 12.8 an architecture of a character-based system can be seen. First, character HMMs are connected to form word HMMs



**Fig. 12.8** (a) A character HMM consisting of several states can be seen. These models are connected to form words. (b) All words are connected in a network for the recognition

according to a lexicon (Fig. 12.8a). Under such an approach, the recognition is restricted to those words that occur in the lexicon. Next, given the word models, the end of each word is connected to the beginning of each word (Fig. 12.8b). As a result, a model for word sequences is obtained.

At each time step, an output is generated according to the current state's output probability distribution. This implies that it is necessary to compute for a given element  $x$  of the sequence the emission probability  $p(x|q)$  of  $x$  being generated in state  $q$  in order to derive the most likely sequence of internal states. Different approaches exist to model the emission probabilities. They will be discussed next.

### Discrete HMM

If  $x$  is from a discrete set of possible elements  $\{x_1, x_2, \dots, x_n\}$ , a simple list can be used to store the individual probabilities. Such a model is called *discrete HMM* and can be used for handwriting recognition by applying a vector space quantization technique on the input elements. The simple representation, fast computation, and the need to train only a few parameters are the most prominent advantages of this approach.

### Continuous HMM

When using real-valued, continuous vectors, arbitrary probability density functions

$$b_i : \mathbb{R}^n \rightarrow \mathbb{R} \text{ with } \int_{\mathbb{R}^n} b_i(x) dx = 1$$

need to be modeled. This is commonly done in one of three ways, using *continuous HMM*, *semicontinuous HMM*, or *hybrid* models.

In the continuous case,  $b_i$  is approximated by a weighted sum of Gaussian normal distributions

$$b_i(x) = \sum_{k=1}^{g_i} c_{k,i} \cdot \mathcal{N}(x; \mu_{k,i}, K_{k,i})$$

with  $\sum_{k=1}^{g_i} c_{k,i} = 1$  for all  $i$  and

$$\mathcal{N}(x; \mu, K) = \frac{1}{\sqrt{2\pi |K|}} \cdot \exp\left(-\frac{1}{2}(x - \mu)' K^{-1}(x - \mu)\right),$$

where  $i = 1, \dots, n$  indicates the state and  $k = 1, \dots, g_i$  the specific Gaussian. Since any probability density function can still be approximated if the covariance matrices are diagonal matrices, a complete description of continuous HMMs is therefore given by a state transition probability matrix  $A$ , a vector indicating the starting probabilities  $\pi$  and a set of weights, means, and the diagonal elements for the covariance matrices for each state  $(c_{k,i}, \mu_{k,i}, \sigma_{k,i,1}, \dots, \sigma_{k,i,n})$ . The number of Gaussian distributions used in the sum to approximate the density function can be chosen by the user.

### Semicontinuous HMM

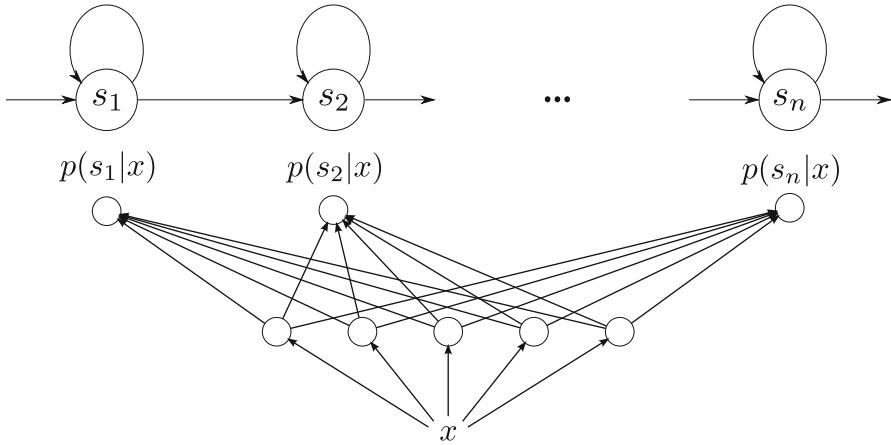
Semicontinuous HMMs make use of the same principle. However, instead of approximating each state's emission probability density function with an own set of Gaussians, one common pool  $\{\mathcal{N}_1, \mathcal{N}_2, \dots, \mathcal{N}_l\}$  of  $l$  Gaussian distribution is used. This way, only the weights  $c_{k,i}$  are specific to a state. Obviously, this approach offers other advantages in addition to a reduced set of parameters that need to be trained. The Gaussians are state independent and can therefore be learned in an unsupervised way, possibly making use of unlabeled data [48].

### Hybrid Approaches

A completely different approach is taken by the so-called hybrid methods. In an ANN/HMM approach [14], a neural network is trained in such a way that it returns for a given element  $x$  the probability for being emitted in each state (See Fig. 12.9). Hence, instead of modeling  $p(x|q_i)$  using a weighted sum of Gaussians,  $p(q_i|x)$  is modeled. Using Bayes' rule, the target function can be rewritten

$$p(x|q_i) = \frac{p(q_i|x)p(x)}{p(q_i)}.$$

The probability  $p(q_i)$  of the HMM being in a specific state can be directly computed from the Model's state transition probabilities. The a priori probability  $p(x)$  is a constant for all states and does therefore not contribute to finding the most likely state transitions. Consequently, it is set to 1. Hence, the likelihood ratio  $\frac{p(q_i|x)}{p(q_i)}$  is used instead of the emission probability.



**Fig. 12.9** In an ANN/HMM hybrid approach, a neural network models the emission probability of each state

By using a sophisticated learning-based approach, any function can theoretically be modeled, just like a weighted sum of Gaussian. However, unlike the classical Baum-Welch approach, training of the neural network is discriminant. Adjusting the weights according to a training sample affects the probability estimates for all states. The authors in [14] also report a remarkable speedup compared to evaluating Gaussian distributions.

The training of hybrid models pose difficulties as the target output for each state must be known for the back-propagation algorithm. One approach is to label every element of every sequences in the training set. Given a transcription, a hidden Markov model can return the most likely segmentation of the sequence into the individual states in a preprocessing phase. Then, during training, back-propagation and Baum-Welch iterations alternate [8]. Also, more sophisticated approaches have been proposed that optimize the neural network and hidden Markov model simultaneously [8].

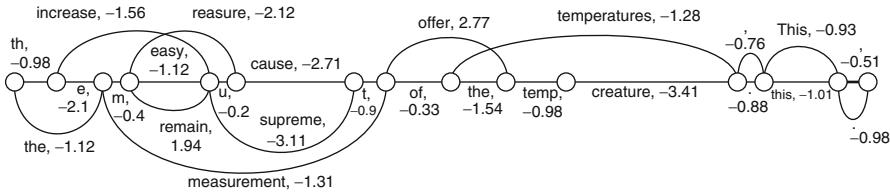
### Language Model Integration

The accuracy of a recognition system can be increased by enhancing the estimation of the prior probabilities  $P(s|\lambda)$  of state sequences  $s$  in the term

$$s_{\text{best}} = \arg \max_{s \in S^*} P(O, s | \lambda) = \arg \max_{s \in S^*} P(O|s, \lambda) P(s|\lambda).$$

With external language information, word transition probabilities  $p(w_1 \rightarrow w_2)$  for each pair of words, so-called *bigram* probabilities, are computed. However, since neither the result of the decoding algorithm nor the bigram probabilities are true probabilities [40], the language information is included via two parameters that also control the influence of the language model on the recognition. These two

the measurement of temperatures. This,



**Fig. 12.10** A sample recognition lattice

parameters are the *grammar scale factor* (*gsf*) and the *word insertion penalty* (*wip*). The *wip* is a fixed value added to the likelihood of every new word. The *gsf* is a factor by which the bigram probability is scaled before it is also added.

The resulting likelihood of a word sequence  $W$ , given an observation  $O$ , is therefore

$$\log \hat{P}(O, W | \lambda) = \log P(O | W, \lambda) + gsf \cdot \log P(W) - l \cdot wip$$

where  $l$  is the number of words in the sequence.

Using a Token Passing algorithm, it is not only possible to produce a  $n$ -best list for single-word recognition but also to produce a recognition lattice. A recognition lattice is a graph, containing the positions of possible word boundaries as node labels and possible words as well as their recognition likelihood as edge labels, as demonstrated in Fig. 12.10. A recognition lattice, produced by the Token Passing algorithm, resembles the subspace of the most likely recognition results.

The so-called bigram probability  $p(w_2|w_1)$  of word  $w_2$  to follow  $w_1$  can be estimated using an external text corpus. This corpus does not need to be handwritten text, but instead should be a representative cross section of the underlying language. For a given word  $w_1$ , one can simply count how often  $w_2$  succeeds it. Nevertheless, using this technique, frequent word pairs are likely to be overestimated, while not all possible word combinations might appear in the corpus, which would result in a probability of zero. Hence, post-processing methods are applied that take probability mass from frequent word pairs and distribute it over the rest. This technique is also known as *smoothing*. A detailed description of language modeling can be found in [21].

## BLSTM Neural Network with CTC Token Passing Algorithm

The second type of recognizer presented in this chapter is based on neural networks. Their general applicability and good performance have made neural networks popular for diverse classification tasks in document analysis. Especially for isolated character recognition, but also for online and off-line recognition of single words,

neural networks have been intensively utilized. For more details on this, see ►Chaps. 11 (Handprinted Character and Word Recognition) and ►26 (Online Handwriting Recognition) of this book.

To decode feature sequences representing text of arbitrary length, the application of neural networks is not completely straightforward since the output is not a single element but a sequence. Actually, one needs to proceed in two steps.

In the first step, a character probability vector for each position in the sequence is estimated. It indicates the probability for each character to be written at that position in the text line. The estimation is done using the actual neural network. In the second step, the probability vector sequence is transformed into a possibly shorter sequence of likely characters or words. This is achieved by using an algorithm called *Connectionist Temporal Classification* (CTC) Token Passing algorithm [25].

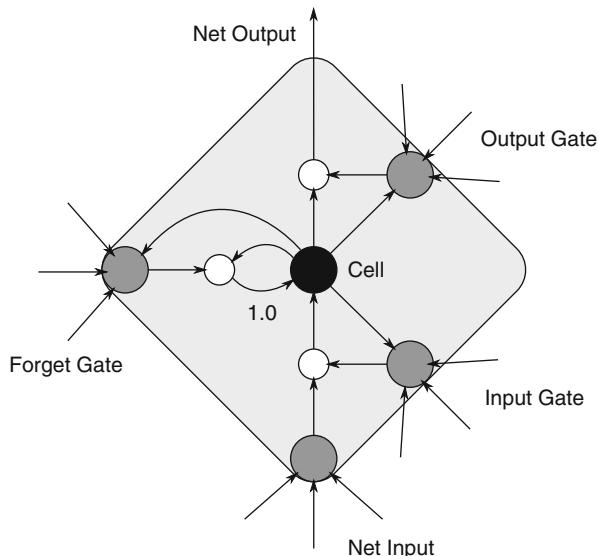
## Neural Networks

The idea behind handwriting recognition using artificial neural networks is to read the sequence of feature vectors, extracted from the image of a text line, into the network. At each position of the input sequence, the network produces an output that is to be interpreted in a subsequent step. From this point of view, neural networks can be regarded as a nonlinear feature transformation function. In the ideal case, the output is a sequence of symbols that can easily be converted into a sequence of words. In a more complex scenario, sophisticated methods, possibly even hidden Markov models, have to be applied.

Using a feed-forward network, only a fixed-sized window of the sequence can be observed at each position. Contextual information, which is important to recognize handwritten text, can only be considered to a limited degree. A recurrent neural network has the advantage that it can be used like a memory to store information over several time steps. However, the value either exponentially decays or increases as it circles around the network, depending upon the weights of the recurrent connections. This drawback is called *vanishing gradient problem* and poses a huge problem for current learning techniques. Hence, such recurrent neural networks are only capable of successfully storing and using information for a few time steps [5].

A recently proposed solution to this problem is *long short-term memory* (LSTM) cells. These cells are specifically designed to address the vanishing gradient problem [28]. An illustration of an LSTM cell can be seen in Fig. 12.11. A core node, with a recurrent weight of 1.0, is in the middle of the cell and serves as a memory. Three nodes are used to control the information flow from the network into the cell and from the cell back into the network. The net input node receives values from the network and forwards its activation into the cell. However, the value is only passed into the core if a second node, the input gate, is activated, or open. Similarly, an output gate regulates if the core's value is fed into the network. A fourth node, the forget node, resets the core's value upon activation. Using these memory cells, information can be stored without loss for arbitrarily many time steps. Interestingly, a similar node architecture has been found in the cortex of the human brain [41].

**Fig. 12.11** Gates control the information flow into and out of each LSTM cell [25]



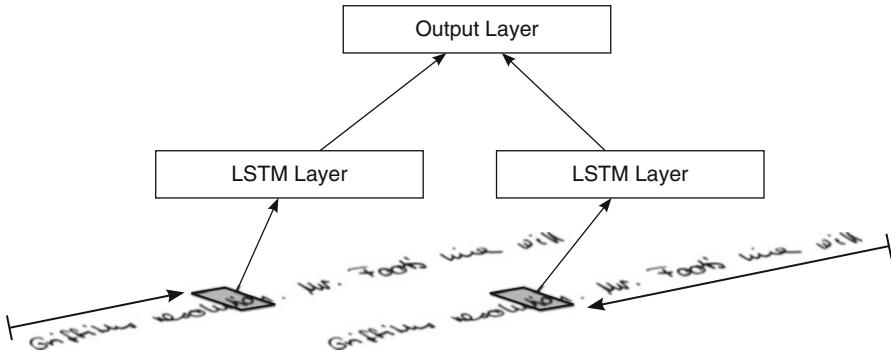
### BLSTM Neural Networks

In sequential data, information is often spread across several time steps. Recurrent neural networks are able to use information from previously processed sequence elements. To use the information not only from past but also from future sequence positions, *bidirectional* LSTM (BLSTM) neural networks can be used. Two distinct parts of the network, each endowed with input and LSTM layers, process the sequence separately; one from left to right and the other from right to left. For off-line information like scanned handwritten text lines, it is not unusual to access context to the left and to the right of each position to recognize a character. For online data, such as speech, an input sequence has to be recorded and segmented into larger segments. This does not necessarily pose a problem, since obvious segmentation points, such as silence between two sentences, can be used. The signal is saved to memory, segmented, and each segment is then processed separately. A BLSTM NN with one hidden LSTM layer processing a line of handwritten text is depicted in Fig. 12.12.

The output layer contains not only one node for each of the possible classes but also one extra node, the  $\varepsilon$  node, to indicate that no decision about the output class can be made at that position.

The activation function in the output layer realizes the softmax normalization

$$y_n(t) = \frac{e^{a_n(t)}}{\sum_k e^{a_k(t)}} ,$$



**Fig. 12.12** An illustration of the mode of operation of the BLSTM neural network. For each position, the output layer sums up the values of the two hidden LSTM layers

where  $a_n(t)$  is the input into node  $n$  at time  $t$ . Due to the normalization, all output activations sum up to one and  $y_n(t)$  can be treated as a posterior class probability  $p(\text{class}(O(t)) = n)$  that the class of the sequence  $O$  at time  $t$  is  $n$ .

In Fig. 12.13 such a probability sequence, where each node corresponds to a character, can be seen. The activation is close to 0 most of the time for normal letters and peaks only at distinct positions. In contrast, the activation level of the  $\varepsilon$  node is nearly constantly 1.

The underlying rationale is now to find a path through the matrix of output activations that represents a meaningful character sequence while simultaneously maximizing its probability score. The probability  $P(s|O)$  of a path  $s$  through the output sequence  $p(c_{k_1}c_{k_2}\dots c_{k_n}|O)$  is given by multiplying the individual probabilities

$$P(s|O) = \prod_{t=1}^T y_{k_t}(t).$$

To recognize class sequences that might be shorter than the input sequence, a given path can be shortened using operator  $\mathcal{B}$ . The operator first deletes consecutive occurrences of the same class and then all  $\varepsilon$  entries:

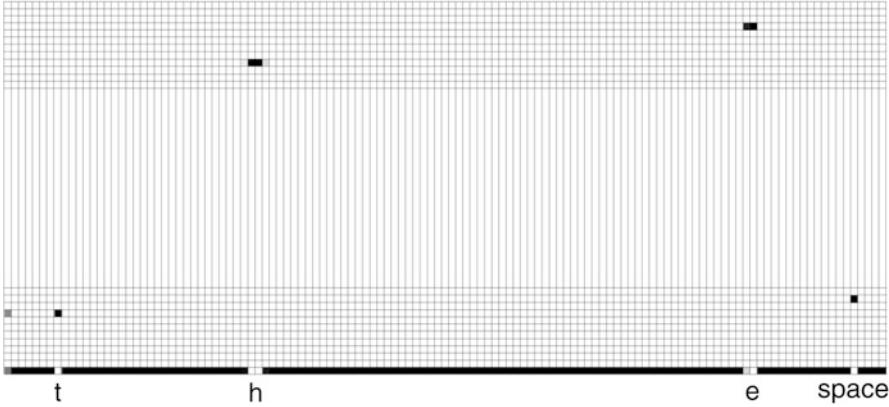
$$\mathcal{B}(c_1, c_2, c_2, c_2, \varepsilon, c_2) = c_1 c_2 c_2$$

$$\mathcal{B}(\varepsilon, c_1, c_1, c_1, c_2, \varepsilon, \varepsilon, c_2) = c_1 c_2 c_2$$

$$\mathcal{B}(\varepsilon, c_1, c_1, c_1, \varepsilon, c_2, c_2) = c_1 c_2$$

### Recognition via the CTC Token Passing Algorithm

The most simple form text recognition using BLSTM NN is to concatenate the most likely character for each time step and apply the shortening operator  $\mathcal{B}$ . In this case, no dictionary or language model is used and a high-recognition accuracy cannot be expected necessarily. Using dynamic programming, however, a dictionary



**Fig. 12.13** An actual output produced from an image of the word “the”

of possible words as well as a language model can be included to improve the recognition accuracy. When only a single word is to be recognized, the optimal path that result in a word from the dictionary using operator  $\mathcal{B}$  can be found efficiently. For continuous text recognition, a more complex version, the *Connectionist Temporal Classification (CTC) Token Passing*, is used that takes bigram probabilities into account to connect different words. A finite-state machine is constructed out of each word representing the inverse function of the shortening operator  $\mathcal{B}^{-1}$  by allowing transitions to the  $\varepsilon$ -node between consecutive characters and direct connections between characters only if they are different. Then, similar to HMM-based recognition where each word is represented as a Markov model, tokens are passed along, taking into account word transition probabilities given by the language model and character observation probabilities given by the network’s output nodes.

## Training

To train a neural network, the weights of all internal connections are initialized randomly and then gradually changed to minimize a global objective function  $\mathcal{O}$  via gradient descent and *back-propagation through time* (BPTT) [64]. For that, the gradient of the objective function w.r.t. the network input for output node  $n$  for each time step  $t$  is needed.

Given a training set of samples  $(O_i, W_i)$  consisting of pairs of observation sequences  $O_i$  and a word sequence  $W_i$ , the goal is to maximize the probability of correctly labeling the entire training set  $\prod_i P(W_i | O_i)$ , or equivalently, minimizing the log probabilities

$$\begin{aligned}\mathcal{O} &= - \sum_i \ln P(W_i | O_i) \\ &= - \ln \sum_{s \in \mathcal{B}^{-1}(W_i)} P(s | O_i) ,\end{aligned}$$

since  $P(W|O)$  is the sum of all paths  $s$  through the matrix of output activations that represent the word sequence. Keeping track of all possible paths is not traceable, but we can split the sum into parts using the Forward-Backward algorithm. Given a point in time  $t$  and a character  $c$ , the probability of an admissible path to be at time  $t$  in character  $c$  can be expressed as

$$P(W|O, s_t = c) = \sum_{s \in \mathcal{B}^{-1}(W) \wedge s_t = c} P(s|O) .$$

Now, with any arbitrary  $t$ , the probability for a single training example  $P(W|O)$  can be computed by summing up all characters in the alphabet  $A$ :

$$P(W|O) = \sum_{c \in A} P(W|O, s_t = c) .$$

As shown in [23], the sought-after gradient can be expressed in terms of these partial sums:

$$\frac{\partial \mathcal{O}}{\partial a_j(t)} = y_j(t) - \frac{P(W|O, s_t = c)}{P(W|O)} .$$

This gradient can then be propagated back into the network to adjust the individual weights. In case of several training elements, the weights can be adjusted after each training element or after computing the average gradients over the entire training set. A separate validation set is used to check the performance of the neural network after each back-propagation iteration to prevent overfitting.

## Summary of the Recognizer

Table 12.2 points out the key references of the presented systems, and a comparison showing the main advantages and disadvantages of the presented recognizers is given in List 12.2. Hidden Markov models are well understood and have a long tradition in being used as recognizers for sequential data. As a consequence, a variety of specific HMM type have been developed for different scenarios. Their capacity to model complex sequences but also the need for training data increases from discrete HMM, via semicontinuous HMM, to continuous HMM. In contrast, the performance of ANN/HMM systems depends on the neural network that models the state probabilities. The hybrid ANN/HMM systems requires a frame-level ground truth, that is, each single-feature vector has to be assigned to a character. However, an approximation is enough for a successful training so that a continuous HMM can be used in a preprocessing step to label each frame.

**Table 12.2** An overview of the most frequently used and established recognition methods

	System	Training	Recognition	Key references
HMM	Discrete	Baum-Welch	Viterbi	[1, 33]
	Continuous	Baum-Welch	Viterbi	[38, 62]
	Semicontinuous	Baum-Welch	Viterbi	[65]
	Hybrid (ANN)	Baum-Welch and back-propagation	Viterbi	[8, 14]
BLSTM NN		Back-propagation	CTC Token Passing	[25]

The BLSTM neural network and the hybrid ANN/HMM both belong to the best performing approaches. Furthermore, both systems have only been introduced recently and therefore still have a large potential of improvement.

---

## Datasets, Software, and Evaluation Methodology

### Overview

In this section, some of the key databases to train and evaluate off-line handwriting recognition methods are presented. Moreover, an overview of software for the HMM and BLSTM NN recognition technologies is presented in section “[Recognition Methods](#),” as well as software tools to create statistical language models. Finally, details for evaluating the recognition rate of a system for continuous handwriting recognition are explained.

### Databases

Due to the variety of possible applications, common databases cover different areas of the field. They range from databases containing historic documents written by a single writer to a collection of modern, French letters. The following list is an overview of commonly used sources of annotated, continuous handwritten data. A regularly updated repository of databases can be found at the website of the IAPR Technical Committee 11 – Reading Systems.<sup>1</sup> A summary of the presented databases is given in Table 12.3.

### Modern Databases

One of the largest and most frequently used databases for modern continuous handwritten data is the *IAM Handwriting Database*<sup>2</sup> [39], consisting of English texts from the Lancaster-Oslo/Bergen Corpus of British English [30]. It therefore forms a representative cross section of the English language. The *RIMES*<sup>3</sup> database is a collection of artificially created letters to individuals and companies in French,

*Hidden Markov models:*

- + Very well understood.
- + As a generative model, new characters can be added easily to an existing system.
- Markov property restricts the inclusion of long-term dependencies.
- The returned likelihood values do not reflect a recognition confidence.

Discrete-HMM with feature vector quantization:

- + Easy and fast approach.
- + Only few parameters to train, works with limited training data.
- Complex probability distribution functions cannot be modeled accurately.

Semicontinuous HMM:

- + No vector quantization needed.
- + The estimation of the individual Gaussian distributions can be done with unlabeled data.
- + A state is given by a small set of coefficients.
- + Fast to train, robust for limited training data.
- Some states' probability distribution functions might only be modeled insufficiently.

Continuous HMM:

- + Can make use of abundant training data.
- + The complexity of each state can be modeled independently.
- Many parameters to train.

Hybrid HMM/ANN:

- + Complex probability distributions functions are learned in a discriminative way.
- + Evaluating the neural network instead of Gaussian distributions decreases the runtime.
- + Among the best performing systems.
- Training is slow and requires a frame-accurately labeled ground truth.

*BLSTM neural network:*

- + The network returns true character occurrence posterior probabilities.
- + The random initialization allows a straightforward generation of recognizer ensembles.
- + Automatic learning of long-term dependencies.
- + Among the best performing approaches to date.
- Random initialization of the neural networks requires that several networks need to be trained and tested to guarantee a good performance.
- Additional characters cannot be added to an existing system.
- An activation peak of an output node does not occur at a predictable position (beginning, middle, end) of the character. Character and word segmentation using BLSTM NNs is difficult.
- Works only well with large amounts of training data.
- Not as well understood and frequently used as HMMs. Nearly no existing software packages.

**List 12.2** A list of advantages and disadvantages for different types of recognizers for continuous handwriting recognition

which makes it therefore useful for business-related applications. Online databases also form an interesting source of data. The given pen trajectories can easily be mapped onto an image to create off-line handwritten texts. Among online databases, *UNIPEN*<sup>4</sup> [26] is the most widely used one. Frequently held “calls for data” ensure a constant growth of the database.

## Historical Databases

Due to the nature of historical documents, the recognition task on these datasets is usually much harder than on modern datasets. The spelling is not standardized, the paper might be degraded with barely legible ink, and the pages can be highly

**Table 12.3** Overview of databases containing continuous handwritten text

Name	Language	Time	Size		Comments
			KWords	Writer	
IAM DB	English	Modern	115	657	
RIMES	French	Modern	250	1,300	Letters
UNIPEN	Several	Modern	74	320	Online data
GW	English	Eighteenth century	Various sets are used		
GERMANA	Several	Nineteenth century	217	1	
RODRIGO	Spanish	Sixteenth century	231	1	

ornamented. On the other hand, the writing style itself is usually more regular within one document. A specific set of *Letters, Orders and Instructions* of George Washington, the GW<sup>5</sup> dataset, evolved to a de facto standard for word retrieval tasks [46].

The GERMANA and RODRIGO datasets<sup>6</sup> represent historic Spanish books from the nineteenth and the sixteenth century, respectively. The GERMANA dataset also contains passages written in other European languages.

## Software

Several software tools for hidden Markov models are available on the Internet, such as the widespread *Hidden Markov Model Toolkit (HTK)*<sup>7</sup> [66] and the *Hidden Markov Model Toolbox for Matlab*.<sup>8</sup> HTK was created to use HMMs for speech recognition and offers sophisticated tools for manipulating existing models. Furthermore, general toolkits for machine learning exist, e.g., TORCH<sup>9</sup> [10], that include options for using HMMs, while for other toolkits, such as WEKA<sup>10</sup> [27], third-party extensions exist.

Comparable toolkits for BLSTM neural networks are not as well developed. To the knowledge of the author, RNNLIB<sup>11</sup> [25] is the only available source of software for this new type of neural network architecture.

To create and modify language models, plenty of programs exist. While, e.g., the MIT<sup>12</sup>, the CMU<sup>13</sup>, and Microsoft®<sup>14</sup> all have their own toolkits, the most frequently used one is SRILM<sup>15</sup> [54].

## Evaluation Methodology

The recognition accuracy of a system that classifies single objects, as in the single-word recognition case, is given by the fraction of correctly recognized elements in relation to all classified elements. Let  $n$  indicate the number of tested elements, out of which  $c$  have been correctly recognized. Then, the accuracy is given by

$$\text{Accuracy} = \frac{c}{n} .$$

For sequential data, when the number of classes is not known, several error cases can occur. First, an element might be correctly recognized as an element, but as an element of the wrong class. This is called a *substitution*. Secondly, an element might falsely be recognized or inserted. Consequently, this is called an *insertion*. Thirdly, an element might not be recognized at all. This is called a *deletion*. Finally, the accuracy is defined by

$$\text{Acc} = \frac{n - S - D - I}{n}$$

where  $S$  is the number of substitutions,  $D$  is the number of deletions, and  $I$  is the number of insertions [62]. In case of text line recognition, the elements are words. For the unconstrained character sequence recognition task, the elements are characters.

The computation of the accuracy can be reduced to computing the string edit distance. Using dynamic programming, the recognized string is matched with the string in the ground truth. This procedure directly returns the number of substitutions, deletions, and insertions. Note that the accuracy can be negative. The edit costs used in the HTK Software [66] for computing the string edit distance are

$$c_{\text{substitution}}(u \rightarrow u) = 0$$

$$c_{\text{substitution}}(u \rightarrow v) = 10$$

$$c_{\text{insertion}}(\varepsilon \rightarrow u) = 7$$

$$c_{\text{deletion}}(u \rightarrow \varepsilon) = 7$$

where  $u \neq v$  are two different elements and  $\varepsilon$  indicates the *empty word*.

---

## Conclusion

In this chapter, an overview of the current state-of-the-art research on continuous handwriting recognition is given, with a focus on off-line, English texts. Segmenting a text line into individual characters or just words can be a challenging task. Contextual information and sometimes even a semantic understanding of the content is needed, which can only be obtained after recognition. Therefore, modern recognition systems are all segmentation-free, sequential approaches which puts handwriting recognition closer to speech recognition than to the recognition of isolated machine-printed characters.

The normalization of the writing slant and the height of the three text areas reduces writing style variation. After normalizing the text line, it is converted into a sequence of feature vectors using a sliding window. Traditionally, recognition is

done using hidden Markov models. A recently proposed type of neural network, however, has introduced a novel recognition technology to the field. Interestingly, these BLSTM neural networks and recent ANN/HMM hybrid systems perform equally well. Further advances have been achieved by replacing more and more steps in the entire processing chain with learning-based systems instead of simple mathematical functions, such as text normalization or feature extraction.

Yet, it has become clear that the construction of a universal recognition system with 100 % accuracy is still impossible to reach in the near future. Open problems arise from the huge variety of different writing styles and the need to grasp the meaning of the text that is to be transcribed. Statistical language models do certainly not provide enough context knowledge and leave no room for out-of-vocabulary words.

---

## Notes

<sup>1</sup><http://www.iapr-tc11.org>

<sup>2</sup><http://www.iam.unibe.ch/fki/databases>

<sup>3</sup><http://www.rimes-database.fr>

<sup>4</sup><http://unipen.nici.kun.nl>

<sup>5</sup>George Washington Papers at the Library of Congress, 1741–1799: Series 2, Letterbook 1, pages 270–279 & 300–309, <http://memory.loc.gov/ammem/gwhtml/gwseries2.html>

<sup>6</sup><http://prhl.itи.upv.es/page/projects/multimodal/idoc>

<sup>7</sup><http://htk.eng.cam.ac.uk>

<sup>8</sup><http://www.cs.ubc.ca/~murphyk/Software/HMM/hmm.html>

<sup>9</sup><http://torch5.sourceforge.net>

<sup>10</sup><http://www.cs.waikato.ac.nz/ml/weka>

<sup>11</sup><http://mnl.sourceforge.net/>

<sup>12</sup><http://code.google.com/p/mitlm>

<sup>13</sup>[http://www.speech.cs.cmu.edu/SLM\\_info.html](http://www.speech.cs.cmu.edu/SLM_info.html)

<sup>14</sup><http://research.microsoft.com/apps/pubs/default.aspx?id=70505>

<sup>15</sup><http://www.speech.sri.com/projects/srilm>

---

## References

1. Arica N, Yarmal-Vural FT (2002) Optical character recognition for cursive handwriting. *IEEE Trans Pattern Anal Mach Intell* 24(6):801–814
2. Ahmed P, Suen CY (1987) Computer recognition of totally unconstrained handwritten zip codes. *Int J Pattern Recognit Artif Intell* 1(1):1–15
3. Baum LE, Petrie T, Soules G, Weiss N (1970) A maximization technique occurring in the statistical analysis of probabilistic functions of Markov chains. *Ann Math Stat* 41(1):164–171
4. Bazzi I, Schwartz R, Makhoul J (1999) An omnifont open-vocabulary OCR system for English and Arabic. *IEEE Trans Pattern Anal Mach Intell* 21(6):495–505

5. Bengio Y, Simard P, Frasconi P (1994) Learning long-term dependencies with gradient decent is difficult. *IEEE Trans Neural Netw* 5(2):157–166
6. Božinović RM, Srihari S (1989) Off-line cursive script word recognition. *IEEE Trans Pattern Anal Mach Intell* 11(1):68–83
7. Brakensiek A, Rigoll G (2004) Handwritten address recognition using hidden Markov models. In: Dengel A et al (eds) *Reading and learning*. Volume 2956 of lecture notes in computer science. Springer, Berlin/Heidelberg, pp 103–122
8. Caillault É, Viard-Gaudin C (2007) Mixed discriminant training of hybrid ANN/HMM. *Int J Pattern Recognit Artif Intell* 21(1):117–134
9. Chen J, Cao H, Prasad R, Bhardwaj A, Natarajan P (2010) Gabor features for offline Arabic handwriting recognition. In: 9th IARP international workshop on document analysis systems, Boston, pp 53–58
10. Collobert R, Bengio S, Mariéthoz J (2002) TORCH: a modular machine learning software library. Technical report IDIAP-RR 02-46, IDIAP Research Institute
11. Dietterich T (2009) Machine learning for sequential data: a review. In: Caelli T, Amin A, Duin R, de Ridder D, Kamel M (eds) *Structural, syntactic, and statistical pattern recognition*. Volume 2396 of lecture notes in computer science. Springer, Berlin/Heidelberg, pp 227–246
12. Do TMT, Artieres T (2006) Conditional random fields for online handwriting recognition. In: International workshop of frontiers in handwriting recognition, La Baule, pp 197–204
13. Duda RO, Hart PE, Stork DG (2001) *Pattern classification*, 2nd edn. Wiley-Interscience, New York
14. España-Boquera S, Castro-Bleda MJ, Gorbe-Moya J, Zamora-Martinez F (2010) Improving offline handwritten text recognition with hybrid HMM/ANN models. *IEEE Trans Pattern Anal Mach Intell* 33(4):767–779
15. Fischer A, Bunke H (2009) Kernel PCA for HMM-based cursive handwriting recognition. In: 13th international conference on computer analysis of images and pattern, Münster, pp 181–188
16. Fischer A, Riesen K, Bunke H (2010) Graph similarity features for HMM-based handwriting recognition in historical documents. In: 12th international conference on frontiers in handwriting recognition, Kolkata. pp 253–258
17. Fischer A, Keller A, Frinken V, Bunke H (2011, submitted) Lexicon-free handwritten word spotting using character HMMs pattern recognition letters 33(7):934–942
18. Frinken V, Fischer A, Mammatha R, Bunke H (2012) A novel word spotting method based on recurrent neural networks. *IEEE Trans Pattern Anal Mach Intell* 34(2), pp 211, 224
19. Gader PD, Mohamed MA, Chiang J-H (1995) Comparison of crisp and fuzzy character neural network in handwritten word recognition. *IEEE Trans Fuzzy Syst* 3(3):357–364
20. Gader PD, Mohamed MA, Chiang J-H (1997) Handwritten word recognition with character and inter-character neural networks. *IEEE Trans Syst Man Cybern* 27(1):158–164
21. Goodman JT (2001) A bit of progress in language modeling – extended version. Technical report MSR-TR-2001-72, Microsoft Research, One Microsoft Way Redmond, WA 98052, 8
22. Gorski N, Anisimov V, Augustin E, Baret O, Maximov S (2001) Industrial bank check processing: the A2iA CheckReader(tm). *Int J Doc Anal Recognit* 3(4):196–206
23. Graves A (2012) *Supervised sequence labelling with recurrent neural networks*. Springer, Heidelberg/New York/Dordrecht/London
24. Graves A, Schmidhuber J (2009) Offline handwriting recognition with multidimensional recurrent neural networks. In: Koller D et al (eds) *Advances in neural information processing systems 21*. MIT Press, Cambridge, pp 545–552
25. Graves A, Liwicki M, Fernández S, Bertolami R, Bunke H, Schmidhuber J (2009) A novel connectionist system for unconstrained handwriting recognition. *IEEE Trans Pattern Anal Mach Intell* 31(5):855–868
26. Guyon I, Schomaker L, Plamondon R, Liberman M, Janet S (1994) UNIPEN project of on-line data exchange and recognizer benchmarks. In: 12th international conference on pattern recognition, Jerusalem, pp 29–33

27. Hall M, Frank E, Holmes G, Pfahringer B, Reutemann P, Witten IH (2009) The WEKA data mining software: an update. *SIGKDD Explor* 11(1):10–18
28. Hochreiter S, Schmidhuber J (1997) Long short-term memory. *Neural Comput* 9(8):1735–1780
29. Jiang H (2005) Confidence measures for speech recognition: a survey. *Speech Commun* 45:455–470
30. Johanson S, Leech GN, Goodluck H (1978) Manual of information to accompany the Lancaster-Oslo/Bergen corpus of British English, for use with digital computers. Technical report, Department of English, University of Oslo, Norway
31. Kavallieratou E, Fakotakis N, Kokkinakis GK (2002) An unconstrained handwriting recognition system. *Int J Doc Anal Recognit* 4(4):226–242
32. Knerr S, Augustin E, Baret O, Price D (1998) Hidden Markov model based word recognition and its application to legal amount reading on French checks. *Comput Vis Image Underst* 70(3):404–419
33. Koerich AL, Sabourin R, Suen CY (2003) Lexicon-driven HMM decoding for large vocabulary handwriting recognition with multiple character models. *Int J Doc Anal Recognit* 6:126–144
34. Lee S-W (ed) (1999) Advances in handwriting recognition. World Scientific, Singapore/River Edge/London
35. Liwicki M, Graves A, Bunke H (2012) Neural networks for handwriting recognition. In: Ogiela MR, Jain LC (eds) Computational intelligence paradigms in advanced pattern classification, vol 386/2012. Springer, Berlin/Heidelberg, pp 5–24
36. Lorigo LM, Govindaraju V (2006) Offline Arabic handwriting recognition: a survey. *IEEE Trans Pattern Anal Mach Intell* 28(5):712–725
37. Marti U-V (2000) Off-line recognition of handwritten texts. PhD thesis, University of Bern, Bern
38. Marti U-V, Bunke H (2001) Using a statistical language model to improve the performance of an HMM-based cursive handwriting recognition system. *Int J Pattern Recognit Artif Intell* 15:65–90
39. Marti U-V, Bunke H (2002) The IAM-database: an English sentence database for offline handwriting recognition. *Int J Doc Anal Recognit* 5:39–46
40. Ogawa A, Takeda K, Itakura F (1998) Balancing acoustic and linguistic probabilities. In: International conference on acoustic, speech, and signal processing, Seattle, pp 181–184
41. O'Reilly RC, Frank MJ (2003) Making working memory work: a computational model of learning in the prefrontal cortex and basal ganglia. *Ics-03-03, ICS*, Mar 2003
42. Pechwitz M, Maergner V (2003) HMM based approach for handwritten Arabic word recognition using the IFN/ENIT-database. In: 7th international conference on document analysis and recognition, Edinburgh, pp 890–894
43. Plötz T, Fink G (2011) Markov models handwriting recognition. Springer, London/Dordrecht/Heidelberg/New York
44. Pramod Sankar K, Ambati V, Pratha L, Jawahar CV (2006) Digitizing a million books: challenges for document analysis. In: 7th IAPR workshop on document analysis systems, Nelson, pp 425–436
45. Rabiner L (1989) A tutorial on hidden Markov models and selected applications in speech recognition. *Proc IEEE* 77(2):257–286
46. Rath TM, Mammatha R (2007) Word spotting for historical documents. *Int J Doc Anal Recognit* 9:139–152
47. Rodríguez JA, Perronnin F (2008) Local gradient histogram features for word spotting in unconstrained handwritten documents. In: 11th international conference frontiers in handwriting recognition, Montréal, pp 7–12
48. Rodríguez-Serrano JA, Perronnin F, Sánchez G, Lladós J (2010) Unsupervised writer adaptation of whole-word HMMs with application to word-spotting. *Pattern Recognit Lett* 31(8):742–749
49. Rutovitz D (1966) Pattern recognition. *J R Stat Soc A (General)* 129(4):504–530
50. Sayre KM (1973) Machine recognition of handwritten words: a project report. *Pattern Recognit* 3(3):213–228

51. Seiler R, Schenkel M, Eggimann F (1996) Off-line cursive handwriting recognition compared with on-line recognition. In: 13th international conference on pattern recognition, Vienna, vol 4, pp 505–509
52. Senior AW, Robinson AJ (1998) An off-line cursive handwriting recognition system. *IEEE Trans Pattern Anal Mach Intell* 20(3):309–321
53. Srihari SN, Srinivasan H, Huang C, Shetty S (2006) Spotting words in Latin, Devanagari and Arabic scripts. *Indian J Artif Intell* 16(3):2–9
54. Stolke A (2002) SRILM – an extensible language modeling toolkit. In: International conference on spoken language processing, Denver, pp 901–904
55. Taira E, Uchida S, Sakoe H (2004) Nonuniform slant correction for handwritten word recognition. *IEICE Trans Inf Syst* E87-D(5):1247–1253
56. Terasawa K, Tanaka Y (2009) Slit style HOG features for document image word spotting. In: 10th international conference on document analysis and recognition, Barcelona, vol 1, pp 116–120
57. Toselli AH, Juan A, González J, Salvador I, Vidal E, Casacuberta F, Keysers D, Ney H (2004) Integrated handwritten recognition and interpretation using finite-state models. *Int J Pattern Recognit Artif Intell* 18(4):519–539
58. Toselli AH, Vidal E, Casacuberta F (2011) Multimodal interactive pattern recognition and applications. Springer, London/New York
59. van der Zant T, Schomaker L, Haak K (2008) Handwritten-word spotting using biologically inspired features. *IEEE Trans Pattern Anal Mach Intell* 30(11):1945–1957
60. Vinciarelli A (2002) A survey on off-line cursive word recognition. *Pattern Recognit* 35(7):1433–1446
61. Vinciarelli A (2003) Offline cursive handwriting: from word to text recognition. Technical report IDIAP-PP 03-24, Institut Dalle Molle Intelligence Artificielle Perceptive (IDIAP), Martigny
62. Vinciarelli A, Bengio S, Bunke H (2004) Offline recognition of unconstrained handwritten texts using HMMs and statistical language models. *IEEE Trans Pattern Anal Mach Intell* 26(6): 709–720
63. Viterbi A (1967) Error bounds for convolutional codes and an asymptotically optimum decoding algorithm. *IEEE Trans Inf Theory* 13:260–269
64. Werbos PJ (1988) Generalization of backpropagation to a recurrent gas model. *Neural Netw* 1:339–356
65. Wienecke M, Fink GA, Gerhard S (2005) Toward automatic video-based whiteboard reading. *Int J Doc Anal Recognit* 7:188–200
66. Young S, Evermann G, Gales M, Hain T, Kershaw D, Liu X, Moore G, Odell J, Ollason D, Povey D, Valtchev V, Woodland P (2006) The HTK book. Technical report, Cambridge University Engineering Department, Dec 2006

## Further Reading

- Fischer A, Keller A, Frinken V, Bunke H (2011, submitted) Lexicon-free handwritten word spotting using character HMMs
- Frinken V, Fischer A, Manmatha R, Bunke H (2012) A novel word spotting method based on recurrent neural networks. *IEEE Trans Pattern Anal Mach Intell*. Accepted for publication
- Goodman JT (2001) A bit of progress in language modeling – extended version. Technical report MSR-TR-2001-72, Microsoft Research, One Microsoft Way Redmond, WA 98052, 8
- Graves A (2012) Supervised sequence labelling with recurrent neural networks. Springer, Heidelberg/New York/Dordrecht/London
- Jiang H (2005) Confidence measures for speech recognition: a survey. *Speech Commun* 45: 455–470

- Liwicki M, Graves A, Bunke H (2012) Neural networks for handwriting recognition. In: Ogiela MR, Jain LC (eds) Computational intelligence paradigms in advanced pattern classification, vol 386/2012. Springer, Berlin/Heidelberg, pp 5–24
- Plötz T, Fink G (2011) Markov models handwriting recognition. Springer, London/Dordrecht/Heidelberg/New York
- Rabiner L (1989) A tutorial on hidden Markov models and selected applications in speech recognition. Proc IEEE 77(2):257–286
- Toselli AH, Vidal E, Casacuberta F (2011) Multimodal interactive pattern recognition and applications. Springer, London/New York

Abdel Belaïd and Mohamed Imran Razzak

## Contents

Introduction.....	428
Writing Systems.....	429
Middle Eastern Writing Systems.....	430
Syriac Writing System.....	430
Arabic Writing System.....	432
Hebrew Writing System.....	439
Writing Recognition Systems.....	441
Preprocessing.....	441
Word Segmentation.....	442
Isolated Character Recognition.....	444
Word Recognition.....	445
Success Clues of the Topic.....	448
Datasets.....	448
Academic Systems.....	448
Commercial Systems.....	450
Conclusion.....	452
Cross-References.....	453
References.....	453
Further Reading.....	457

---

A. Belaïd (✉)

Université de Lorraine – LORIA, Vandœuvre-lès-Nancy, France

e-mail: [abdel.belaïd@loria.fr](mailto:abdel.belaïd@loria.fr)

M.I. Razzak

Department of Computer Science and Engineering, Air University, Pakistan

College of Public Health and Health Informatics, King Saud bin Abdulaziz University for Health Sciences, Riyadh, Saudi Arabia

e-mail: [imranrazzak@hotmail.com](mailto:imranrazzak@hotmail.com)

**Abstract**

The purpose of this chapter is to give a rapid synthesis of the state of the art concerning Middle Eastern character recognition systems. This includes the presentation of the different scripts used, their characteristics, the techniques used especially for them, and the difficulties and challenges faced by researchers to treat them. The chapter will also make a rapid review of the best systems evaluated during international competitions, of published datasets specialized in these scripts, as well as of the list of existing commercial systems.

**Keywords**

Middle Eastern writing systems • Arabic • Hebrew • Syriac • Thaana • Word segmentation and recognition • Character recognition

---

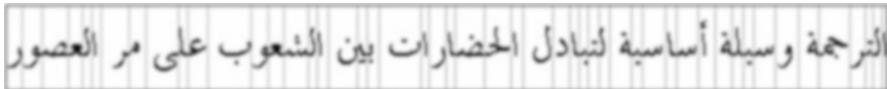
## Introduction

The scripts used in the Middle East mainly include Arabic, Hebrew, Syriac, and Thaana. They have a common origin dating back to the ancient Phoenician alphabet. They are all alphabetical with small character sets. As a consequence, there are a lot of morphological similarities which have to be taken into account in handwriting recognition systems.

**Main Differences with Latin:** Middle Eastern scripts are more complex than Latin scripts. The complexity comes from their inherent morphology, such as context-sensitive shape, cursiveness, and overlapping of word parts. Except for Thaana, they all include diacritical marks. These marks accompany the characters, either above, below, or inside, in order to represent vowel sounds. The text is written from right to left. Among these scripts, Arabic and Syriac are cursive even when they are typeset. In those scripts, the character shapes change according to the relative position with respect to the word, as opposed to Hebrew and Thaana. Rules for text rendering are specified in the block for Arabic and Syriac, whereas this is not required for Hebrew, as only five letters have position-dependent shapes and final shapes are coded separately.

**Challenges:** Context identification is necessary prior to recognition, since the character shape changes according to the surrounding characters. However, this is not straightforward, especially when the ligature is vertical, which can happen frequently when the text is handwritten. Moreover, the size of each ligature varies due to cursiveness and size of neighbor characters. Thus, the ligature has no symmetry in height and width, which makes the very similar contours difficult to segment and to recognize [8] (cf. Fig. 13.1).

Another source of complexity is the extensive amount of dots associated with characters. Their positions vary with respect to corresponding characters.



**Fig. 13.1** Grapheme segmentation process illustrated by manually inserting *vertical lines* at the appropriate grapheme connection points

For example, they are often attached to characters and sometimes depicted as small lines. As a consequence, baseline detection affects segmentation heavily and obviously. Different graphemes do not have a fixed height or a fixed width. Moreover, neither the different nominal sizes of the same font scale linearly with their actual line heights nor the different fonts with the same nominal size have a fixed line height.

Finally, words are composed of sub-words, which can be considered as an advantage, but often these sub-words provide some additional confusion in segmentation because of non-heterogeneous white space separation between them.

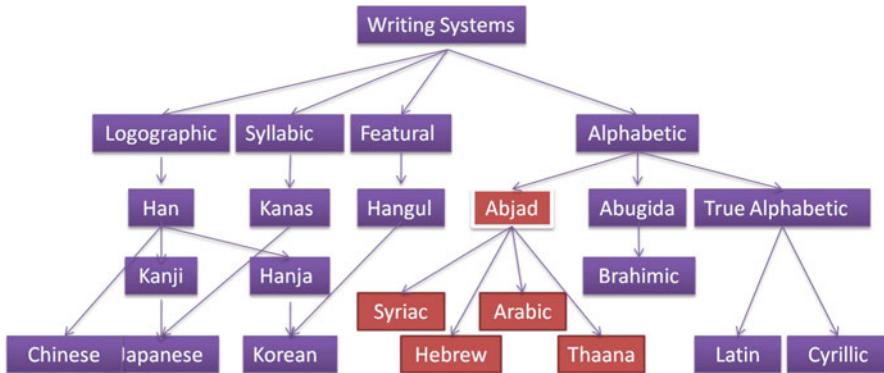
The remainder of this chapter is organized as follows. Section “[Writing Systems](#)” presents background on the writing systems, describing their derivation tree from the origin. Section “[Middle Eastern Writing Systems](#)” presents more writing systems of the Middle East by giving details about each of them such as the scripts and the morphological peculiarities. It also includes tables that classify the scripts according to their writing aspects. Section “[Writing Recognition Systems](#)” describes the writing recognition systems. It outlines the different steps needed for designing recognition systems. Section “[Success Clues of the Topic](#)” provides some clues to the success of this research through shared databases, the academic systems and the number of commercial systems. Section “[Conclusion](#)” proposes some observations about the scripts studied and their concomitant recognition systems.

---

## Writing Systems

According to Ghosh et al. [40], there are four classes of writing systems: logographic, syllabic, featural, and alphabetic (see Fig. 13.2), described as follows:

- The logographic script refers to ideograms representing complete words as “Han” which is mainly associated with Chinese, Japanese, and Korean writings. They are composed of several harmoniously associated strokes.
- In the syllabic system, each symbol represents a phonetic sound or syllable. This is the case of Japanese which uses both logographic Kanji and syllabic Kanas. The stroke gathering is less dense than in the logographic scripts.
- In the featural alphabet, the shapes of the letters encode phonological features of the phonemes represented. This is the case of Korean Hangul, where the five main consonants repeat the shape of the lips and tongue when producing the corresponding sound and remind the five basic elements of eastern philosophy.
- The alphabetic class consists of characters reproducing phonemes of the spoken language. Each class is subdivided into three subclasses: true alphabetic,



**Fig. 13.2** Writing systems (From [40])

ABUGIDA, and ABJAD. The true alphabetic corresponds to Latin and Cyrillic. ABUGIDA is related to Brahmic scripts, while ABJAD refers to Arabic and Hebrew.

In fact, ABJAD (أبجَد) is an acronym derived from the first four consonants of the Hebrew/Arabic/Persian alphabets: “Alif,” “Ba,” “Jeem,” and “Dal.” It expresses the fact that the vowels are not spelled out and characters are proposed “essentially” for consonantal sounds. It contrasts with ABUGIDA in which each symbol represents a syllable (consonant+vowel). This is similar in Hindi or Amharic, as well as an alphabet where each symbol represents a smaller unit of sound and the vowels are not omitted.

## Middle Eastern Writing Systems

The Middle Eastern writing systems are part of the ABJAD writing system and have the same Phoenician origin. They include Arabic, Hebrew, Syriac, and Thaana. This group is divided into East Semitic (including the Akkadian), North west Semitic (Canaanite Phoenician, Hebrew, Aramaic), and Southern Semitic. Arabic is the latter group with the Ethiopian and South Arabian groups.

### Syriac Writing System

The Syriac language has been primarily used since around the second century BC as one of the Semitic ABJADS directly descending from the Aramaic alphabet. It shares similarities with the Phoenician, Hebrew, and Arabic alphabets.

The alphabet has 22 letters that can be linked or not, depending of their position in the word. There are three main types of fonts: Estrangelo; Western, Jacobite or Serto script; and Eastern or Nestorian script (see Figs. 13.3–13.5). Serto and Estrangelo

kap	yodh	téith	héith	zâyn	waw	hé	dalâth	gamâl	béith	alâp

taw	sheen	rész	qop	şadhe	pé	'ain	simkâth	nun	meem	lamâdh

Fig. 13.3 Syriac Estrangelo (From <http://www.omniglot.com/writing/syriac.htm>)

kap	yodh	téith	héith	zâyn	waw	hé	dalâth	gamâl	béith	alâp

taw	sheen	rész	qop	şadhe	pé	'ain	simkâth	nun	meem	lamâdh

Fig. 13.4 Western, Jacobite or Serto script (From <http://www.omniglot.com/writing/syriac.htm>)

kap	yodh	téith	héith	zâyn	waw	hé	dalâth	gamâl	béith	alâp

taw	sheen	rész	qop	şadhe	pé	'ain	simkâth	nun	meem	lamâdh

Fig. 13.5 Eastern or Nestorian script (From <http://www.omniglot.com/writing/syriac.htm>)

differ both in style and use. The Estrangelo script is used more for titles, whereas Serto is used for the body of the documents.

The letters are cursive, depending on where in the word or sub-word they are located, and thus take different shape. They can be pointed with vowel diacritics but can be written unpointed, which means that semantics are required for interpretation. Figure 13.6 shows a Syriac sample text in Nestorian script.

حُكْمٌ لَّكُنْهُ هَذِهِ لَفْظَهُ تَجْعَلُهُ تَمَّ هَذِهِ  
 هَمَّ لَفْظَهُمْ. حُكْمٌ لَّكُنْهُمْ. حُكْمٌ لَّكُنْهُنَّهُمْ لَمَّا  
 لَمَّا حَسَّنُوا حَسَّنَوْهُمْ لَمَّا قَدِّمُوا قَدِّمَهُمْ لَمَّا  
 قَدِّمُوا هُنَّهُمْ. لَمَّا حَسَّنُوا حَسَّنَهُمْ لَمَّا  
 قَدِّمُوا هُنَّهُمْ

**Fig. 13.6** Syriac text (From <http://www.omniglot.com/writing/syriac.htm>)

As reported in [67], some characters seem quite similar, with only small differences. This is, for instance, the case of “dalâth” and “résh” “taw” in the beginning of a word or sub-word and “alâp” at the end of a word or sub-word; “béith” and “kap” in the middle of a word or sub-word and at the beginning of a word or sub-word; and “waw,” “qop,” “lamâdh,” “héith,” and “yodh.” The characters “dalâth” and “resh” are differentiated by the presence of a single dot. Such a brittle feature does not provide high confidence in recognition. Other than the location of the dot, both “dalâth” and “resh” appear as identical. The starting character “taw” and the finishing one “alâp” have similar structures. They are both represented by a vertical line with a small stroke on the right hand side. However, the “alâp” character has a small stroke or tail that does not finish at the point where it joins the horizontal stroke. Table 13.1 traces the similarities and differences between letters.

As in other Semitic languages, Syriac words are built out of trilateral roots, which is a collection of three Syriac consonants and vowels serving as a “glue.” Figure 13.7 shows an example of words generated from the root.

## Arabic Writing System

The languages written with the Arabic writing system are mentioned in Table 13.2. Its evolution is traced in the following paragraphs:

**The Origin:** Arabic writing appeared in the sixth century. It took its origins in the Phoenician script. It has inherited its 22 letters and the use of ligatures to connect a letter to the next. Therefore, most Arabic letters can take four forms: isolated, initial, medial, and final.

**The Koranic Addition:** The arrival of Islam and the idea to make the Koran a stable reference have profoundly marked the history of the Arabic writing. Several challenges have emerged, particularly in terms of lack of expressiveness of the language with just 22 letters (see Fig. 13.8). Thus, in addition to the 22 letters of the Phoenician alphabet, Arab philologists invented 6 new letters.

**Table 13.1** Aramaic letters that look alike (From <http://allthingsaramaic.com/aramaic-letters-alike.php>)

Characters	Differences and resemblances
 (Beet) and  (Kap)	Kap is more rounded than Beet. Beet has straight lines with sharp edges, and the line at the bottom right corner sticks out
 (Gamal),  (Noon) and  (Kap)	Gamal has a “foot” that exceeds on the right. Noon is square on the edges, without foot. Both Gamal and Noon have about half the width of other letters; for cons, Kap is larger and rounder
 (Dalat),  (Resh) and  (Final Kap)	Dalat has a tittle which sticks out at the top right corner, whereas Resh is rounded. Dalat is squarer than Resh; they are different from Kap (final) because they remain above the baseline
 (Heh),  (Chet) and  (Taw)	Heh and Chet have similar shapes, but Heh has a gap in the left vertical line, whereas Chet does not, are different from Taw which is more rounded at the top right corner
 (Waw),  (Zeyn) and  (Final Noon)	Final Noon can only ever occur at the end of an Aramaic word. It goes below the lower guide line. Its vertical line is generally slightly sloped, unlike Waw and Zeyn
 (Yood) and  (Waw)	Has the gap at the bottom left corner, but Tet has the gap at the top left. The line of Tet extends almost into the middle of the letter, but this does not happen with Meem
 (Meem) and  (Tet)	Has the gap at the bottom left corner, but Tet has the gap at the top left. The line of Tet extends almost into the middle of the letter, but this does not happen with Meem
 (Final Meem) and  (Semkat)	Has the gap at the bottom left corner, but Tet has the gap at the top left. The line of Tet extends almost into the middle of the letter, but this does not happen with Meem
 (Sadeh),  (Ev) and  (Final Sadeh)	Has the gap at the bottom left corner, but Tet has the gap at the top left. The line of Tet extends almost into the middle of the letter, but this does not happen with Meem

彖 – šqal: "he has taken"  
彖 – nešqōl: "he will take"  
彖 – šāqeł: "he takes, he is taking"  
彖 – šaqqeł: "he has lifted/raised"  
彖 – ašqeł: "he has set out"  
彖 – šqālā: "a taking, burden, recension, portion or syllable"  
彖 – šeqlē: "takings, profits, taxes"  
彖 – šaqlülä: "a beast of burden"  
彖 – šūqālā: "arrogance"

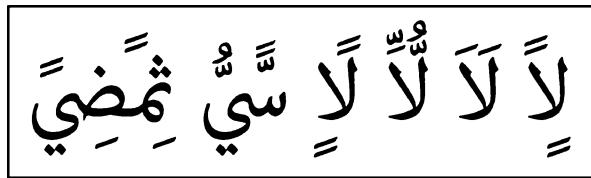
**Fig. 13.7** Syriac words  
(From [http://en.wikipedia.org/wiki/Syriac\\_language](http://en.wikipedia.org/wiki/Syriac_language))

**Table 13.2** List of languages used in the framework of the writing system

Writing system	Languages
Arabic	Azeri (Iran), Balochi, Baluchi, Beja, Berber, Dari, Fulani, Hausa, Judeo-Spanish (until the twentieth century), Kabyle, Kanuri (on occasion), Konkani, Kashmiri, Kazakh in China, Kurdish (Iran and Iraq), Kyrgyz, Malagasy (until the nineteenth century), Malay (fourteenth to seventeenth century), Mandekan, Mazanderani, Morisco, Mozarabic (now extinct), Ottoman Turkish, Pashtu, Pashto, Persian/Farsi, Punjabi (Pakistan), Rajasthani, Saraiki, Shabaki, Sindhi, Spanish (formerly before sixteenth century, a.k.a. Aljamiado), Swahili (on occasion), Tatar, Tajik (on occasion), Tausug, Urdu, Uyghur
Hebrew	Aramaic, Bukhori, Hulaula, Judeo-Berber, Judeo-Iraqi Arabic, Judeo-Moroccan, Judeo-Tripolitanian Arabic, Judeo-Tunisian Arabic, Judeo-Portuguese, Judeo-Spanish, Judeo-Yemenite, Juhuri, Lishan Didan, Lishana Deni, Lishanid Noshan, Shuadit, Yiddish, Zaphritic
Syriac	Garshuni, Assyrian Neo-Aramaic, Bohtan Neo-Aramaic, Chaldean Neo-Aramaic, Hertevin, Koy Sanjaq Surat, Senaya, Syriac, Turoyo
Thaana	Dhivehi

1	ف	ج	ل	أ	م	ي	ه	ك	د	س	و	ز	ت	ب	خ	ض	ظ	غ	ن	ع	ص	ق	ر	ش	م	ل	ح	ط	ي	ك	ف	س	و	ز	ه	ج	ل	أ	م	ي	ف	خ	ض	ظ	غ	ن	ع	ص	ق	ر	ش	م	ل	ح	ط	ي	ك	ف	س	و	ز	ه	ج	ل	أ	م	ي	ف	خ	ض	ظ	غ	ن	ع	ص	ق	ر	ش	م	ل	ح	ط	ي	ك	ف	س	و	ز	ه	ج	ل	أ	م	ي	ف	خ	ض	ظ	غ	ن	ع	ص	ق	ر	ش	م	ل	ح	ط	ي	ك	ف	س	و	ز	ه	ج	ل	أ	م	ي	ف	خ	ض	ظ	غ	ن	ع	ص	ق	ر	ش	م	ل	ح	ط	ي	ك	ف	س	و	ز	ه	ج	ل	أ	م	ي	ف	خ	ض	ظ	غ	ن	ع	ص	ق	ر	ش	م	ل	ح	ط	ي	ك	ف	س	و	ز	ه	ج	ل	أ	م	ي	ف	خ	ض	ظ	غ	ن	ع	ص	ق	ر	ش	م	ل	ح	ط	ي	ك	ف	س	و	ز	ه	ج	ل	أ	م	ي	ف	خ	ض	ظ	غ	ن	ع	ص	ق	ر	ش	م	ل	ح	ط	ي	ك	ف	س	و	ز	ه	ج	ل	أ	م	ي	ف	خ	ض	ظ	غ	ن	ع	ص	ق	ر	ش	م	ل	ح	ط	ي	ك	ف	س	و	ز	ه	ج	ل	أ	م	ي	ف	خ	ض	ظ	غ	ن	ع	ص	ق	ر	ش	م	ل	ح	ط	ي	ك	ف	س	و	ز	ه	ج	ل	أ	م	ي	ف	خ	ض	ظ	غ	ن	ع	ص	ق	ر	ش	م	ل	ح	ط	ي	ك	ف	س	و	ز	ه	ج	ل	أ	م	ي	ف	خ	ض	ظ	غ	ن	ع	ص	ق	ر	ش	م	ل	ح	ط	ي	ك	ف	س	و	ز	ه	ج	ل	أ	م	ي	ف	خ	ض	ظ	غ	ن	ع	ص	ق	ر	ش	م	ل	ح	ط	ي	ك	ف	س	و	ز	ه	ج	ل	أ	م	ي	ف	خ	ض	ظ	غ	ن	ع	ص	ق	ر	ش	م	ل	ح	ط	ي	ك	ف	س	و	ز	ه	ج	ل	أ	م	ي	ف	خ	ض	ظ	غ	ن	ع	ص	ق	ر	ش	م	ل	ح	ط	ي	ك	ف	س	و	ز	ه	ج	ل	أ	م	ي	ف	خ	ض	ظ	غ	ن	ع	ص	ق	ر	ش	م	ل	ح	ط	ي	ك	ف	س	و	ز	ه	ج	ل	أ	م	ي	ف	خ	ض	ظ	غ	ن	ع	ص	ق	ر	ش	م	ل	ح	ط	ي	ك	ف	س	و	ز	ه	ج	ل	أ	م	ي	ف	خ	ض	ظ	غ	ن	ع	ص	ق	ر	ش	م	ل	ح	ط	ي	ك	ف	س	و	ز	ه	ج	ل	أ	م	ي	ف	خ	ض	ظ	غ	ن	ع	ص	ق	ر	ش	م	ل	ح	ط	ي	ك	ف	س	و	ز	ه	ج	ل	أ	م	ي	ف	خ	ض	ظ	غ	ن	ع	ص	ق	ر	ش	م	ل	ح	ط	ي	ك	ف	س	و	ز	ه	ج	ل	أ	م	ي	ف	خ	ض	ظ	غ	ن	ع	ص	ق	ر	ش	م	ل	ح	ط	ي	ك	ف	س	و	ز	ه	ج	ل	أ	م	ي	ف	خ	ض	ظ	غ	ن	ع	ص	ق	ر	ش	م	ل	ح	ط	ي	ك	ف	س	و	ز	ه	ج	ل	أ	م	ي	ف	خ	ض	ظ	غ	ن	ع	ص	ق	ر	ش	م	ل	ح	ط	ي	ك	ف	س	و	ز	ه	ج	ل	أ	م	ي	ف	خ	ض	ظ	غ	ن	ع	ص	ق	ر	ش	م	ل	ح	ط	ي	ك	ف	س	و	ز	ه	ج	ل	أ	م	ي	ف	خ	ض	ظ	غ	ن	ع	ص	ق	ر	ش	م	ل	ح	ط	ي	ك	ف	س	و	ز	ه	ج	ل	أ	م	ي	ف	خ	ض	ظ	غ	ن	ع	ص	ق	ر	ش	م	ل	ح	ط	ي	ك	ف	س	و	ز	ه	ج	ل	أ	م	ي	ف	خ	ض	ظ	غ	ن	ع	ص	ق	ر	ش	م	ل	ح	ط	ي	ك	ف	س	و	ز	ه	ج	ل	أ	م	ي	ف	خ	ض	ظ	غ	ن	ع	ص	ق	ر	ش	م	ل	ح	ط	ي	ك	ف	س	و	ز	ه	ج	ل	أ	م	ي	ف	خ	ض	ظ	غ	ن	ع	ص	ق	ر	ش	م	ل	ح	ط	ي	ك	ف	س	و	ز	ه	ج	ل	أ	م	ي	ف	خ	ض	ظ	غ	ن	ع	ص	ق	ر	ش	م	ل	ح	ط	ي	ك	ف	س	و	ز	ه	ج	ل	أ	م	ي	ف	خ	ض	ظ	غ	ن	ع	ص	ق	ر	ش	م	ل	ح	ط	ي	ك	ف	س	و	ز	ه	ج	ل	أ	م	ي	ف	خ	ض	ظ	غ	ن	ع	ص	ق	ر	ش	م	ل	ح	ط	ي	ك	ف	س	و	ز	ه	ج	ل	أ	م	ي	ف	خ	ض	ظ	غ	ن	ع	ص	ق	ر	ش	م	ل	ح	ط	ي	ك	ف	س	و	ز	ه	ج	ل	أ	م	ي	ف	خ	ض	ظ	غ	ن	ع	ص	ق	ر	ش	م	ل	ح	ط	ي	ك	ف	س	و	ز	ه	ج	ل	أ	م	ي	ف	خ	ض	ظ	غ	ن	ع	ص	ق	ر	ش	م	ل	ح	ط	ي	ك	ف	س	و	ز	ه	ج	ل	أ	م	ي	ف	خ	ض	ظ	غ	ن	ع	ص	ق	ر	ش	م	ل	ح	ط	ي	ك	ف	س	و	ز	ه	ج	ل	أ	م	ي	ف	خ	ض	ظ	غ	ن	ع	ص	ق	ر	ش	م	ل	ح	ط	ي	ك	ف	س	و	ز	ه	ج	ل	أ	م	ي	ف	خ	ض	ظ	غ	ن	ع	ص	ق	ر	ش	م	ل	ح	ط	ي	ك	ف	س	و	ز	ه	ج	ل	أ	م	ي	ف	خ	ض	ظ	غ	ن	ع	ص	ق	ر	ش	م	ل	ح	ط	ي	ك	ف	س	و	ز	ه	ج	ل	أ	م	ي	ف	خ	ض	ظ	غ	ن	ع	ص	ق	ر	ش	م	ل	ح	ط	ي	ك	ف	س	و	ز	ه	ج	ل	أ	م	ي	ف	خ	ض	ظ	غ	ن	ع	ص	ق	ر	ش	م	ل	ح	ط	ي	ك	ف	س	و	ز	ه	ج	ل	أ	م	ي	ف	خ	ض	ظ	غ	ن	ع	ص	ق	ر	ش	م	ل	ح	ط	ي	ك	ف	س	و	ز	ه	ج	ل	أ	م	ي	ف	خ	ض	ظ	غ	ن	ع	ص	ق	ر	ش	م	ل	ح	ط	ي	ك	ف	س	و	ز	ه	ج	ل	أ	م	ي	ف	خ	ض	ظ	غ	ن	ع	ص	ق	ر	ش	م	ل	ح	ط	ي	ك	ف	س	و	ز	ه	ج	ل	أ	م	ي	ف	خ	ض	ظ	غ	ن	ع	ص	ق	ر	ش	م	ل	ح	ط	ي	ك	ف	س	و	ز	ه	ج	ل	أ	م	ي	ف	خ	ض	ظ	غ	ن	ع	ص	ق	ر	ش	م	ل	ح	ط	ي	ك	ف	س	و	ز	ه	ج	ل	أ	م	ي	ف	خ	ض	ظ	غ	ن	ع	ص	ق	ر	ش	م	ل	ح	ط	ي	ك	ف	س	و	ز	ه	ج	ل	أ	م	ي	ف	خ	ض	ظ	غ	ن	ع	ص	ق	ر	ش	م	ل	ح	ط	ي	ك	ف	س	و	ز	ه	ج	ل	أ	م	ي	ف	خ	ض	ظ	غ	ن	ع	ص	ق	ر	ش	م	ل	ح	ط	ي	ك	ف	س	و	ز	ه	ج	ل	أ	م	ي	ف	خ	ض	ظ	غ	ن	ع	ص	ق	ر	ش	م	ل	ح	ط	ي	ك	ف	س	و	ز	ه	ج	ل	أ	م	ي	ف	خ	ض	ظ	غ	ن	ع	ص	ق	ر	ش	م	ل	ح	ط	ي	ك	ف	س	و	ز	ه	ج	ل	أ	م	ي	ف	خ	ض	ظ	غ	ن	ع	ص	ق	ر	ش	م	ل	ح	ط	ي	ك	ف	س	و	ز	ه	ج	ل	أ	م	ي	ف	خ	ض	ظ	غ	ن	ع	ص	ق	ر	ش	م	ل	ح	ط	ي	ك	ف	س	و	ز	ه	ج	ل	أ	م	ي	ف	خ	ض	ظ	غ	ن	ع	ص	ق	ر	ش	م	ل	ح	ط	ي	ك	ف	س	و	ز	ه	ج	ل	أ	م	ي	ف	خ	ض	ظ	غ	ن	ع	ص	ق	ر	ش	م	ل	ح	ط	ي	ك	ف	س	و	ز	ه	ج	ل	أ	م	ي	ف	خ	ض	ظ	غ	ن	ع	ص	ق	ر	ش	م	ل	ح	ط	ي	ك	ف	س	و	ز	ه	ج	ل	أ	م	ي	ف	خ	ض	ظ	غ	ن	ع	ص	ق	ر	ش	م	ل	ح	ط	ي	ك	ف	س	و	ز	ه	ج	ل	أ	م	ي	ف	خ	ض	ظ	غ	ن	ع	ص	ق	ر	ش	م	ل	ح	ط	ي	ك	ف	س	و	ز	ه	ج	ل	أ	م	ي	ف	خ	ض	ظ	غ	ن	ع	ص	ق	ر	ش	م	ل	ح	ط	ي	ك	ف	س	و	ز	ه	ج	ل	أ	م	ي	ف	خ	ض	ظ	غ	ن	ع	ص	ق	ر	ش	م	ل	ح	ط	ي	ك	ف	س	و	ز	ه	ج	ل	أ	م	ي	ف	خ	ض	ظ	غ	ن	ع	ص	ق	ر	ش	م	ل	ح	ط	ي	ك	ف	س	و	ز	ه	ج	ل	أ	م	ي	ف	خ	ض	ظ	غ	ن	ع	ص	ق	ر	ش	م	ل	ح	ط	ي	ك	ف	س	و	ز	ه	ج	ل	أ	م	ي	ف	خ	ض	ظ	غ	ن	ع	ص	ق	ر	ش	م	ل	ح	ط	ي	ك	ف	س	و	ز	ه	ج	ل	أ	م	ي	ف	خ	ض	ظ	غ	ن	ع	ص	ق	ر	ش	م	ل	ح	ط	ي	ك	ف	س	و	ز	ه	ج	ل	أ	م	ي	ف	خ	ض	ظ	غ	ن	ع	ص	ق	ر	ش	م	ل	ح	ط	ي	ك	ف	س	و	ز	ه	ج	ل	أ	م	ي	ف	خ	ض	ظ	غ	ن	ع	ص	ق	ر	ش	م	ل	ح	ط	ي	ك	ف	س	و	ز	ه	ج	ل	أ	م	ي	ف	خ	ض	ظ	غ	ن	ع	ص	ق	ر	ش	م	ل	ح	ط	ي	ك	ف	س	و	ز	ه	ج	ل	أ	م	ي	ف	خ	ض	ظ	غ	ن	ع	ص	ق	ر	ش	م	ل	ح	ط	ي	ك	ف	س	و	ز	ه	ج	ل	أ	م	ي	ف	خ	ض	ظ	غ	ن	ع	ص	ق	ر	ش	م	ل	ح	ط	ي	ك	ف	س	و	ز	ه	ج	ل	أ	م	ي	ف	خ	ض	ظ	غ	ن	ع	ص	ق	ر	ش	م	ل	ح	ط	ي	ك	ف	س	و	ز	ه	ج	ل	أ	م	ي	ف	خ	ض	ظ	غ	ن	ع	ص	ق	ر	ش	م	ل	ح	ط	ي	ك	ف	س	و	ز	ه	

**Fig. 13.10** Arabic vocalization signs



**Vocalization:** Since Arabic script is a purely consonantal writing, ten signs were invented to clarify the pronunciation. There are three brief vowels and seven orthographic signs added above and below consonants (see Fig. 13.10).

**Arabic Extension:** Associated to the Islamization movement, the Arabic alphabet was spread to countries speaking languages which do not belong to the Semitic family. This led to the need for new letters, to represent a variety of sounds which were absent in standard Arabic scripts. As a consequence, new shapes were invented to represent new sounds, instead of borrowing from other scripts. Some languages have borrowed all the Arabic letters (28 consonants plus “hamza”) such as Persian, Ottoman Turkish, Urdu, and Pashto. Others have borrowed only a few, such as Kurdish Sorani, which borrowed only 21 letters. The new letters are formed by combining different dots with existing basic letters. It appears that this enhancement seems natural even instinctive, to many of those who use the Arabic script. To make some unknown phonemes in Arabic such as “p,” “v,” “g,” and “Tch,” new letters were invented by adding dots and diacritical marks to Arabic letters, by the closest pronunciation. They are summarized as follows:

- Persian and Urdu: “p” – b with three dots below
- Persian and Urdu: “ch” – j with three dots below
- Persian and Urdu: “g” – k with a double top
- Persian and Urdu: “zh” – z with three dots above
- In Egypt: “g” – j. That is because Egyptian Arabic has g where other Arabic dialects have j
- In Egypt: j – j with three dots below, same as Persian and Urdu “ch”
- In Egypt: “ch” – written as “t-sh”
- Urdu: retroflex sounds – as the corresponding dentals but with a small letter above (This problem in adapting a Semitic alphabet to write Indian languages also arose long before this: see Brahmi.)

For more clarity, Fig. 13.11 shows visual illustration of those aforementioned cases.

**Vowels:** The vowel notation is the main difficulty to which the introducers of the Arabic alphabet were faced. Despite a large number of consonants, the Arabic script uses only three signs to rate the three Arabic short vowels. These signs are added either above or below the consonants. Therefore, such a system, applied to languages where the vowels are numerous, gives unsatisfactory results. As an example, this may be the primary reason for the Turkish alphabet to move to Latin.

	Persian/ Farsi	Urdu	Malay	Sindhi	Kurdish	Uyghur	Arwi	Egyptian
p	پ	ف		پ			ب	
ch		ج					ج	
g			گ				ك	ج
zh	ژ						ڙ	
j			ج					ج
ng			غ			ڭ		
ny			ڻ					
v			و		ق	ۋ		
retroflex		ڦ, ڤ						

**Fig. 13.11** Adapting the Arabic alphabet for other languages: [http://en.wikipedia.org/wiki/History\\_of\\_the\\_Arabic\\_alphabet](http://en.wikipedia.org/wiki/History_of_the_Arabic_alphabet)

	Vowel description	I.	M.	F.	Is.
ئا (a)	Slightly rounded long	ئا	ا	ا	ئا
ئو (o)	Oral rounded long	ئو	و	و	ئو
ئۇو (û)	Rounded with minimum aperture	ئۇو	وو	وو	ئۇو
ئى (ö)	Diphthong formed by the gradual opening of lips	-	وي	وي	ئوى
ئه (e)	mid unrounded short	ئه	ە	ە	ئه
ئى (i)	vowel aperture average	ئى	-	-	ئى
ئى (ê)	close front long	ئىد	ې	ى	ئى

**Fig. 13.12** Kurdish Sorani vowels, where the initials are as follows: I. for initial form, M. for medium, F. for finale, and Is. for isolated

On the contrary, other languages have kept the Arabic alphabet while substituting the short vowels in Arabic letters. This is the case of Kurdish Sorani, which uses a group of two and three letters to record eight vowels (see Fig. 13.12).

**Graphism Multiplication:** In the Arabic writing, the letters are connected to each other. This practice leads to four different morphologies of the same letter, depending on its location in the word: initial, medial, final, or isolated, except for two letters which have only two forms. Figure 13.13 shows a few examples.

**Ligatures:** The Arabic script and its derivatives obey to the same ligation rules. They have three types of ligatures: contextual ligatures, language bindings, and

**Fig. 13.13** Graphism multiplication, where the initials are as follows: I. for initial form, M. for medium, F. for finale, and Is. for isolated

Arabic letters	I.	M.	F.	Is.
ب	ب	ب	ب	ب
ص	ص	ص	ص	ص
ع	ع	ع	ع	ع
ز	ز	ز	ز/ز	ز

**Fig. 13.14** Examples of contextual and aesthetic ligatures

Contextual ligatures				
بـعـثـ	→	بـعـثـ	بـعـ	بـ
بـعـثـتـ	→	بـعـثـتـ	بـعـ	بـ
بـعـثـتـاـ	→	بـعـثـتـاـ	بـعـ	بـ
Aesthetic ligatures				
At the beginning of words	J + ح	ج	→	خ
	ش + ح	ش	→	خ
	ج + ن	ج	→	خ
At the end of words	ر + ي	ر	→	ي
	ي + ن	ن	→	ي
	ي + ت	ت	→	ي

aesthetic ligatures. A contextual ligature is a string taking special shapes according to their position in the word, following strict grammatical rules and linked solely to writing. The language bindings are essential for writing a given language and obeying grammatical rules, which make them close to digraphs. The aesthetic ligatures are optional graphics that exist for aesthetic reasons, readability, and/or tradition. They can be replaced by their components without changing the grammatical validity or the meaning of the text. Figure 13.14 gives some examples of contextual and aesthetic ligatures.

**Arabic Writing Styles:** The development of Arabic calligraphy led to the creation of several decorative styles that were designed to accommodate special needs. The most outstanding of these styles are the following: Nastaliq, Koufi, Thuluthi, Diwani, Rouqi, and Naskh. An example of each one of these styles is given in Fig. 13.15.

Nastaliq	أَبْجَدْ هَوَزْ حُطِّي كَلْمَنْ سَعْفَصْ قَرَشَتْ شَخَذْ ضَطَغْ
Koufi	أَبْجَدْ هَوَزْ حُطِّي كَلْمَنْ سَعْفَصْ قَرَشَتْ شَخَذْ ضَطَغْ
Thuluthi	أَبْجَدْ هَوَزْ حُطِّي كَلْمَنْ سَعْفَصْ قَرَشَتْ شَخَذْ ضَطَغْ
Diwani	أَبْجَدْ هَوَزْ حُطِّي كَلْمَنْ سَعْفَصْ قَرَشَتْ شَخَذْ ضَطَغْ
Rouq'i	أَبْجَدْ هَوَزْ حُطِّي كَلْمَنْ سَعْفَصْ قَرَشَتْ شَخَذْ ضَطَغْ
Naskh	أَبْجَدْ هَوَزْ حُطِّي كَلْمَنْ سَعْفَصْ قَرَشَتْ شَخَذْ ضَطَغْ

**Fig. 13.15** Examples of decorative styles

Usually, Naskh and Nastaliq are mostly followed by Arabic script-based languages. Nastaliq is mostly followed for Urdu, Punjabi, Sindhi, etc., whereas Naskh is mostly followed for Arabic, Persian, etc. (see Fig. 13.16). The difference between the two styles Naskh and Nastaliq is very significant for Urdu. Naskh style is closer to the traditional Arabic style except for some final letters. The aesthetic style is more pronounced in Nastaliq with very oblique ligatures. It also shows more pronounced variations of the letters according to their position in the word.

**Ghost Character Theory:** Considering the various Arabic scripts, an effort has been made to reassemble their shapes in order to facilitate their computer encoding. The National Language Authority, under Dr. Attash Durrani's supervision, led in early 2000 to the proposition of a standard for Urdu keyboard based on the “ghost character theory.”

This theory learns that all Arabic script-based languages can be written with only 44 ghost characters. Ghost characters consist of 22 basic shapes called Kashti, shown in Fig. 13.17, and 22 dots (diacritical marks). However, each base ligature has its own phonemes and meanings in every language, with the same or different number of diacritical marks. Thus, the basic shapes (glyphs) are the same for all Arabic script-based languages with only difference in font, i.e., Naksh, Nastaliq, and diacritical marks followed by every language [32]. The “bey” contains 32 shapes shown in Fig. 13.18.

a	b
<p>قائد اعظم کے ارشادات</p> <p>قوم کے تین سیون:</p> <p>تعمیر قوم کے بڑے بڑے ستون کیا ہیں؟ آئیے میں آپ کو بتاتا ہوں۔ کسی قوم کو ایک مملکت کا حکمران اور ملک چلانے کے قابل ضرورت ہے۔</p>	<p>قائد اعظم کے ارشادات</p> <p> القوم کے تین سیون:</p> <p>تعمیر قوم کے بڑے بڑے ستون کیا ہیں؟ آئیے میں آپ کو بتاتا ہوں۔ کسی قوم کو ایک مملکت کا حکمران اور ملک چلانے کے قابل ضرورت ہے۔</p>

Fig. 13.16 Urdu styles: (a) Naskh. (b) Nastaliq

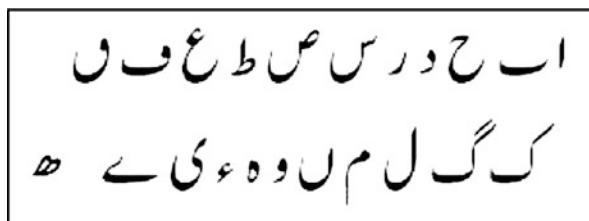


Fig. 13.17 Ghost characters used in Arabic script based language [55]

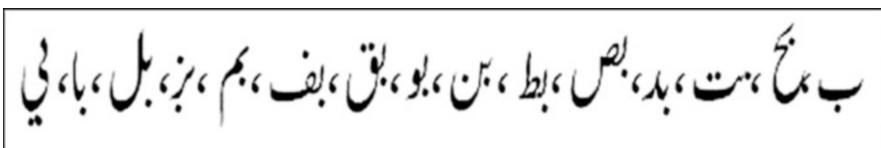


Fig. 13.18 Complexity of Nastaliq script i.e., different shapes with associated characters [55]

## Hebrew Writing System

**The Hebrew Alphabet:** Square script, block script, or, more historically, the Assyrian script is used in the writing of the Hebrew language, as well as other Jewish languages, most notably Yiddish, Ladino, and Judeo-Arabic. There are two types of script forms: the ancient Hebrew known as the Paleo-Hebrew script and the current, square-shaped, stylized form of the Aramaic script. The alphabet uses 27 characters to represent 22 consonants. Five consonants change shape once they are placed at the end. The five final shape letters are considered as additional, separate letters of the alphabet.

Kaf	Yod	Tet	Het	Zayin	Vav	He	Dalet	Gimel	Bet	Alef
כ	י	ת	ח	ז	ו	ה	ד	ג	ב	א
Tav	Shin	Resh	Qof	Tsadi	Pe	Ayin	Samekh	Nun	Mem	Lamed
ת	ש	ר	ק	כ	פ	ע	ס	נ	מ	ל

Fig. 13.19 Hebrew alphabet (From Wikipedia)



Fig. 13.20 Typeface examples – from left to right, top to bottom: NARKISS, KOREN, AHARONI, CHAYIM, SCHOCKEN, and GILL (Reproduced (and rearranged) from [66])

Aramaic square script	אֲבָנְדָהוֹזְחַטִּיבְלַמְסֻעְפְּצָקְרָשָׁת
DSS Hebrew	אֲפָגָהְוֹזְחַטִּיבְלַמְסֻעְפְּצָקְרָשָׁת
Paleo Hebrew	אֲפָגָהְוֹזְחַטִּיבְלַמְסֻעְפְּצָקְרָשָׁת
Moabite Stone	אֲפָגָהְוֹזְחַטִּיבְלַמְסֻעְפְּצָקְרָשָׁת
Ancient Hebrew	אֲפָגָהְוֹזְחַטִּיבְלַמְסֻעְפְּצָקְרָשָׁת
Early Semitic	אֲפָגָהְוֹזְחַטִּיבְלַמְסֻעְפְּצָקְרָשָׁת

Fig. 13.21 Different writing styles for Hebrew

Unlike Arabic, the Hebrew letters do not take different shapes depending on the surrounding letters (see Fig. 13.19). Hebrew does not have capital letters. Several typefaces are however used for printed text (see Fig. 13.20). Figure 13.21 shows different writing styles for Hebrew.

**Similarities to Arabic:** As in Arabic, diacritics accompany the characters, either above, below, or inside, in order to represent vowel sounds (see Fig. 13.22).

**Differences with Arabic:** Hebrew is not a cursive script, which means that the letters are not connected (see Fig. 13.23).

כָּל בְּנֵי הָאָדָם נוֹלְדוּ בְּנֵי חֹרִין וְשׁוֹנִים בַּעֲרָקֶם  
 וּבַזְכִּירָתֵיכֶם. כְּלָמָד חֻנְנוּ בַּתְּבוּנָה וּבַמְצָפָון, לְפִיכְךָ  
 חֹבֶה עַלְيָהָם לְנָהָוג אִישׁ בְּרַעַתְּבָה בְּרוּם שֶׁל אָחָתָה.

**Fig. 13.22** Diacritics in Hebrew (From: <http://www.omniglot.com/writing/hebrew.htm>)



**Fig. 13.23** Hebrew cursive text example (From <http://www.omniglot.com/writing/hebrew.htm>)

## Writing Recognition Systems

The following lists the main steps that are needed for designing recognition systems. Within this framework, new methods that have been proposed for the aforementioned scripts as well as existing traditional methods will be explained, with a focus on how they have been adapted.

### Preprocessing

This first phase for getting the word shape consists of several steps that are described in the sequel.

**Dots and Diacritical Marks Removal:** It is customary to remove strokes like diacritical dots or marks in the preprocessing phase, so that overall appearance of the writing does not change. They then introduce them again in the post-processing phase. This is however not a robust strategy/technique, since they cannot be equally detected because of their structural properties. Furthermore, in Urdu and Farsi, some retroflex marks correspond to small characters which need to be recognized.

Sari et al. [59] split the writing area into three bands and use upper and lower bands for locating and removing the diacritical marks. This idea is echoed by Miled et al. [50] who define a set of symbols. Zahour et al. [68] divide the image into vertical bands of uniform width, in which they proceed to classify all the connected

components. The method reuses the size where small blocks are generally classified as diacritics. Razzak et al. [56] remove the diacritical marks based on their size and position, to allow for efficient feature extraction. After that, the combination of recognized based shapes plus diacritical marks results in an efficient multi-language character recognition system. Finally, in Menasri et al. [48], several steps are followed for the diacritic mark extraction. First, a filtering step is used to exclude the “alif” and the single letters. This initial filtering is used to detect the base band which is exploited to locate the diacritics above or below. Second, another filtering of diacritics is made in these bands, referring to the position and size.

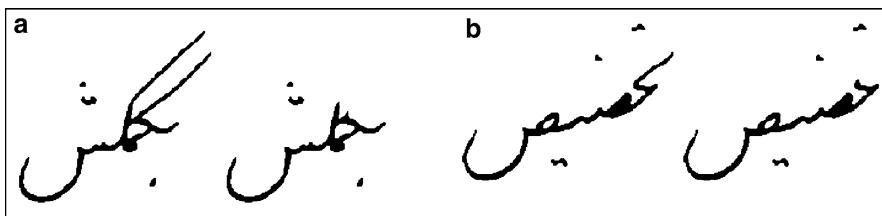
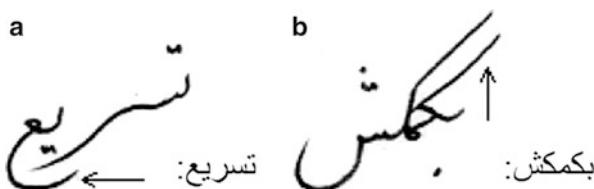
**Baseline Extraction:** The baseline is virtual and a foundation line on which characters are combined to form the ligatures. It provides its orientation and can be used as a guide for segmentation. A survey provided in [12] summarizes the different comprehensive approaches using the horizontal projection and detection of peaks, skeleton analysis by using linear regression as in [54] for Arabic and in [55] for Urdu, word contour representation by local minima and linear regression as in [37], or those based on the principle of components analysis [22].

In Urdu, Farsi, and some Arabic styles, specially Nastaliq writing style, the ascenders and descenders cause incorrect detection of the baseline because of their oblique orientation and long tail (see Fig. 13.24). This is why Safabakhsh and Adibi [58] proposed to eliminate certain ascenders, such as those of “Kaf” and “Gaf” (see Fig. 13.24b), and descenders of letters like “Jim,” “Che,” “He,” “Khe,” “Ayn,” and “Ghayn” (see Fig. 13.24a) to help the segmentation and the baseline extraction. For the ascenders, some features such as the overlapping with other strokes and a large change in the stroke width near its head are considered. Figure 13.25 shows correct elimination (a) and incorrect elimination (b). For the descenders, the elimination is based on the multiplication of black runs on the same line, allowing to eliminate the curve part of final characters (see Fig. 13.26). Razzak et al. [57] estimated the baseline locally on base stroke with additional knowledge of previous words for the handwritten Arabic script. As the handwritten Arabic script is more complex and may not exactly on the same baseline, the primary task was to extract global baseline that eventually helps for local baseline extraction. For more understanding, the reader is referred to Figs. 13.24–13.26 where different approaches are categorized according to the techniques used.

## Word Segmentation

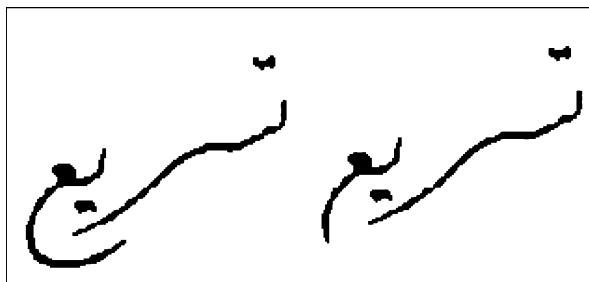
The purpose of segmentation is to subdivide the shape such that each pair of connected characters will be split into two. In the current state of the art, there are several different approaches. It goes without saying that for Arabic handwriting, methods based on the vertical histogram are not appropriate. Similarly, methods using the vertical thickness of the stroke are not suitable for free script. Three main

**Fig. 13.24** Problems arise from descenders and ascenders: (a) “Ayn” will be considered before “Ye”; (b) “Kaf” will be considered before “Be” [58]



**Fig. 13.25** Operation of ascender elimination algorithm: (a) correct operation; (b) incorrect operation [58]

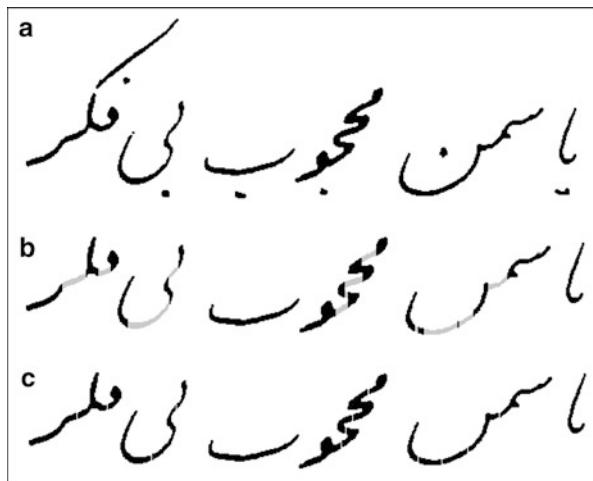
**Fig. 13.26** Operation of descender elimination algorithm [58]



methods are suitable for the Arabic-based languages. They are based on regularities and singularities, local minima of the upper contour, and the right-to-left order.

**Segmentation Based on Regularities and Singularities.** This is proposed for Arabic handwritten words in [51]. Safabakhsh and Adibi [58] used a similar approach for Urdu. Singularities or islands correspond to the vertical parts of the image and their connections, while regularities correspond to horizontal ligatures, loops, and small curves along the baseline, obtained by a subtraction between singularities and the image. Similarities are obtained by an opening operation with vertical element whose height is larger than the pen width. The connections are reached by a closing operation with a horizontal structural element having a small

**Fig. 13.27** (a) The words.  
 (b) Singularities and regularities specified by *black* and *gray colors*, respectively.  
 (c) Resulting segmentation [58]



width. Small loops are filled in a preprocessing phase to avoid their segmentation. Figure 13.27 shows an example of this type of segmentation.

**Segmentation Using Local Minima of the Word Upper Contour.** This is proposed first by Olivier et al. [52] for Arabic words. The local minima of the upper contour are first selected. Then, if these points are in favorable situation, like being at the frontiers of loops, they are considered as segmentation points. This method has been modified by Safabakhsh and Adibi [58] to extend it to Nastaliq handwritten words. In Nastaliq style, when character “Re” is connected to another character before it, it is written without any upper contour minima between it and the previous character, which cannot work here. To face this problem, the algorithm starts by detecting the “Re” and the overlapping zones. Then, similar procedures are followed to detect local minima and segmentation points. Figures 13.28 and Fig. 13.29 give examples of such a segmentation.

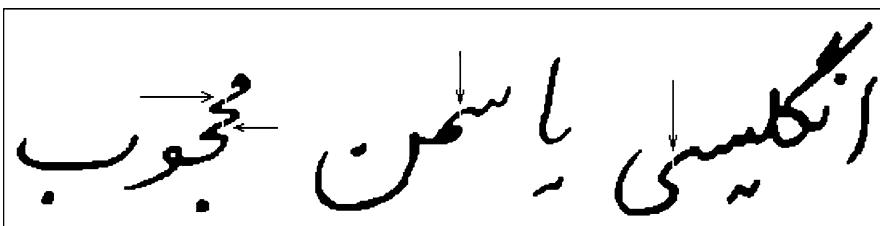
**Segmentation Finding Right-to-Left Order.** This is proposed first by Chen et al. [24] and later simplified by Safabakhsh and Adibi [58]. The idea is that once ascender and descender are removed and the order of sub-words found, the order of segments is obtained by considering the right-most segment as the first one in the sub-word. Then, one traverses the outer contour and considers the order in which segments are visited.

## Isolated Character Recognition

Some researches closely examine character recognition. Alansour and Alzoubady [5] proposed a neocognitron to classify handwritten characters. The input of this



**Fig. 13.28** “Re” detection algorithm for three words [58]



**Fig. 13.29** Overlap detection algorithm for three words [58]

particular neural network (NN) is composed of structural features. The system assumes a context of a handwritten Arabic document recognition software that should be able to segment words directly into letters. Asiri and Khorsheed [14] proposed to use two different NN architectures for handwritten Arabic character recognition. For all NNs, the inputs correspond to a certain number of Haar wavelet transform coefficients. Cowell and Hussain [28] worked on isolated Arabic printed characters. A character image is normalized and a quick matching method is used. In [46], an automatic identification of hand-printed Hebrew characters is described. The squared shape of the letters led to the use of primitive-type straight via the Hough transform. These primitives are augmented by indices such as corners and end points. Then, a NN is used for recognition. Fakir et al. [35] proposed to use the Hough transform for the recognition of printed Arabic characters.

## Word Recognition

While doing Arabic script recognition research, most of the proposed recognition systems can be extended to Latin, Chinese, and Japanese without considering the particular difficulties associated with different scripts. This means that the recognition system provides genericity, i.e., application independency. Today, advances produced for Arabic script, for example, will serve the derivatives languages with which they share many similarities. Thus, the usual recognition methods for recognizing and assigning them the most characteristic works of the literature will be shown concerning Arabic script-based languages recognition. Belaïd and

Choisy [17] gave a comprehensive review of the Arabic recognition approaches. The approaches are classified into four, according to the human perception of writing. They are:

- Global-based vision classifiers
- Semi-global-based vision classifiers
- Local-based vision classifiers
- Hybrid-level classifiers

**Global-Based Vision Classifiers:** In this holistic approach, the word is regarded as a whole, allowing correlations with the totality of the pattern. This common approach avoids the heavy task of letter location and word segmentation. The key idea involves detecting a set of shape primitives in the analyzed word and arranging them properly in the word space [6, 36, 44]. Some systems failed to evaluate the gap between two consecutive word parts (PAW or part of Arabic word) to decide the word limits. In [64], the presence of the “Alef” was reported as the first letter of many Arabic words as the most relevant feature.

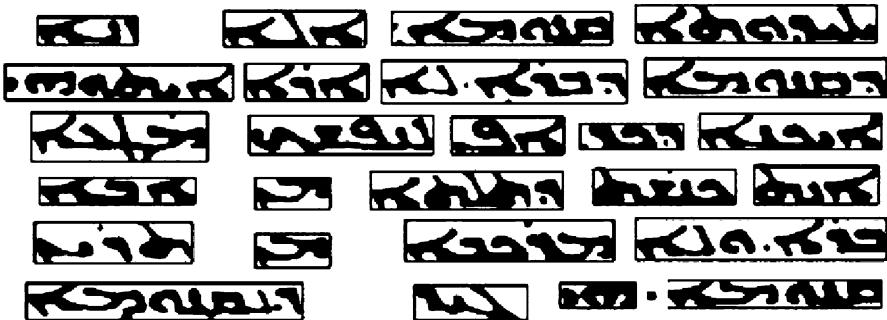
Feature selection in this word vision is essentially of global nature. They have been addressed by researchers for both recognition and writer identification. Among these features can be mentioned multichannel Gabor filtering [13]; 2D DWT combined with morphological variations of writing [38]; combination of gradient, curvature, density, horizontal and vertical run lengths, stroke, and concavity features [15]; hybrid spectral-statistical measures (SSMs) of texture [7]; curvature measurements from minimum perimeter polygon [3]; fractal dimensions by using the “box-counting” method [23]; edge-based directional probability distributions and moment invariants [11]; white and black run length; and edge-hinge features [30].

**Semi-global Based Vision Classifiers:** More specifically, the nature of Arabic writing allows us to describe the language in fewer natural level, i.e., the PAW level. Indeed, Arabic words are built by a concatenation of several independent written parts that give another natural segmentation level. This natural segmentation allows us to refine the analysis by reducing the number of base vocabulary. It explains some approaches that are based their work on this level. By reducing the base vocabulary, it allows the possibility of extending the dictionary [18, 20].

The global features shown previously can be reported in a similar manner here for the PAW images.

**Local-Based Vision Classifiers:** The objective of this word vision is to focus on the interpretation of letters and or smaller entities for their interpretation. In other words, the process is to gather, bind and confront these entities to identify the word. Such an analysis level leads to the Sayre dilemma. This problem is usually eluded by the use of implicit or explicit segmentation methods.

Very recently, El Abed and Margner have organized a competition at the ICDAR conferences [33]. This competition was made mainly under the purview of feature selection, where the major extraction methods were evaluated. They are sliding



**Fig. 13.30** Syriac writing: interword gaps are larger than intra-word gaps [27]

window with pixel features, skeleton direction-based features, and sliding window with local features.

In addition to these conventional features, the literature abounds with primitive extraction techniques such as analytical approaches with or without consideration of parts of words and with or without segmentation into letters. Examples are the work of Abandah et al. [2] who used mRMR to select four sets of features for four classifiers, each specialized in recognizing letters of the four letter forms. Fahmy and Al Ali proposed in [4, 34] a system with structural features. Razzak et al. [56] presented a hybrid approach using HMM and fuzzy logic for Arabic-script based languages written in both Nastaliq and Naskh styles. Fuzzy logic is used for feature extraction, clustering, and post-processing. Clocksin and Fernando proposed an analytic system for Syriac manuscripts [27]. In contrast to Arabic writing, the word segmentation simply happens as the intra-word gaps seem to be clearly smaller than interword gaps. The grammatical grammar functions almost appear as word prefixes or suffixes instead of separated words, so that a huge dictionary is inevitable to provide global word approach (see Fig. 13.30).

Miled et al. [50] proposed an analytical approach based on HMMs for the recognition of Tunisian town names. They integrate the notion of PAW in their system. They grouped letters with the same body but different diacritics to “solve” the problem of diacritic detection and classification.

**Hybrid-Level Classifiers:** By combining different strategies, it is possible to come near to the principle of reading – the analysis must be global for a good synthesis of information, while being based on local information that are suitable to make this information emerge [26, 62]. Such an approach combination allows us to better see how a person reads, which is to first analyze global word shapes, and search for local information only to discriminate ambiguous cases. As local-based approaches gather local information up to words, they could be hybrid ones. An important difference that exists between the two approaches is that hybrid approaches attempt a multilevel analysis of the writing, while local-based approaches gather local information.

Hybrid approaches aim at providing different levels of features and interpretation [63]. Accordingly, Maddouri et al. proposed [62] a combination of global and local models based on a transparent NN (TNN).

**Contribution to Natural Language Processing:** Usually, NLP is kept as the last step of the recognition system, to be able to correct the errors by providing some considerations stemming from the writing language [25]. Arabic is characterized by a complex morphological structure. It is a highly inflected language where prefixes and suffixes are appended to the stem of a word to indicate tense, case, gender, number, etc. For that reason, it seems that words are not the best lexical units in this case and, perhaps, sub-word units would be a better choice. In the literature, few researches are done in that direction. As mentioned in Cheriet [25], the question of incorporating NLP in a recognition system is a non-resolved problem. Till now, it is not definitely clear where the incorporation of NLP is more profitable for a writing recognition system. Ben Cheikh's research [19] incorporates the morphology analysis within models which are conceived to collaborate and respectively learn and recognize roots, schemes, and conjugation elements of words.

Table 13.3 categorizes other systems according to the approach, the classifier, the dataset, and the writing style.

---

## Success Clues of the Topic

The successful achievement in any research topic is measured by the number of commercial systems, of distributed databases, and by the number of competitions held in international conferences. This section outlines some of these tell-tale signs of success of research in the Middle Eastern character recognition.

## Datasets

Datasets of any printed or handwritten language are the most important part in order to measure accuracy and it gives equally for recognition system design. Table 13.4 lists the existing publicly available datasets that have been used over several years.

## Academic Systems

Since 2005, a series of competitions took place at the important conferences dedicated to document analysis like ICDAR and ICFHR. During these competitions, datasets are fixed and systems are experimented on them. Table 13.5 reports the results of four competitions of Arabic handwritten word recognition systems, from 2005 to 2011. The dataset is limited to IFN/INIT with a selection of this dataset (sets d, e,  $f_a$ , f, s). All participating systems were based mainly on hidden Markov models (HMM).

**Table 13.3** Comparison between techniques

System	Approach	Classifier	Dataset	Writing style	R. rate
Biologically inspired fuzzy-based expert system	Simultaneous segmentation and recognition	Fuzzy logic	Full dataset	Nastaliq, Naskh	87.3 %
Hybrid approach HMM and fuzzy logic [56]	Ligature-based approach	HMM and fuzzy logic	14,150 words	Nastaliq, Naskh	87.6 %
Arabic character recognition		Genetic algorithm and visual encoding	200 words	Naskh	97 %
Online Urdu character recognition [43]	Ligature-based approach	Back propagation neural network	240 ligatures	Nastaliq	93 %
Online Arabic handwritten recognition with templates [65]		Template-based matching	Full dataset	Naskh	68.2 %
Bio-inspired handwritten Farsi [21]		KNN ANN SVM	Farsi digits (MNIST)	Naskh	81.3 % (KNN) 94.65 % (ANN) 98.75 % (SVM)
Arabic handwritten using matching algorithm [53]	Ligature-based approach	Matching algorithm and decision tree	Arabic alphabets	Naskh	98.8 %
Recognition based segmentation of Arabic [29]	Simultaneous segmentation and recognition	HMM	Arabic	Naskh	88.8 %
Online Arabic characters by Gibbs modelling [49]		Gibbs modelling of class conditional densities	Arabic characters	Naskh	84.85 % (direct Bayes) 90.19 % (indirect Bayes)
Arabic character recognition using HMM [9]		HMM	Arabic full data set	Naskh	78.25 %

**Table 13.4** Datasets for Eastern scripts

Name	Script	Component	Nature	Content
IFN/ENIT	Arabic	Handwritten words	Towns	411 writers, 26,400 samples
APTI	Arabic	Multi-font, multi-size, and multi-style text	Printed text	45,313,600 single-word images totaling to more than 250 million characters
AHDB	Arabic	Arabic words and texts	Form words	100 writers
ERIM	Arabic	Machine-printed documents	Arabic books and magazines	750 pages, 1,000,000 characters
CEDARABIC	Arabic	Handwritten documents	Document pages	10 writers, 20,000 words
ADAB	Arabic	Online	984 Tunisian town names	15,158 pen traces, 130 different writers
HODA	Farsi	Handwritten digits	Form digits	102,352 digits
CENPARMI	Farsi	Numerical strings, digits, letters, legal amounts and dates	Binary forms of handwritten digits and characters	11,000 training and 5,000 test samples while characters set includes 7,140 training and 3,400 test samples

## Commercial Systems

There are a lot of commercial applications for Middle Eastern scripts, like ABBYY Finereader, Readiris, HOOCR (only for Hebrew), Sakhr, Omnipage, and VERUS, whereas for online MyScript, Sakhr is publicly available. Sakhr provides 99.8 % accuracy for high-quality documents and 96 % accuracy for low-quality documents. It has supports for Arabic, Farsi, Pashto, Jawi, and Urdu, and it also supports bilingual documents: Arabic/French, Arabic/English, and Farsi/English. Moreover, it can handle image and text recognition captured by mobile devices. It can also auto-detect translation language.

US government evaluators assess Sakhr as the best available Arabic OCR. Sakhr online intelligent character recognition (ICR) recognizes Arabic cursive handwritten input through a normal pen with 85 % word accuracy.

Two groups with three systems, IPSAR, and UPV-BHMM (UPV-PRHLT-REC1 and UPV-PRHLT-REC2), were presented at ICDAR11 and tested on the APTI dataset. IPSAR is based on the HMM and follows stages: extracting a set of features from the input images, clustering the feature set according to a predefined codebook, and, finally, recognizing the characters. Very interestingly, IPSAR does not require segmentation. UPV-BHMM presented two systems UPV-PRHLT-REC1 and UPV-PRHLT-REC2. UPV-BHMM systems are based on window sliding of adequate width to capture image context at each horizontal position. The IPSAR

**Table 13.5** Results of the three best systems at ICDAR 2007

Name	Classifier	Features	Set d	Set e	Set $f_a$	Set f	Set s
Results of the four best systems at ICDAR 2011							
JU-OCR [1]	Random forest	Explicit grapheme segmentation	75.49	63.75	64.96	63.86	49.75
CENPARMI-OCR [61]	SVM	Gradient features, Gabor features, Fourier features	99.90	99.91	40.00	40.00	35.52
RWTH-OCR [31]	HMM	Appearance-based image slice features	99.67	98.61	92.35	92.20	84.55
REGIM [42]	PSO-HMM	Karhunen-Loeve transform [33]	94.12	86.62	80.60	79.03	68.44
Results of the three best systems at ICFHR 2010							
UPV-PRHLT [39]	HMM	Sliding window	99.38	98.03	93.46	92.20	84.62
CUBS-AMA	HMM		89.97	80.80	81.75	80.32	67.90
RWTH-OCR [31]	HMM	Simple appearance-based image slice features	99.66	98.84	92.35	90.94	80.29
Results of the three best systems at ICDAR 2009							
MDLSTM [41]	NN	Raw input	99.94	99.44	94.68	93.37	81.06
Ai2A [48]	GMHMM	Geometric features with sliding windows	97.02	91.68	90.66	89.42	76.66
RWTH-OCR [31]	HMM	Simple appearance-based image slice features	99.79	98.29	87.17	85.69	72.54
Results of the three best systems at ICDAR 2007							
Siemens [60]	HMM	Feature vector sequence	94.58	87.77	88.41	87.22	73.94
MIE	LDC [45]	Word length estimation	93.63	86.67	84.38	83.34	68.40
UOB-ENST	HMM [10]	Features with respect of the baseline	92.38	83.92	83.39	81.93	69.93

system provided good results for the “Traditional Arabic” and “Diwani Letter” fonts in font sizes 10, 12, and 24. However, the system UPV-PRHLTREC1 was the winner of this first competition.

During this ICDAR competition, three groups, VisionObjects, AUC-HMM, and FCI-CU-HMM, have submitted their systems. These systems were tested over the ADAB dataset. There were two levels of evaluation. The first level of evaluation

**Table 13.6** Examples of commercial OCRs

System	Script	Comments	Performance
Sakhr OCR	Arabic, Farsi, Jawi, Dari, Pashto, Urdu	Arabic, Farsi, English, French	96–99 %
VERUS OCR NovoDynamics	Arabic, Farsi, Persian, Dari, Pashto		
Readiris	Arabic, Farsi, and Hebrew	<ul style="list-style-type: none"> <li>– Pro features: standard scanning support and standard recognition features</li> <li>– Corporate features: volume scanning support and advanced recognition features</li> </ul>	
Kirtas KABIS employs SAKHR engine for Arabic	Arabic (Naskh, Kofi), Farsi, Jawi, Pashto, and Urdu	– SureTurn robotic arm uses vacuum system to gently pick up and turn one page at a time	
HOCR	Hebrew	Support for all Hebrew, English, and Western European languages	
ABBYY FineReader	Arabic, Hebrew	– Dictionary for some languages – free trial is available	99 %
Ligature-OCR	Hebrew	Omnifont reading based on stochastic algorithms and neural networks	11,000 old Hebrew books
Freeocr	Hebrew	Books and reports, selected zones	
OCR program	Yiddish	Omnifont	Line by line

was based on the subsets 1–4, and systems shows recognition rate better than 80 % on sets 1–4, whereas the second-level evaluation was performed on set 5 and set 6. The recognition rate was limited between 60.28 and 98.97 %. The system of Vision Objects was the winner of this competition (Table 13.6).

## Conclusion

Middle Eastern languages have very typical scripts, especially Syriac and Arabic. Therefore, special attention must be given to the design of the complete recognition system. Based on the thorough analysis of the different techniques, the following observations can be made:

- **Feature representation:** Low-level features are language independent. Once extracted (similarly for all the scripts), the training process can arrange their

proximity to the language studied. In contrast, high-level features are language dependent and need to develop specific extraction methods to retrieve all information. Obviously, a combination of these two kinds of features should perform better where each feature level is used to complement the drawback of the other.

- **PAWs:** Contrarily to Latin script, the basic entity is not the word. Global approaches should be based on PAW. Analytical ones gain by integrating this information level. A first effect reduces the vocabulary complexity by gathering the information on an intermediate level.
- **Segmentation, in words or in letters?:** Since Arabic writing is often described as more complex than Latin, it seems obvious that a letter segmentation cannot be effective.
- **Appropriateness of approaches:** The most suitable for these scripts. Hybrid ones seem very promising. They efficiently combine different perceptive levels, allowing discrimination of words without a complete description. In comparison with global approaches, the addition of local information allows it to extend the vocabulary with less confusion. Compared to local approaches, hybrid ones can avoid the full-segmentation problems and are less disturbed by information loss.

---

## Cross-References

- ▶ [A Brief History of Documents and Writing Systems](#)
- ▶ [Continuous Handwritten Script Recognition](#)
- ▶ [Datasets and Annotations for Document Analysis and Recognition](#)
- ▶ [Handprinted Character and Word Recognition](#)
- ▶ [Machine-Printed Character Recognition](#)
- ▶ [Text Segmentation for Document Recognition](#)

---

## References

1. Abandah G, Jamour F (2010) Recognizing handwritten Arabic script through efficient skeleton-based grapheme segmentation algorithm. In: Proceedings of the 10th international conference on intelligent systems design and applications (ISDA), Cairo, pp 977–982
2. Abandah G, Younis K, Khedher M (2008) Handwritten Arabic character recognition using multiple classifiers based on letter form. In: Proceedings of the 5th IASTED international conference on signal processing, pattern recognition, and applications (SPPRA), Innsbruck, pp 128–133
3. Abdi MN, Khemakhem M, Ben-Abdallah H (2010) Off-line text-independent Arabic writer identification using contour-based features. *Int J Signal Image Process* 1:4–11
4. Abuhaiba ISI, Holt MJJ, Datta S (1998) Recognition of off-line cursive handwriting. *Comput Vis Image Underst* 71:19–38
5. Alansour AJ, Alzoubady LM (2006) Arabic handwritten character recognized by neocognitron artificial neural network. *Univ Sharjah J Pure Appl Sci* 3(2):1–17
6. Al-Badr B, Haralick RM (1998) A segmentation-free approach to text recognition with application to Arabic text. *Int J Doc Anal Recognit (IJDAR)* 1:147–166

7. Al-Dmour A, Abu Zitar R (2007) Arabic writer identification based on hybrid spectral-statistical measures. *J Exp Theor Artif Intell* 19:307–332
8. Al Hamad HA, Abu Zitar R (2010) Development of an efficient neural-based segmentation technique for Arabic handwriting recognition. *Pattern Recognition* 43(8):2773–2798
9. Al-Habian G, Assaleh K (2007) Online Arabic handwriting recognition using continuous Gaussian mixture HMMS. In: International conference on intelligent and advanced systems (ICIAS), Kuala Lumpur, pp 1183–1186
10. Al-Hajj R, Likforman-Sulem L, Mokbel C (2005) Arabic handwriting recognition using baseline dependant features and hidden Markov modeling. In: 8th international conference on document analysis and recognition (ICDAR), Seoul, vol 2, pp 893–897
11. Al-Ma-adeed S, Mohammed E, Al Kassis D, Al-Muslih F (2008) Writer identification using edge-based directional probability distribution features for Arabic words. In: IEEE/ACS international conference on computer systems and applications, Doha, pp 582–590
12. Al-Shatnawi AM, Omar K (2008) Methods of Arabic language baseline detection, the state of art. *ARISER* 4(4):185–193
13. Al-Zoubeidy LM, Al-Najar HF (2005) Arabic writer identification for handwriting images. In: International Arab conference on information technology, Amman, pp 111–117
14. Asiri A, Khorsheed MS (2005) Automatic processing of handwritten Arabic forms using neural networks. *Trans Eng Comput Technol* 7:147–151
15. Awaida SM, Mahmoud SA (2010) Writer identification of Arabic handwritten digits. In: IWFAHR10, Istanbul, pp 1–6
16. Bar-Yosef I, Beckman I, Kedem K, Dinstein I (2007) Binarization, character extraction, and writer identification of historical Hebrew calligraphy documents. *Int J Doc Anal Recognit* 9(2–4):89–99
17. Belaïd A, Choisy Ch (2006) Human reading based strategies for off-line Arabic word recognition. In: Summit on Arabic and Chinese handwriting (SACH'06), University of Maryland, College Park, 27–28 Sept 2006
18. Ben Amara N, Belaid A (1996) Printed PAW recognition based on planar hidden Markov models. In: Proceedings of the 13th international conference on pattern recognition, Vienna, 25–29 Aug 1996, vol 2, pp 220–224
19. Ben Cheikh I, Kacem A, Belaïd A (2010) A neural-linguistic approach for the recognition of a wide Arabic word lexicon. In: Document recognition and retrieval XVII, San Jose, pp 1–10
20. Bippus R (1997) 1-dimensional and pseudo 2-dimensional HMMs for the recognition of German literal amounts. In: ICDAR'97, Ulm, Aug 1997, vol 2, pp 487–490
21. Borji A, Hamidi M, Mahmoudi F (2008) Robust handwritten character recognition with feature inspired by visual ventral stream. *Neural Process Lett* 28:97–111
22. Burrow P (2004) Arabic handwriting recognition. M.Sc. thesis, University of Edinburgh, Edinburgh
23. Chaabouni A, Boubaker H, Kherallah M, Alimi AM, El Abed H (2010) Fractal and multi-fractal for Arabic offline writer identification. In: ICPR-10, Istanbul, pp 1051–4651
24. Chen MY, Kundu A, Srihari SN (1995) Variable duration hidden Markov model and morphological segmentation for handwritten word recognition. *IEEE Trans Image Process* 4(12):1675–1688
25. Cheriet M (2006) Visual recognition of Arabic handwriting: challenges and new directions. In: SACH 06, College Park, Sept 2006, pp 129–136
26. Choisy Ch, Belaïd A (2003) Coupling of a local vision by Markov field and a global vision by neural network for the recognition of handwritten words. In: ICDAR'03, Edinburgh, 3–6 Aug 2003, pp 849–953
27. Clocksin WF, Fernando PPJ (2003) Towards automatic transcription of Syriac handwriting. In: Proceedings of the international conference on image analysis and processing, Mantova, pp 664–669
28. Cowell J, Hussain F (2002) A fast recognition system for isolated Arabic character recognition. In: IEEE information visualization IV2002 conference, London, July 2002, pp 650–654

29. Daifallah K, Zarka N, Jamous H (2009) Recognition-based segmentation algorithm for on-line Arabic handwriting. In: 10th international conference on document analysis and recognition, Barcelona, pp 886–890
30. Djeddi C, Souici-Meslati L, Ennaji A (2012) Writer recognition on Arabic handwritten documents. International Conference on Image and Signal Processing, Agadir, Morocco, June 2012, pp 493–501
31. Drew P, Heigold G, Ney H (2009) Confidence-based discriminative training for model adaptation in offline Arabic handwriting recognition. In: Proceedings of the international conference on document analysis and recognition (ICDAR), Barcelona, pp 596–600
32. Durani A (2009) Pakistani: lingual aspect of national integration of Pakistan. [www.nlauit.gov.pk](http://www.nlauit.gov.pk)
33. El Abed H, Margner V (2007) Comparison of different preprocessing and feature extraction methods for off line recognition of handwritten Arabic words. In: Proceedings of the 9th ICDAR 2007, Curitiba, pp 974–978
34. Fahmy MMM, Al Ali S (2001) Automatic recognition of handwritten Arabic characters using their geometrical features. *Stud Inform Control* J 10:81–98
35. Fakir M, Hassani MM, Sodeyama C (2000) On the recognition of Arabic characters using Hough transform technique. *Malays J Comput Sci* 13(2):39–47
36. Farah N, Khadir MT, Sellami M (2005) Artificial neural network fusion: application to Arabic words recognition. In: Proceedings of the European symposium on artificial neural networks (ESANN'2005), Bruges, 27–29 Apr 2005, pp 151–156
37. Farooq F, Govindaraju V, Perrone M (2005) Pre-processing methods for handwritten Arabic documents. In: Proceedings of the 2005 eighth international conference on document analysis and recognition (ICDAR), Seoul, pp 267–271
38. Gazzah S, Ben Amara NE (2007) Arabic handwriting texture analysis for writer identification using the DWT-lifting scheme. In: 9th ICDAR, Curitiba, vol.2, pp 1133–1137
39. Gimenez A, Khoury I, Juan A (2010) Windowed bernoulli mixture hmms for arabic handwritten word recognition. In: 2010 international conference on frontiers in handwriting recognition (ICFHR), Kolkata, pp 533–538
40. Ghosh D, Dube T, Shivaprasad AP (2010) Script recognition-a review. In: IEEE Trans Pattern Anal Mach Intell 32(12):2142–2161
41. Graves A, Schmidhuber J (2009) Offline handwriting recognition with multidimensional recurrent neural networks. Advances in neural information processing systems 22, NIPS'22, Vancouver, MIT Press, pp 545–552
42. Hamdani M, El Abed H, Kherallah M, Alimi AM (2009) Combining multiple HMMs using on-line and off-line features for off-line Arabic handwriting recognition. In: Proceedings of the 10th international conference on document analysis and recognition (ICDAR), Seoul, pp 201–205
43. Husain SA, Sajjad A, Anwar F (2007) Online Urdu character recognition system. In: IAPR conference on machine vision applications (MVA2007), Tokyo, pp 3–18
44. Khorsheed MS, Clocksin WF (2000) Multi-font Arabic word recognition using spectral features. In: Proceedings of the 15th international conference on pattern recognition, Barcelona, 3–7 Sept 2000, vol 4, pp 543–546
45. Kimura F, Shridhar M, Chen Z (1993) Improvements of a lexicon directed algorithm for recognition of unconstrained handwritten words. In: 2nd international conference on document analysis and recognition (ICDAR), Tsukuba, pp 18–22
46. Kushnira M, Abe K, Matsumotoa K (2003) Recognition of handprinted Hebrew characters using features selected in the Hough transform space. *Pattern Recognit* 18:103–114
47. Lorigo LM, Govindaraju V (2006) Offline Arabic handwriting recognition: a survey. *PAMI* 28(5):712–724
48. Menasri F, Vincent N, Cheriet M, Augustin E (2007) Shape-based alphabet for off-line Arabic handwriting recognition. In: Proceedings of the ninth international conference on document analysis and recognition (ICDAR), Curitiba, vol 2, pp 969–973

49. Mezghani N, Mitiche A, Cheriet M (2008) Bayes classification of online Arabic characters by Gibbs modeling of class conditional densities. *IEEE Trans Pattern Anal Mach Intell* 30(7):1121–1131
50. Miled H, Olivier C, Cheriet M, Lecoutie Y (1997) Coupling observation/letter for a Markovian modelisation applied to the recognition of Arabic handwriting. In: Proceedings of the 4th international conference on document analysis and recognition (ICDAR), Ulm, pp 580–583
51. Motawa D, Amin A, Sabourin R (1997) Segmentation of Arabic cursive script. In: Proceedings of the 4th international conference on document analysis and recognition (ICDAR), Ulm, vol 2, pp 625–628
52. Olivier C, Miled H, Romeo K, Lecourtier Y (1996) Segmentation and coding of Arabic handwritten words. In: IEEE proceedings of the 13th international conference on pattern recognition, Vienna, vol 3, pp 264–268
53. Omer MAH, Ma SL (2010) Online Arabic handwriting character recognition using matching algorithm. In: The 2nd international conference on computer and automation engineering (ICCAE), Beijing
54. Pechwitz M, Märgner V (2002) Baseline estimation for Arabic handwritten words. In: Proceedings of the eighth international workshop on frontiers in handwriting recognition (IWFHR), Niagara-on-the-Lake, p 479
55. Razzak MI, Husain SA, Sher M, Khan ZS (2009) Combining offline and online preprocessing for online Urdu character recognition. In: Proceedings of the international multiconference of engineers and computer scientists (IMECS) 2009, Hong Kong, vol I. Knowledge-based systems
56. Razzak MI, Anwar F, Husain SA, Belaid A, Sher M (2010) HMM and fuzzy logic: a hybrid approach for online Urdu script-based languages character recognition. *Knowl-Based Syst* 23:914–923
57. Razzak MI, Husain SA, Sher M (2010) Locally baseline detection for online Arabic script based languages character recognition. *Int J Phys Sci* 5(6). ISSN:1992–1950
58. Safabakhsh R, Adibi P (2005) Nastaaligh handwritten word recognition using a continuous-density variable duration HMM. *Arab J Sci Eng* 30(1 B):95–118
59. Sari T, Souici L, Sellami M (2002) Off-line handwritten Arabic character segmentation algorithm: Acsa. In: Proceedings of the eighth international workshop on frontiers in handwriting recognition (IWFHR), Niagara-on-the-Lake, p 452
60. Schambach M-P (2003) Model length adaptation of an HMM based cursive word recognition system. In: 7th international conference on document analysis and recognition (ICDAR), Edinburgh, 3–6 Aug 2003, vol 1, pp 109–113
61. Cristianini N, Shawe-Taylor J, (2000) An introduction to support vector machines and other kernel-based learning methods. Cambridge University Press, Cambridge/New York
62. Shoussi Maddouri S, Amiri H, Belaïd A, Choisy Ch (2002) Combination of local and global vision modeling for Arabic handwritten words recognition. In: 8th IWHFR, Niagara-on-the-Lake, pp 128–132
63. Souici L, Sellami M (2004) A hybrid approach for Arabic literal amounts recognition. *AJSE Arabian J Sci Eng* 29(2B):177–194
64. Srihari S, Srinivasan H, Babu P, Bhole C (2005) Handwritten Arabic word spotting using the CEDARABIC document analysis system. In: Symposium on document image understanding technology, College Park, 2–4 Nov 2005, pp 67–71
65. Sternby J, Morwing J, Andersson J, Friberg Ch (2009) On-line Arabic handwriting recognition with templates. *Pattern Recognit* 42:3278–3286
66. Tamari I (1985) Curator. New Hebrew Letter type. An exhibition catalog in Hebrew and English. University Gallery, Tel Aviv University, Tel Aviv

67. Tse E, Bigun J (2007) A Base-line character recognition for Syriac-Aramaic. In: Proceedings of the IEEE international conference on systems, man and cybernetics, Montréal, 7–10 Oct 2007, pp 1048–1055
68. Zahour A, Likforman-Sulem L, Boussellaa W, Taconet B (2007) Text line segmentation of historical Arabic documents. In: Proceedings of the ninth international conference on document analysis and recognition (ICDAR), Curitiba, vol 1, pp 138–142

## Further Reading

A good overview of Arabic handwriting recognition can be found in [47]. For a good example of a Hebrew recognition system, the reader is referred to [16].

Srirangaraj Setlur and Zhixin Shi

## Contents

Introduction.....	460
History and Importance.....	460
Evolution of the Problem.....	461
Applications.....	461
Main Difficulties.....	461
Summary of the State of the Art.....	463
Recognition of CJK Scripts.....	466
Preprocessing.....	466
Segmentation.....	466
Character Recognition.....	467
Radical-Based Algorithms.....	467
Classification.....	470
Post-processing.....	473
Recognition of Indic Scripts.....	475
Preprocessing.....	475
Segmentation.....	477
Discriminative Features.....	480
Classification.....	482
Post-processing.....	482
Conclusion.....	483
Cross-References.....	484
References.....	484
Further Reading.....	486

---

S. Setlur • Z. Shi

Department of Computer Science and Engineering, University at Buffalo, The State University of New York, Buffalo, NY, USA  
e-mail: [setlur@buffalo.edu](mailto:setlur@buffalo.edu); [zshi@buffalo.edu](mailto:zshi@buffalo.edu)

---

**Abstract**

This chapter deals with the automated recognition of Asian scripts and focuses primarily on text belonging to two script families, viz., oriental scripts such as Chinese, Japanese, and Korean (CJK) and scripts from the Indian subcontinent (Indic) such as Devanagari, Bangla, and the scripts of South India. Since these scripts have attracted the greatest interest from the document analysis community and cover most of the issues potentially encountered in the recognition of Asian scripts, application to other Asian scripts should primarily be a matter of implementation. Specific challenges encountered in the OCR of CJK and Indic scripts are due to the large number of character classes and the resultant high probability of confusions between similar character shapes for a machine reading system. This has led to a greater role being played by language models and other post-processing techniques in the development of successful OCR systems for Asian scripts.

---

**Keywords**

Character recognition • Chinese • CJK • Classifiers • Devanagari • Features • Hindi • HMM • Indic • Japanese • Korean • Language models • Neural networks • OCR • Post-processing

---

## Introduction

### History and Importance

Asian scripts are among the oldest scripts of the world and hence have played an important role in shaping early civilizations and influenced the development of other modern scripts. Scripts in general have been subject to religious and cultural influences over time, and many scripts in use today have their origins rooted in early Asian scripts (►Chap. 9 (Language, Script and Font Recognition)).

While Asian scripts such as CJK and Indic scripts support a large number of languages and dialects, the number of distinct scripts used to represent these languages is not as numerous. Asian scripts belong to three main classes [1]:

1. **East Asian scripts** are based on the East Asian or Han ideographs that were developed in China in the second millennium BCE. These are also known as the CJK scripts (for Chinese, Japanese, and Korean – the three main languages that use these scripts). The primary scripts in this category are Han (Chinese), Hiragana and Katakana (Japanese), and Hangul (Korean). Chinese ideographs are also used in modern Japanese (Kanji) and in modern Korean (Hanja).
2. **South and Southeast Asian Scripts** are based on the ancient Brahmi script, oldest instances of which have been found from third century BCE. South Asian scripts are used by at least 22 major languages and hundreds of minor languages or dialects spoken by populations in and around the Indian subcontinent. The prominent scripts and the major languages for which they are used

(in parentheses) are Devanagari (Sanskrit, Hindi, Marathi, Nepali to name a few), Bangla (Bengali, Asomiya, Manipuri), Gurumukhi (Punjabi), Gujarati (Gujarati), Oriya (Oriya, Santhali), Kannada (Kannada), Tamil (Tamil), Telugu (Telugu), and Malayalam (Malayalam).

Southeast Asian scripts are also derived from Brahmi and exhibit some idiosyncrasies that distinguish them from South Asian scripts. The scripts belonging to this category are Thai, Philippine scripts, and Indonesian scripts (Balinese and Buginese). This chapter covers OCR research on only the South Asian or Indic scripts.

3. **West Asian Scripts** are influenced by the Phoenician script – these have a strong Arabic influence and hence will be covered by the chapter on Middle Eastern Character Recognition (► [Chap. 13](#) (Middle Eastern Character Recognition)).

## Evolution of the Problem

The successful advent of the first generation OCR systems for machine-printed Latin characters in the early 1960s spurred an immediate interest in the machine reading of non-Latin scripts. Among the CJK scripts, the earliest reported attempt at printed Chinese character recognition was in 1966 [2]. Research into recognition of Indic scripts such as Devanagari began in the early 1970s [3]. Despite these early endeavors, research into recognition of Indic scripts did not pick up steam until the 1990s whereas there has been a more widespread and sustained interest in recognition of CJK scripts over the years. This is perhaps a result of the fragmentation of the market since many of the Indic scripts and languages are localized to small geographic regions.

## Applications

While traditional OCR application domains such as postal routing and bank check recognition have also been targeted in the context of Asian character recognition, digital library applications have been seen to be particularly appealing due to the rich heritage of the ancient cultures that gave birth to these languages and scripts. Hence, interesting preprocessing applications for enhancement of historical documents on media such as papyrus, palm-leaf, and stone inscriptions prior to word spotting, transcript mapping, OCR, and other downstream applications have also seen considerable research effort.

## Main Difficulties

Asian scripts, in general, share the common trait that a single written form is used to represent multiple languages that are distinct and, many a time, mutually

unintelligible. This often results in regional variations in writing of the same character. Asian scripts also have a large number of character classes. This section describes the difficulties that Asian scripts pose for machine reading.

### CJK Scripts

The CJK scripts owe their origin to the Han ideographs developed in China in the second millennium BCE. Several standard romanizations are commonly used to refer to East Asian ideographic characters. They include hànzì (Chinese), kanzi (Japanese), kanji (colloquial Japanese), hanja (Korean), and chuhán (Vietnamese). The following English terms used to describe these character sets are interchangeable: Han character, Han ideographic character, East Asian ideographic character, or CJK ideographic character.

The Unicode Standard has a Han character repertoire of 75,215 characters. However, less than 4,000 of these make up over 99 % of the characters seen in actual use today. So, for OCR applications, the effective target class space is slightly less than 4,000 characters which still is a daunting number.

### Indic Scripts

Indic scripts are all *abugidas* or writing systems where the effective unit in the script is the orthographic syllable where the core represents a vowel or a consonant with an inherent vowel. The consonant can be preceded by one or more optional consonants. The canonical structure of this orthographic syllable unit is (((C)C)C)V where C is a consonant and V is a vowel. This usually also corresponds to a phonological syllable.

*Large Character Set:* All nine Indic scripts more or less share the same relatively small basic set of vowels and consonants. However, syntactically the number of character shapes tends to be in the hundreds since consonants and vowels combine according to the canonical form (((C)C)C)V to form distinct characters. In addition to independent vowels and consonants, Indic scripts use alphabetic pieces known as vowel modifiers or dependent vowel signs. The position of the vowel modifiers or *maatraas* and the resulting variation in the shape of the consonants is very different in each of the nine scripts.

Figure 14.1 shows the vowels in all nine scripts and a single consonant with each of the vowel modifiers applied. Some vowels are not present in all scripts and the corresponding entries are shown blank. The basic consonants in each of the nine scripts are shown in Fig. 14.2. These scripts are also characterized by a large number of consonant conjunct forms where the characters change shape depending on their context. The appearance of the consonants is sometimes also affected by its ordering with respect to other characters. This results in a large number of character glyphs and hence poses a challenge for reading systems.

The nature of the variations in shapes between characters is physically very minute and hence classifiers find it difficult to overcome this confusion.

IPA	Devanagari	Bangla	Gurmukhi	Gujarati	Oriya	Tamil	Telugu	Kannada	Malayalam
a	अ का अ का	অ কা আ কা	ਅ ਕਾ	અ કા	ଆ କା	அ கா	అ కా	ಆ ಕಾ	അ കാ
a:	आ का आ का	আ কা আ কা	ਅ ਕਾ	અ કા	ଆ କା	அ கா	అ కా	ಆ ಕಾ	അ കാ
i	इ की ई की	ই কী ঈ কী	ਇ ਕੀ	ઇ କି	ঈ କି	இ କි	ఇ କි	ଇ କි	ഇ കി
i:	आई की	আই কী	ਏ ਕੀ	ઈ କି	ଆ କି	ஐ କි	ఏ କි	ଇ କି	ഐ കി
u	उ कू ऊ औ औ	উ কূ ঊ ঔ ঔ	ਊ କୁ	ଓ କୁ	ও କି	ஓ କୁ	ఉ କୁ	ଓ କି	ഉ കു
u:	ऊ कू औ	ও কূ ঔ	ଓ କୁ	ও	ଔ	ଓ	ও	ଓ	ഔ
ṛ	ऋ कृ	ঋ কৃ	ੱਕੁ	ર	ରୁ	ରୁ	ରୁ	ରୁ	ରୁ
ṛ:	ঋ কৃ	ରୁ	ରୁ	ରୁ	ରୁ	ରୁ	ରୁ	ରୁ	ରୁ
—	ଲ କୁ	ল কু	ଲୁ	ଲ	ଲୁ	ଲୁ	ଲୁ	ଲୁ	ଲୁ
।।	ଲୁ କୁ	ଲୁ	ଲୁ	ଲୁ	ଲୁ	ଲୁ	ଲୁ	ଲୁ	ଲୁ
e	ऐ कে	এ কে	ਐ	ଏ	ଏ	எ	ఎ	ಎ	എ
e:	আই এ	আই কে	আই	ଏ	ଏ	ଏ	ଏ	ଏ	ଏ
ai	আই	আই	ଆই	ଏଇ	ଏଇ	ଏଇ	ଏଇ	ଏଇ	ଏଇ
o	আো কো	ও কো	আো	ও	ଓ	ଓ	ଓ	ଓ	ଓ
o:	আো কো	ও কো	আো	ও	ଓ	ଓ	ଓ	ଓ	ଓ
au	আৌ ও	কৌ	আৌ	ও	ଆৌ	ଓ	ଓ	ଓ	ଓ
ঠ	ক	ক	କ	କ	କ	କ	କ	କ	କ
ঠঠ	ক	ক	କ	କ	କ	କ	କ	କ	କ

**Fig. 14.1** Vowels and a consonant with corresponding vowel modifiers

Additionally, handwritten characters are further compounded by writer idiosyncrasies in writing these complex shapes. Post processing using language models attains greater importance in the classification of such large character sets with minor structural differences.

## Summary of the State of the Art

While the state of the art in CJK character recognition has seen significant advances over the past few decades, evaluation of early research results was on relatively smaller private data sets, which made the comparison of the efficacy of different methods very difficult. Recent years have seen the release of large openly accessible data sets such as CASIAHWDB/OLHWDB [4] and international contests such as the ICDAR 2011 Chinese Handwriting Recognition Competition [5] that have made the objective assessment of the state of the art in recognition of CJK scripts feasible.

Recognition systems for machine-printed CJK documents are mature and readily available. The recognition performance for machine-printed documents has reached

	IPA	Kannada	Devanagari	Bengali	Gurmukhi	Gujarati	Tamil	Telugu	Kannada	Malayalam
k	k	ಕ	ક	କ	ਕ	ક	க	క	କ	ക
k <sup>h</sup>	g	ಗ	ગ	ଘ	ਗ	ગ	ங்	ଗୁ	ଗୁ	ഘു
g	g <sup>h</sup>	ಗ	ଘ	ঝ	ঝ	ঝ	ங	ଙୁ	ଙୁ	ঝু
y	e	ಯ	ୟ	ଯ	ୟ	ୟ	ய	ୟ	ୟ	ୟ
e	e <sup>h</sup>	ଯ	ୟ	ଯ	ୟ	ୟ	ୟ	ୟ	ୟ	ୟ
j	ju	ଜୁ	ଜ	ଜ	ଜ	ଜ	ஜ	ଜୁ	ଜୁ	ଜୁ
j <sup>h</sup>	ju <sup>h</sup>	ଜୁହ	ଜ	ଜ	ଜ	ଜ	ଜୁହ	ଜୁହ	ଜୁହ	ଜୁହ
t	t <sup>h</sup>	ତୁ	ତ	ତ	ତ	ତ	த	ତୁ	ତୁ	ତୁ
d	d <sup>h</sup>	ଦୁ	ଦ	ଦ	ଦ	ଦ	ତ	ଦୁ	ଦୁ	ଦୁ
n	n <sup>h</sup>	ନୁ	ନ	ନ	ନ	ନ	ந	ନୁ	ନୁ	ନୁ
p	p <sup>h</sup>	ପୁ	ପ	ଫ	ଫ	ଫ	ப	ପୁ	ପୁ	ପୁ
b	b <sup>h</sup>	ବୁ	ବ	ବ	ବ	ବ	வ	ବୁ	ବୁ	ବୁ
b <sup>h</sup>	m	ମୁ	ମ	ମ	ମ	ମ	ம	ମୁ	ମୁ	ମୁ
m	j	ଯୁ	ଯ	ର/ର୍	ର	ର	ர	ଯୁ	ଯୁ	ଯୁ
r	r <sup>h</sup>	ରୁ	ର	ର	ର	ର	ர	ରୁ	ରୁ	ରୁ
l	l <sup>h</sup>	ଲୁ	ଲ	ଲ	ଲ	ଲ	ல	ଲୁ	ଲୁ	ଲୁ
v	s	ଶୁ	ଶ	ଶ	ଶ	ଶ	ஶ	ଶୁ	ଶୁ	ଶୁ
s	s <sup>h</sup>	ଶୁହ	ଶ	ଶ	ଶ	ଶ	ஶ	ଶୁହ	ଶୁହ	ଶୁହ
h	h	ହୁ	ହ	ହ	ହ	ହ	ஹ	ହୁ	ହୁ	ହୁ

Fig. 14.2 Consonants in Indic scripts

high levels of accuracy. For example, the THOCR system developed by the Department of Electronic Engineering, Tsinghua University, reports the highest correct character recognition rate of 98.6 % [6].

Handwritten document analysis and recognition in CJK scripts has been a very active research topic over the past 40 years. Despite these research efforts handwritten text recognition can still be considered an unsolved problem as evidenced by low accuracies on freely written samples [7]. To stimulate research in this field, the National Laboratory of Pattern Recognition (NLPR), Institute of Automation of Chinese Academy of Sciences (CASIA), has released large data sets of free handwriting such as CASIA-HWDB/OLHWDB [4].

A sense for the accuracy of Chinese handwriting recognition can be obtained by reviewing the most recent results of the ICDAR 2011 Chinese Handwriting Recognition Competition [5]. The results from multiple participants represent the state-of-the-art performance in handwritten Chinese character recognition for both online as well as offline documents. The databases CASIA-HWDB and CASIA-OLHWDB contain offline and online handwritten characters and continuous text written by 1,020 writers. The samples include both isolated characters as well as handwritten text. The data sets of isolated characters contain about 3.9 million samples of 7,356 classes (7,185 Chinese characters and 171 symbols), and the datasets of handwritten text contain about 5,090 pages and 1.35 million character samples. All the data have been segmented and annotated at character level, and each data set is partitioned into standard training and test subsets.

In the case of OCR of Indic scripts, much of the published research is on Devanagari and Bangla scripts and there is relatively little work on other scripts. In the absence of standard open data sets and competitions, results have been reported on closed sets of various sizes and in some cases on data sets of unspecified size. Also, much of the work appears to be on digit and character recognition (both machine print and handwritten) and there are only a handful of papers that specify performance results at the word recognition level. Even though character recognition results are reported in the mid-to-upper 90 % range, word recognition performance tops out at about 87 % even with the use of various post-processing methods and language models. While references to in-house end-to-end systems have been reported in the literature, they do not appear to be widely available commercially even for the processing of machine-printed documents. Initiatives such as the Technology Development for Indian Languages (TDIL) program of the Government of India have served to coordinate the OCR research and development efforts for Indic scripts in all of the premier academic research labs in the country. In a consortium-mode project, this effort has led to significant progress in the development of OCRs for a number of Indic scripts. The BBN Byblos OCR System also has Indic language support. The state of the art clearly shows significant performance gains in the past few years in basic recognition but also indicates that much still needs to be done before handwritten and multilingual, multi-script documents can be automatically processed for digital transcription.

## Recognition of CJK Scripts

This section highlights the key challenges in the recognition of documents in Chinese, Japanese, and Korean scripts.

### Preprocessing

Traditionally CJK texts are printed or handwritten using black ink on evenly colored background such as plain white paper or red paper in the case of special documents of cultural significance. Binarization of scanned CJK documents is not very different from documents in other scripts. Targeted preprocessing techniques required for CJK document images are for text line orientation detection, word segmentation, character segmentation, and script identification.

CJK documents may have either horizontal or vertical text layout. Both text layout directions may be used in the same document page. When horizontal layout is used, the text reads from left to right and the line order is from top to bottom. When vertical layout is used, the text reads from top to bottom and the line order is from right to left. Text line orientation detection algorithms presented in [8,9] are based on the observation that inter-character spacing is generally smaller than interline spacing. A connected-component-based approach using a minimal spanning tree is applied in calculating the predominant direction of the text line.

Most text line separation algorithms designed for Latin-based language document images are also applicable to CJK documents. Due to the complexity of the character structure, the text sizes in CJK documents are generally bigger than Latin-based language documents. This is a minor source of concern for those algorithms which are size dependent/sensitive. Overall text line separation for CJK documents, both machine-printed as well as handwritten, is relatively easier than for Latin-based language documents.

### Segmentation

In Latin-based language documents, words are generally separated by relatively wider white spaces within a text line. Words that are split across text line are hyphenated. But in Chinese documents words are not delimited by any mark or space. Line boundaries cannot be used for word segmentation either. Punctuations indicate only the boundary of sentences. In Japanese documents the word boundaries are often found at the transition from Hiragana characters to Kanji. In Korean documents, words are separated by spaces but may be broken across text lines without hyphenation. For above reasons, most CJK document recognition systems proceed directly to character segmentation after line separation.

Accurate character segmentation is a vital step in the processing of CJK documents. When a document includes only CJK characters, segmentation of printed

characters is generally an easy problem. Simple projection profile methods are often sufficient. When a document contains multiple scripts such as Chinese with English, or CJK character with Latin numerals, character segmentation may be nontrivial. For multiple script documents, a two-path approach has worked well where the first pass labels the text line segments which are either CJK or Latin characters, and then in the second pass, the characters in each language are segmented with a language-dependent algorithm.

Segmentation of handwritten CJK characters is a much more complicated problem. There are several methods for handwritten CJK character segmentation based on a combination of character recognition and statistical approaches. Final character segmentation results are determined from multiple segmentation candidates.

## Character Recognition

Character recognition algorithms for CJK can be broadly categorized into three groups: radical-based algorithms, stroke-based algorithms, and holistic approaches.

### Radical-Based Algorithms

Most CJK character such as Chinese characters and Korean characters are logo-graphic, which are visual symbols representing a minimal meaningful unit of the language. Each character is a graph that is composed of simple basic graphs called radicals. Each radical appears in a specific position, such as left-hand radical and upper radical. A small number of such radical primitives can be put together to compose many different Chinese and Korean characters [10].

Radical-based approaches decompose the large set of characters into a much smaller set of radicals. Then the character recognition is reduced to matching of radicals to a character image. The best match determines the identity of the character (Fig. 14.2). Modeling radicals is the most important processing step in radical-based recognition. An active shape modeling method has been successfully used for modeling radicals [10]. Firstly a radical character image is converted to its skeleton representation. Then the critical landmark points on the skeleton are labeled. Finally using the landmark points as features, a principal component analysis is applied to calculate the main shape parameters that capture the radical variations. In [10] a subset of 200 most common and productive radicals were selected to cover 2,154 Chinese characters out of the 3,755 GB2313 set. Based on the possible location of radical locations in forming the character, the character image is decomposed into multiple partitions such as left-right, up-down, or quadric sections. To match radicals in a character, a chamfer distance is used to find the optimal match of radicals in the modeled set to that in the partitions of the character image.

### Stroke-Based Algorithms

Stroke-based approach uses finer level structures for composing a character image. Rather than modeling radicals, stroke-based approach decomposes a character image into a set of strokes, which are usually defined as the writing segment from a pen-down to a pen-up in the ideal cases. The recognition of the character is reduced to recognition of the stroke shapes and positional information of the stroke set in forming the character [11].

Recognition of a stroke image is usually done by extracting the direction features such as the four directions or eight directions (Fig. 14.3).

To extract stroke segments, segment feature points are first identified on the skeleton of the character image. The segment feature points include end points and fork points. Skeleton sections between feature points are extracted as stroke segments. Statistical-structural scheme based on Markov random fields (MRFs) can be used in the recognition of the stroke sequence [12]. The stroke segments are represented by the probability distribution of their locations, directions, and lengths. The states of MRFs encode the information from training samples. Through encouraging and penalizing different stroke relationships by assigning proper clique potentials to spatial neighboring states, the structure information is described by neighborhood system and multi-state clique potentials and the statistical uncertainty is modeled by single state potentials for probability distributions of individual stroke segment. Then, the recognition problem can be formulated into MAP-MRF framework to find an MRF that produces the maximum a posteriori for unknown input data. More details can be found in [12].

### Statistical Feature Approaches

Although radical-based approach and stroke-based approach reduce the number of target classes to a small set of radicals or strokes, they often encounter difficulties in matching radicals to the character image or finding the optimal stroke sequence. Statistical approach is used to directly extract features from the character image. The most popular features for Chinese character recognition include peripheral shape features, stroke density features, stroke directional features, and gradient features [13, 14].

*Peripheral features* [13]: To extract peripheral features, an input character image is first partitioned into  $2 \times 4$  bins. The character image is separated into left and right parts each of which consists of four horizontal strips labeled by  $S_h^1 S_h^2 \dots S_h^8$  in the character image frame. Suppose strip  $S_h^i$  consists of  $k$  lines. Let  $|\lambda_h^i|$  be the distance between the outermost stroke edge and the character image frame along  $i$ -th line and  $H_h^i \times V_h^i$  be the subarea of the  $i$ -th horizontal strip  $S_h^i$ . The horizontal peripheral background area in the  $i$ -th horizontal strip  $S_h^i$  can be represented by the following formula:

$$A_h^i = \left\{ \sum_{i=1}^k |\lambda_h^i| \right\} / (H_h^i \times V_h^i)$$



**Fig. 14.3** Example of the two-pass method for character segmentation. *Left:* Identification of multiple language script. *Right:* Character segmentation result

Similarly, horizontal peripheral line difference  $\Psi_h^i$ , vertical peripheral back ground  $A_v^i$ , and vertical peripheral line difference  $\Psi_v^i$  are defined as

$$\begin{aligned} A_h^i &= \left\{ \sum_{i=1}^k |\lambda_h^i| \right\} / (H_h^i \times V_h^i) \\ \Phi_h^i &= \mathfrak{R}_h^i / \sum_S \mathfrak{R}_h^i \\ \psi_h^i &= \left\{ \sum_{i=1}^k ||\lambda_h^i - \lambda_h^{i+1}|| \right\} / (H_h^i \times V_h^i) \end{aligned}$$

*Stroke Density Features* [13]: 16 stroke density features can be extracted in two different ways, depending on the directions of the elastic meshing used (see Fig. 14.4). Suppose there are  $m$  horizontal inspection lines in horizontal strip  $S_i$ , the crossing count on inspection line  $r_j$  is

$$\frac{1}{2} \sum_x \overline{f(x, y)} f(x + 1, y)$$

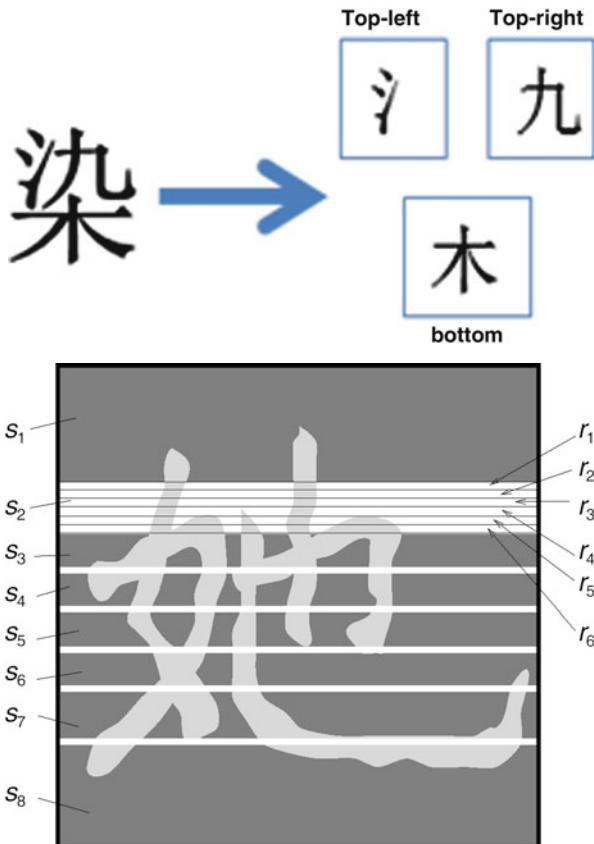
The horizontal regional stroke density (HRSD) is the quotient of the accumulation of the crossing counts in  $S_i$  divided by the number of horizontal inspection lines in horizontal strip  $S_i$ . It is denoted by  $\mathfrak{R}_h^i$  and computed by the following formula:

$$\mathfrak{R}_h^i = \left\{ \frac{1}{2} \sum_{j=1}^m \sum_x \overline{f(x, y)} f(x + 1, y) \right\} / m$$

Similarly horizontal regional stroke density distribution (HRSDD)  $\Phi_h^i$ , vertical regional stroke density (VRSD)  $\mathfrak{R}_v^i$ , and vertical regional stroke density distribution (VRSDD)  $\Phi_v^i$  can be calculated by (Fig. 14.5)

$$\begin{aligned} \Phi_h^i &= \mathfrak{R}_h^i / \sum_S \mathfrak{R}_h^i \\ \mathfrak{R}_v^i &= \left\{ \frac{1}{2} \sum_{j=1}^m \sum_x \overline{f(x, y)} f(x, y + 1) \right\} / m, \quad \Phi_v^i = \mathfrak{R}_v^i / \sum_S \mathfrak{R}_v^i \end{aligned}$$

**Fig. 14.4** Example of a Chinese character decomposed into idealized radicals



**Fig. 14.5** Stroke density features (From Ref. [13])

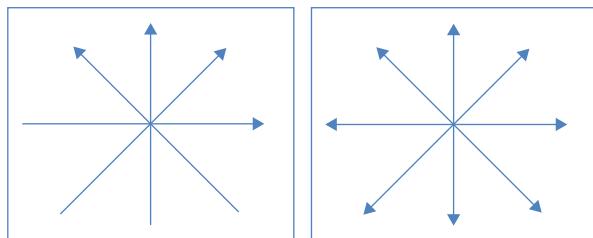


*Gradient Features* [14]: To extract gradient features,  $3 \times 3$  Sobel operators are used to get the horizontal and vertical grayscale gradient at each image pixel, respectively.  $L$  directions are defined with an equal interval  $2\pi/L$ , and the gradient vector is decomposed as illustrated in Fig. 14.6. An  $L$ -dimensional gradient code is calculated at each image pixel. To quantify the gradient code into a feature vector, the  $65 \times 65$  normalized image is divided equally into  $13 \times 13$  sub-blocks; within each sub-block the  $L$ -dimensional gradient codes are summed up and then the resolution of sub-block is down sampled to  $7 \times 7$  by a Gaussian filter, resulting in a  $d$ -dimensional gradient feature, where  $d = 7 \times 7 \times L$ . Finally a variable transformation  $y = x^{0.4}$  is applied on each element of the extracted feature vector to make its distribution more Gaussian-like (Fig. 14.7).

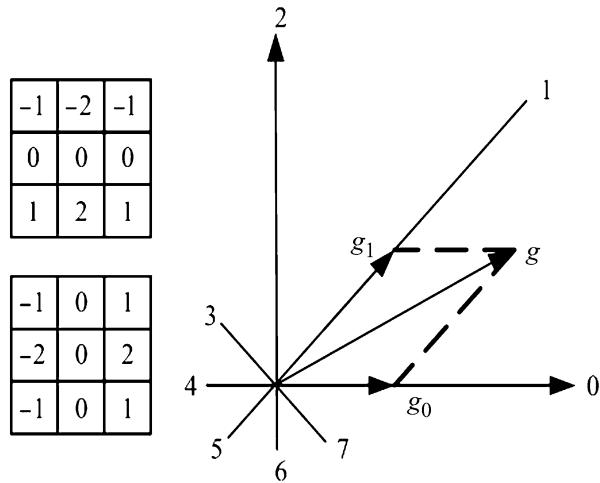
## Classification

Classification is one of the key modules in any feature-based OCR system. Many classification methods have been designed and implemented in OCR systems for

**Fig. 14.6** Stroke features are extracted from four directions (left) or eight directions



**Fig. 14.7** Gradient feature extraction using Sobel operators and gradient vector decomposition (eight directions for illustration)  
(From Ref. [14])



recognition of Asian scripts. Among them the most popular and successful methods include nearest-neighbor (NN)-based methods, methods based on quadratic discriminant functions and their modifications (QDF and MQDF), and convolutional neural networks (CNN).

### Nearest-Neighbor Classifiers

The nearest-neighbor classifiers are built upon preselected prototypes which are feature vectors with assigned class ids [6]. The nearest-neighbor (NN) rule assigns an unknown sample to the class of its nearest prototype. The performance of a nearest-neighbor classifier is affected by the selection scheme of the training prototypes, the number of the prototypes, and the metrics used for measuring the distances. The  $k$ -nearest-neighbor ( $k$ -NN) rule is a popular nonparametric technique. The performance of  $k$ -NN classifier asymptotically approximates the Bayesian classifier if the number of training samples approaches infinity. The nearest-neighbor (1-NN) rule is a special case of  $k$ -NN rule. It was proved that the asymptotic classification error of 1-NN rule is bounded by twice the Bayesian error rate [15]. However due to the requirement of large size prototypes needed especially in the case of Asian character recognition, which deals with large number of classes, the computation efficiency and storage space are among the challenges.

Many approaches have been used to improve the performance of NN classifier in various aspects. To improve the classification performance of NN rule under a finite number of samples, different distance metrics have been used. In addition to improving the efficiency, the selection of prototypes from the original training sample set by adding or pruning can preserve the classification performance by using some heuristics. Some algorithms have been proposed to select prototypes with the aim of optimizing the classification performance [16].

### **Modified Quadratic Discriminant Function Classifier (MQDF)**

For regular handwritten Chinese character recognition the feature distributions are near Gaussian distributions, which can be classified by the modified quadratic discriminant function classifier (MQDF). For a  $d$ -dimensional feature vector  $x$ , the QDF distance can be represented as follows:

$$g_i(x) = (x - \mu_i)^T \phi_i^{-1} (x - \mu_i) + \log |\phi_i|, i = 1, 2, \dots, C$$

where  $C$  is the number of character classes and  $\mu_i$  and  $\phi_i$  denote the mean vector and the covariance matrix of class  $\omega_i$ . Applying decomposition on  $\phi_i$  and replacing the minor eigenvalues with a constant  $\sigma^2$  to compensate for the estimation error caused by small training set, the MQDF f or a  $d$ -dimensional feature vector  $x$  is given as formula:

$$\begin{aligned} g_i(x) &= \frac{1}{\sigma^2} \left\{ \|x - \mu_i\|^2 - \sum_{j=1}^q \left( 1 - \frac{\sigma^2}{\lambda_{ij}} \right) [\phi_{ij}^T (x - \mu_i)]^2 \right\} \\ &\quad + \sum_{j=1}^q \log \lambda_{ij} + (d - q) \log \sigma^2, i = 1, 2, \dots, C \end{aligned}$$

The formula defines the distance of sample  $x$  and class  $\omega_i$ .  $\lambda_{ij}$  and  $\phi_{ij}$  denote the  $j$ th eigenvalue in descending order and the corresponding eigenvector of the covariance matrix of class  $\omega_i$ , respectively;  $q$  denotes the number of dominant principal axes,  $q < d$ . The parameters are usually estimated using maximum likelihood (ML) framework on the training set. During classification, MQDF calculates distances between the test sample and each class according to above formula and arranges classes according to their corresponding distances in the increasing order to obtain the multiple candidate recognition results.

### **Convolutional Neural Network (CNN)**

Convolutional neural networks are a type of neural networks which combine feature extraction and classification. They have been shown to perform better than probability density function approaches such as Gaussian Bayesian methods and Gaussian mixture models or nonparametric models such as  $K$ -nearest-neighbor classifiers. Biologically inspired from the human visual system, the distinctive

characteristic feature of a convolutional neural network is its integral functionality in combining multiple modules of conventional recognition systems. Usually character recognition systems have three basic modules: preprocessing feature extraction, and classification. In convolution neural network approach, raw images are taken as input and appropriate features are implicitly extracted for classification using a trainable neural network.

Convolutional feed forward neural networks were first described by LeCun et al. [17]. Convolutional neural networks are based on the modeling of the human retina structure. They are designed to recognize two-dimensional shapes with a high degree of invariance to shift, scaling, and distortion [17]. The main advantage of a convolutional neural network architecture is its implicit capability for automatic feature extraction. The input is the raw image and the output layer represents the winning classes as a classifier. Different layers are responsible for feature extraction, mapping, and subsampling tasks in addition to a traditional multilayer perceptron structure in the last stage working as a classifier. To train a CNN, the back propagation algorithm is successfully generalized for CNN's topology, which is known as deep neural networks learning [18]. The general topology of a CNN uses three architectural structures: local receptive field, shared weights, and spatial sub sampling [17]. Specifically, CNN architecture is made of one input layer and three types of hidden layers and one output layer. The first kind of hidden layers is responsible for convolution; the second kind of hidden layers is responsible for local averaging, subsampling, and resolution reduction; and the third kind of hidden layers act as a traditional multilayer perceptron classifier. In the first and second types, the extracted local features are saved in the output planes which are called feature maps. Then the extracted spatial features are classified in the last kind of layers which is a multilayer perceptron with two hidden layers.

Convolutional neural networks have been implemented and used successfully for Asian language character recognition. A CNN with nine hidden layers was used by one of the entries in the ICDAR 2011 Offline Chinese Handwriting Competition [5].

## Post-processing

Post-processing is a vital step for improving recognition performance especially for Asian character recognition due to the high likelihood of confusion between character shapes given the large set of classes. In order to improve the recognition rate, language models that can utilize context can be used to correct the raw recognition results. Post-processing approaches based on linguistic knowledge also include using a lexicon [19, 20] or syntactic and semantic rules [21] to correct the spelling of words and using statistical language models (LMs) [22, 23] to select the best sequence from the candidate characters given by the OCR engine.

Statistical language models (LMs) have been widely used for the contextual post-processing to improve the accuracy in recognizing the Chinese scripts [24–27].

In Asian languages, conventional  $n$ -gram LMs such as character bigram, character trigram, or word bigrams have been utilized in post-processing of recognition results. Class-based LMs have also been shown to improve the performance of recognition systems when combined with conventional word-based LMs [28]. Hybrid bigram LMs combining both word- and class-based bigrams have also been tried [29].

The most widely used LMs are the  $n$ -gram models. The conventional  $n$ -gram Chinese LMs can be based on either characters or words. Based on characters, for  $n = 2, 3$ , the character-based bigram model and the character-based trigram model are expressed, respectively, as follows:

$$p(S) = p(s_1) \prod_{t=2}^T p(s_t | s_{t-1})$$

and

$$p(S) = p(s_1)p(s_2|s_1) \prod_{t=3}^T p(s_t | s_{t-2}s_{t-1})$$

where  $S = s_1s_2 \dots s_T$  is a sequence of characters given by an isolated character recognizer, in which each output  $s_t$  may include the top  $K$  candidates  $c_1c_2 \dots c_K$  with the corresponding distance measurement values  $D = d_1d_2 \dots d_K$ . The output of the system  $O = o_1o_2 \dots o_T$  is the final sentence.

Considering the top  $K$  candidates for each output  $s_t$ , there are  $K^T$  possible sentences. The post-processing task is then to select the best sentence from all the  $K^T$  sentences. Applying the rule of maximal posterior probability, the output  $O$  of a recognition system is formulated as

$$O = \arg \max_S p(S) \times \prod_{t=1}^T p(s_t | x_t)$$

where  $S$  is one of above statistical LMs;  $p(s_t | x_t)$  is the posterior probability of  $s_t$  given  $x_t$ , which can be computed from candidate confidence.

Similarly the word-level bigram and trigram models are

$$p(S) = p(w_1) \prod_{t=2}^T p(w_t | w_{t-1})$$

and

$$p(S) = p(w_1)p(w_2|w_1) \prod_{t=3}^T p(w_t | w_{t-2}w_{t-1})$$

where  $S = w_1w_2 \dots w_T$  is a sequence of words given by a word recognizer.

## Recognition of Indic Scripts

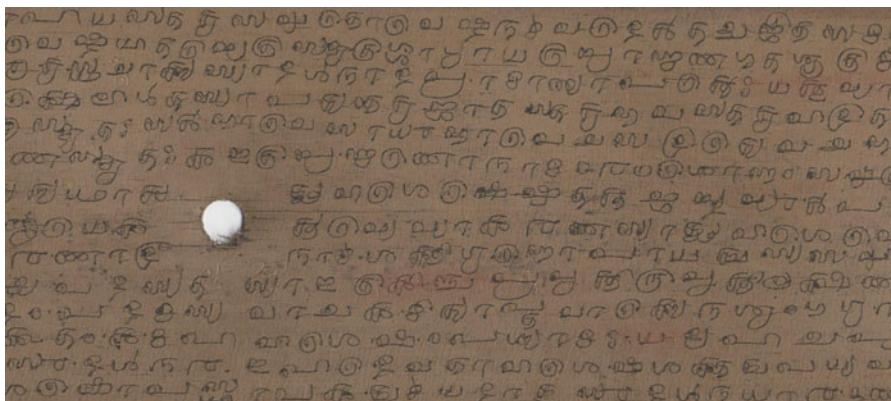
This section highlights the key challenges in the recognition of documents in Indic scripts.

### Preprocessing

Preprocessing of document images for OCR applications usually involves (i) page segmentation to narrow in on the text regions, (ii) noise removal of various types, (iii) binarization to separate the foreground text from the background since most recognition techniques have been designed for binarized images, and (iv) line segmentation ([►Chaps. 4](#) (Imaging Techniques in Document Analysis Process) and [►8](#) (Text Segmentation for Document Recognition)). In general, most of the methods developed for preprocessing are script-agnostic. However, occasionally there are features of certain scripts that need to be taken into account while applying general preprocessing algorithms and, for some script-specific traits, may even require targeted preprocessing. There are also some document types that are more frequently encountered in conjunction with certain scripts. The emphasis of this chapter is on only those elements of preprocessing that have particular relevance to a specific Asian script itself or to document types that are more frequently encountered in the context of Asian scripts.

A few of the preprocessing issues that require adaptations for Indic scripts are addressed here:

- (i) The *shirorekha* or headline which is a horizontal bar that connects characters in a word in scripts such as Devanagari and Gurmukhi – location and removal of these *shirorekhas* in machine-printed documents is fairly trivial and can be achieved using simple projection profile-based methods [30]. Skew is also easy to detect due to the locally linear *shirorekha* lines [31, 32]. However, in handwritten documents, the *shirorekhas* may not be very uniform and effective detection and removal requires localized adaptations of the projection profile approach [33, 34]. Morphological operations such as erosion and dilation have also been used to identify *shirorekhas* in handwritten images [35].
- (ii) Line segmentation Separating text lines correctly and grouping diacritics and other small components with the correct line are crucial to the success of downstream OCR processes. In handwritten documents in general, it is common to see variability in skew between different lines in the same document leading to varying distances between lines as well as overlapping and touching of strokes from multiple lines. This is further exacerbated in handwritten documents using scripts such as the Indic scripts due to the presence of vowel modifiers and conjuncts that are dispersed all around the base character region leading to touching between strokes from multiple lines. Localized adaptive projection profile [36] and morphology-based methods [37] have been used to detect the location of text lines, but splitting touching components intelligently using



**Fig. 14.8** Historical palm-leaf manuscript

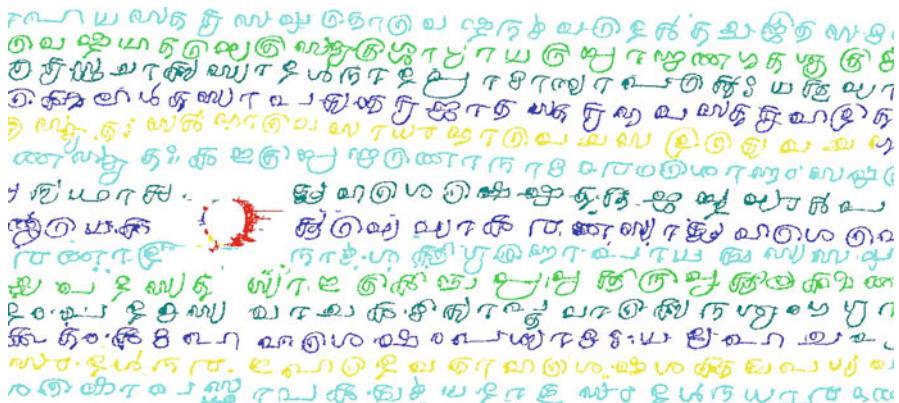
stroke following and other approaches appears to have not been investigated in any great detail for handwritten Indic script documents.

- (iii) Historical documents in Indic scripts can be found on a variety of non-paper surfaces such as palm leaves, parchment, and stone and wood inscriptions. Ancient palm-leaf manuscripts relating to religion, science, medicine, and astronomy are still available for reference today due to many ongoing efforts for preservation of ancient documents by libraries and universities around the world. These manuscripts typically last a few centuries but with time the leaves degrade and the writings become illegible. Image-processing techniques have been used to help enhance the images of these manuscripts so as to enable readability of the written text for human eyes as well as machine reading.

Figure 14.8 shows an example of a palm-leaf manuscript.

Most document image enhancement algorithms have been designed primarily for binarization of modern documents. An overview of the traditional thresholding algorithms for text segmentation are given in [38] which compares various techniques such as Otsu's, entropy techniques, and the minimal error technique. Entropy-based methods specifically designed for historical document segmentation that deal with the noise inherent in the paper especially in documents written on both sides such as background noise and bleed-through text using direct image matching and directional wavelets as well as other methods designed for improving human readability while maintaining the original “look and feel” of the documents fail on palm-leaf manuscripts due to low contrast and varying color intensity of the background.

One method used for enhancing digital images of palm-leaf and other historical manuscripts approximates the background of a grayscale image using piecewise linear and nonlinear models. Normalization algorithms are used on the color channels of the palm-leaf image to obtain an enhanced gray scale image and an adaptive local connectivity map is used to try to segment lines of text from the enhanced images with the objective of facilitating techniques such as keyword



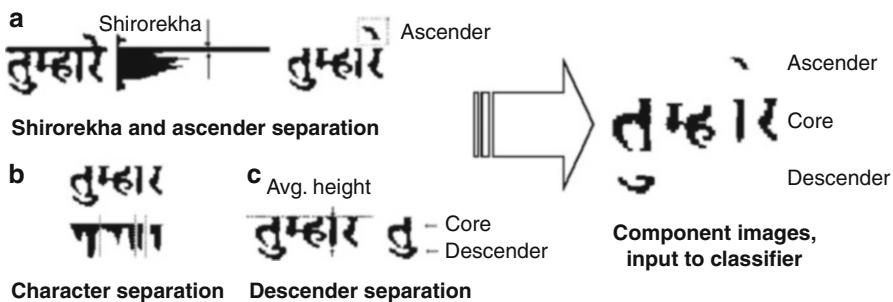
**Fig. 14.9** Palm-leaf manuscript image after preprocessing (binarization and line segmentation)

spotting or partial OCR and thereby making it possible to index these documents for retrieval from a digital library. Figure 14.9 shows the results of line segmentation on the binarized image obtained using this method [39].

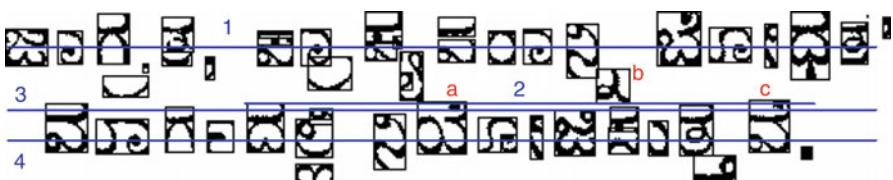
## Segmentation

Segmentation of characters can be a particularly critical issue for Indic scripts depending on the system architecture. This stems from the use of vowel modifiers and conjunct consonants.

- (i) Placement of vowel modifiers – in scripts such as Devanagari, the modifiers are typically connected to the characters and can usually be found on either side of or above and below the base character, e.g., the base consonant ka (क) takes the shapes (का, कि, की, कृ, कू, कृ, के, कै, को, कौ, कं, कः) with different vowel modifier, whereas in scripts such as Kannada, the modifiers can involve multiple pieces positioned around the character (up, down, left, right) and may or may not be connected to the base character, e.g., the base consonant ka (ಕ) takes the shapes ಕಾ, ಕಿ, ಕೀ, ಕು, ಕೂ, ಕೃ, ಕೇ, ಕೈ, ಕೋ, ಕೌ, ಕೆ, ಕೊ, ಕೊಂ, ಕೊಃ.
- (ii) Conjunct consonants – in Devanagari, conjuncts are represented by partial character shapes for the initial consonants followed by the full consonant shape for the last consonant (e.g., sya (स्य), ghne (घ्ने)) with the last consonant taking the vowel modifiers. In Kannada, the conjunct consonants have the first consonant in the sequence retain its complete shape whereas the second and subsequent consonants are smaller components sometimes with a new distinct shape under the first consonant and the first consonant takes the vowel modifiers (e.g., kree (ಕ್ರೀ), dve (ಡ್ವೀ)). In Tamil, the initial consonants are represented in their full form with a dot and the last consonant in its full form (e.g., pra (ப்ர)) and the last consonant takes the vowel modifiers. So, depending on the script and the techniques being used, one would need to decide whether to segment



**Fig. 14.10** Segmentation-driven OCR for Devanagari



**Fig. 14.11** Line segmentation in a Kannada document [43]

the character images into semantic sub-parts such as half-consonant, consonant, vowel, and vowel modifier prior to recognition or whether to treat the shape as a separate class.

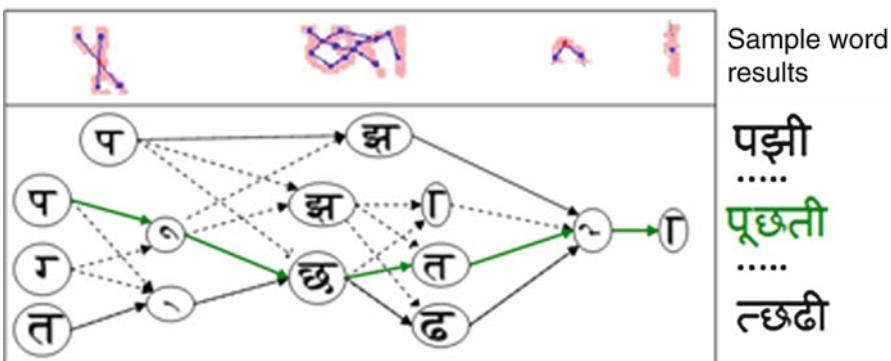
Segmentation-driven OCR techniques split words into smaller units, with some using composite characters and others using finer semantic subparts as classification units. In the case of Devanagari, horizontal and vertical profiles are used to remove the shirorekha and separate the characters within the word. Word separation techniques developed for Latin characters work well for word segmentation in other Indic scripts that do not use the shirorekha (Fig. 14.10).

In the former design, the character images are processed directly using classifiers to obtain recognition results. The number of character classes is very large when compared to the number of unique semantic subparts. The ILT image data set [40] has 973 character classes, and the EMILLE text data set [41] has 5,573 character classes, and both have approximately 128 unique semantic subparts. Typically, classifier design increases in difficulty with an increase in class space. For example, 537 character classes from the ILT data set occur less than 25 times each. Due to lack of training samples, these classifiers often drop rare characters from the class space. Since a large number of classes have few samples, removing rare characters from the class space adversely affects overall OCR results. Using characters as the unit of classification for Devanagari is therefore less desirable [42] (Fig. 14.11).

Font variations and poor print quality add to the challenge. Techniques that assume uniform font attributes and rely on information extracted from a paragraph are not very robust especially when considering multi-font documents. A method



**Fig. 14.12** Generating block adjacency graph representation of a Devanagari word image



**Fig. 14.13** An example of a scheme where recognition results are generated based on multiple segmentation hypotheses and post-processing based on language models is used to identify the correct hypothesis

that does not depend on font statistics and accommodates character distortions is critical to process typical Indic documents.

Techniques that use the semantic subparts as classification units benefit from a reduction in the number of classes. However, segmenting characters into the semantic subparts before classification is often nontrivial and depends on using a representation that is suited to carry out this segmentation. Simple segmentation using structural attributes like location and number of vertical strokes in a character or average character size in the paragraph often fails in segmenting conjuncts such as (स्य) ghne (अने) since the characters are conjoined with each other along nonlinear boundaries.

Graph-based representations provide a better handle on segmenting these connections along nonlinear boundaries so that the split segments are more representative of the true character shapes and hence can provide better classification performance. Block adjacency graphs have been used as a successful representation scheme to deal with touching characters and conjuncts [42] (Fig. 14.12).

A recognition-driven segmentation approach uses multiple segmentation hypotheses to generate character recognition results followed by the use of post-processing techniques such as dictionary lookup for lexicon pruning and language models to identify the correct segmentation hypothesis (Fig. 14.13).

## Discriminative Features

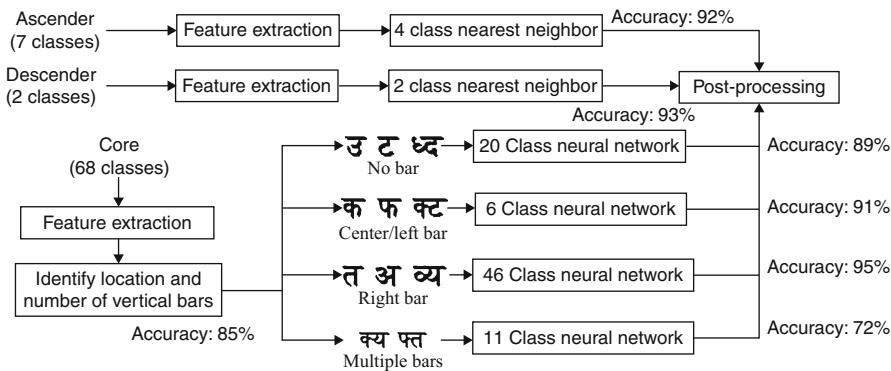
Selection of the right set of features that help discriminate between the desired classes is at the core of the recognition task. In approaches such as template matching, the entire image can be considered as the feature and in the case of machine text, the approach can provide good results. But the dimensionality of the feature is proportional to the size of the image and the high dimensionality ends up being a significant bottleneck for a real-time system. So, an added element to consider in designing effective features is to keep the dimensionality of the feature vector to a manageable size. It can be seen that many of the structural and statistical features that have been used for Latin-based scripts have also been successfully used for Indic scripts. A few of the most effective features that have been used for Indic script recognition will be reviewed in this section.

- (i) Structural or geometric features: Structural features tend to explicitly capture structural aspects of text such as ascenders, descenders, loops, branch points, and end points. So, in the case of Indic scripts, features such as presence or absence of a vertical bar as in କ vs ଖ would be a structural feature. They are useful features in the recognition of machine-printed text but are not very reliable features in the recognition of handwritten text due to the wide variations seen in handwriting in representing the same structure.
- (ii) Statistical features: Statistical features are numerical measures computed over images or regions of images. They include, but are not limited to, measures such as pixel densities, histograms of slope along strokes, moments, curvature, and Fourier descriptors. Statistical features are among the most widely used features for handwritten character and word recognition. Extraction of reliable features and meaningful quantization of the measurements are the most challenging aspects of using statistical features.

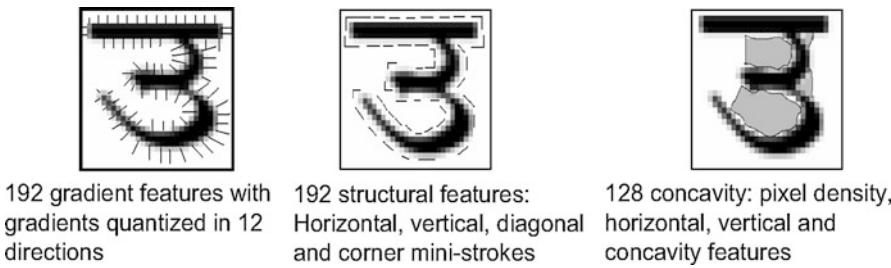
Since high dimensionality is the major bottleneck for template-matching approaches, using principal component analysis for dimensionality reduction and choosing a sufficient number of features, researchers have been able to attain high character recognition rates on scanned clean machine-printed documents partially subject to degradation models. While this method worked well on cleaner document images, a decrease in performance can be expected for newspaper and other real-life documents.

Tree-based classifiers that use simple features to roughly partition the class space into more manageable number of classes have also been successfully used in machine-printed Devanagari OCR. This method can be extended to other Indic scripts also by selecting the appropriate sets of features for pruning (Fig. 14.14).

Gradient, structural, and concavity (GSC) features are a combination of features that have been effective across many scripts including Latin and Arabic and have been successfully used in Indic script recognition as well. The gradient features represent the local orientation of handwritten strokes, the structural features extend the gradient features to longer distances and provide information about stroke



**Fig. 14.14** Tree-based (multistage) classifier scheme for Devanagari OCR



**Fig. 14.15** GSC features from Devanagari character

trajectories, and the concavity features serve to detect certain stroke relationships at long distances. They also include information such as pixel density, existence of loops, and arcs and their direction. The extraction of the features is done by using mathematical filters such as Sobel operators and scanning and tracing the character image. For each type of feature, statistical sampling is done using fixed size grids. The character image is size normalized before extracting features. The nature of the GSC feature emphasizes the shape of the characters at three different levels of granularity, fine, medium, and coarse, and hence has been effective in obtaining high recognition rates on the ILT data set [40] which contains scanned images of documents spanning a wide spectrum of variations in the quality of paper and print [40]. Even with these features, it is hard to discern the fine details such as the minute differences in vowel modifiers that are used extensively in some Indic scripts. Post-processing can help overcome some of these problems (Fig. 14.15).

Variable grid sizes have been used with GSC to account for the varying widths of patterns.

Most machine-printed systems in the literature report performance at the character level on data sets that have perfectly segmented characters.

## Classification

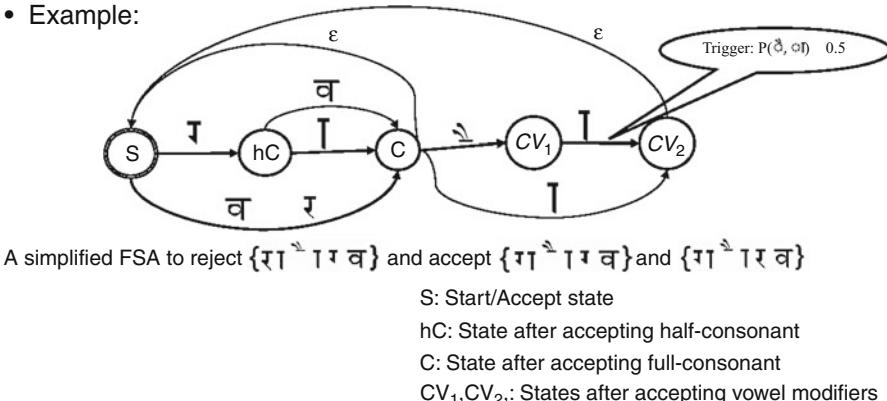
Many types of features and classification approaches have been reported in the OCR literature for Latin script recognition. Various combinations of these features and classification schemes have been tried with varying levels of success for Indic script recognition. Two broad approaches to classifier design for Indic scripts have been using complex classifier models for directly classifying the sample into one of the  $n$ -classes possible or using a hierarchical divide-and-conquer strategy where an ensemble of binary or  $k$ -ary ( $k < n$ ) classifiers are integrated to form the final  $n$ -class classifier. Several flavors of neural-network based classifiers (MLP, PNN),  $k$ -nearest-neighbor (KNN), HMM-based approaches, support vector machines (SVM), modified quadratic discriminant function (MQDF), and generalized Hausdorff image comparison have all been used for Indic script recognition. Given the wide range of feature-classifier combinations that have been used for character recognition in Indic scripts and the lack of sufficient information about the various data sets over which the numbers have been reported, it is difficult to generalize any particular combination of features-classifiers as being superior to others for Indic script recognition. However, an analysis of the word recognition approaches that have been reported indicates that effective post-processing is very crucial.

## Post-processing

In order to get satisfactory recognition results at the word and sentence levels, post-processing appears to be a necessity, especially for Indic scripts where the differences between some character shapes is very minute leading to confusions that cannot be overcome during classification. Initial approaches to achieve improved word-level performance consisted of techniques such as a simple dictionary lookup for error correction. Use of probabilistic language models to enhance recognition performance prior to using dictionary lookup that have been used successfully for Latin OCR was first used for Indic scripts in [42].

In [42], a stochastic finite state automaton (SFSA) is used to combine rule-based script composition validity checking and a probabilistic  $n$ -gram language model into a single framework. Different linguistic modeling units were evaluated in deciding the granularity that would be best suited for Devanagari word recognition. The general idea in designing an SFSA was to create a finite-state machine (FSM) that accepts strings from a particular language and assigning data-driven probabilities to transitions between states. If a candidate word is accepted, a score or probability is obtained and if rejected, corrections can be made using the minimum set of transitions that could not be traversed [44]. Here, the SFSA is modeled using script composition rules, and the transition probabilities are modeled on character or composite character  $n$ -grams. Transition probabilities for the SFSA are obtained from text corpus using the formula:

- Stochastic FSA can represent rules and statistical measures.
- Example:



**Fig. 14.16** Stochastic FSA used for Devanagari text recognition

$$a_{ij}(o) = \frac{\text{count of observing } o \text{ going from state } i \text{ to } j}{\text{Number of transitions from state } i \text{ to } j}$$

An  $n$ -gram model specifies the conditional probability of each unit, or token of text  $c_n$  with respect to its history  $c_1, \dots, c_{n-1}$ :

$$P(c_n | c_1 \dots c_{n-1}) = \frac{P(c_1 \dots c_n)}{P(c_1 \dots c_{n-1})} = \frac{\text{Count}(c_1 \dots c_n)}{\text{Count}(c_1 \dots c_{n-1})}$$

and perplexity values are reported over the test set ( $Pp$ ):

$Pp(D, q) = 2^{-\frac{1}{N} \sum_x \log q(x)}$  where  $q$  represents the probability assigned by an  $n$ -gram model. A perplexity of  $k$  indicates that while predicting a token, the language model is as surprised as a uniform, independent selection from  $k$  tokens (Fig. 14.16).

## Conclusion

As indicated in the preceding sections, there has been considerable progress in the development of techniques for Asian character recognition. Many different features and classification approaches have been successfully used to achieve high levels of accuracy in character recognition. As with other scripts and languages, handwritten text recognition is still a big challenge for Asian scripts. Unlike Arabic and cursive Latin scripts where a significant challenge is due to the difficulty of segmenting words into characters, the challenge for Asian scripts is in the ability to deal with the large number of character classes that differ only slightly in shape

leading to confusion between classes during classification. Language models for post-processing are likely to continue to play a big role in the quest to develop practical end-to-end systems that can process free-form handwritten text. Increasing availability of large data sets and a heightened interest in competitions at ICDAR and other conference venues offer the promise of rapid advances in the near future.

---

## Cross-References

- ▶ [Analysis of the Logical Layout of Documents](#)
  - ▶ [Language, Script, and Font Recognition](#)
  - ▶ [Page Segmentation Techniques in Document Analysis](#)
  - ▶ [Text Segmentation for Document Recognition](#)
- 

## References

1. The Unicode Standard, 6.0.0 (2011) The Unicode Consortium, Mountain View
2. Nagy G (1988) Chinese character recognition: a twenty-five year retrospective. In: Proceedings of the 12th international conference on pattern recognition, Rome, pp 163–167
3. Pal U, Chaudhuri BB (2004) Indian script character recognition: a survey. *Pattern Recognit* 37:1887–1899
4. Liu CL et al (2011) CASIA online and offline Chinese handwriting databases. In: Proceedings of 11th ICDAR, Beijing
5. Liu CL et al (2011) ICDAR 2011 Chinese handwriting recognition competition. In: ICDAR, Beijing
6. Ding X (2009) Advanced topics in character recognition and document analysis: research works in intelligent image and document research lab, Tsinghua University. In: Proceedings of the SPIE, San Jose, CA
7. Liu CL et al (2010) Chinese handwriting recognition contest 2010. In: Proceedings of the 2010 Chinese conference on pattern recognition, Chongqing, China
8. Kimura F (2007) OCR technologies for machine printed and hand printed Japanese text. In: Digital document processing. Major directions and recent advances. Springer, London
9. Baird HS, Ittner DJ (1993) Language-free layout analysis. In: Proceedings of the second international conference on document analysis and recognition, Tsukuba
10. Suen Y, Huang EM (1984) Computational analysis of the structural compositions of frequently used Chinese characters. In: Computer processing of Chinese and Oriental languages, World Scientific Publishing, Singapore
11. Tang KT, Leung H (2007) Reconstructing strokes and writing sequences from Chinese character images. In: Proceedings of the international conference on machine learning and cybernetics, Hong Kong
12. Zeng J, Liu ZQ (2005) Markov random fields for handwritten Chinese character recognition. In: ICDAR 2005: Proceedings of the 9th international conference on document analysis and recognition, Seoul
13. Tang Y et al (1998) Offline recognition of Chinese handwriting by multifeature and multilevel classification. *IEEE Trans PAMI* 20(5):556–561
14. Liu HL, Ding X (2005) Handwritten character recognition using gradientfeature and quadratic classifier with multiple discrimination schemes. In: Proceedings of the ninth international conference on document analysis and recognition, Seoul

15. Liu CL, Nakagawa M (2001) Evaluation of prototype learning algorithms for nearest neighbor classifier in application to handwritten character recognition. *Pattern Recognit* 34(3): 601–615
16. Srihari SN, Hong T, Shi Z (1997) Cherry Blossom: a system for reading unconstrained handwritten page images. In: *Symposium on document image understanding technology (SDIUT)*, Washington, DC
17. LeCun Y, Bengio Y (1995) Convolutional neural network for images, speech, and time series. In: Arbib MA (ed) *The handbook of brain theory and neural networks*. MIT, Cambridge
18. LeCun Y, Bottou L, Bengio Y, Haffner P (1998) Gradient based learning applied to document recognition. *Proc. IEEE*, 86(11): 2278–2324
19. Wimmer Z, Dorizzi B (1999) Dictionary preselection in a neuro-Markovian word recognition system. In: *Proceedings of the 5th international conference on document analysis and recognition*, Bangalore
20. Procter S, Illingworth J, Mokhtarian F (2000) Cursive handwriting recognition using hidden Markov models and a lexicon-driven level building algorithm. *Proc IEE Vis Image Signal Process*, pp 332–339
21. Marti U, Bunke H (1999) A full English sentence database for off-line handwriting recognition. In: *Proceedings of the 5th international conference on document analysis and recognition*, Bangalore
22. Brakensiek A, Willett D, Rigoll G (2000) Unlimited vocabulary script recognition using character n-grams. In: *Proceedings of the 22nd DAGM symposium*, Kiel. Tagungsband Springer
23. Zhuang L, Bao T, Zhu XY (2004) A Chinese OCR spelling check approach based on statistical language models. In: *Proceedings of the IEEE international conference on system, man and cybernetics*, Hague
24. Tung CH, Lee HJ (1994) Increasing character recognition accuracy by detection and correction of erroneously identified characters. *Pattern Recognit* 27(9):1259–1266
25. Chang CH (1996) Simulated annealing clustering of Chinese words for contextual text recognition. *Pattern Recognit Lett* 17(1):57–66
26. Lee HJ, Tung CH (1997) A Language model based on semantically clustered words in a Chinese character recognition system. *Pattern Recognit* 30(8):1339–1346
27. Wong PK, Chan C (1999) Post-processing statistical language models for a handwritten Chinese character recognizer. *IEEE Trans Syst Man Cybern Part B Cybern* 29(2):286–291
28. Martin S, Liermann J, Ney H (1998) Algorithms for bigram and trigram word clustering. *Speech Commun* 24:9–37
29. Li YX, Tan CL, Ding X (2005) A hybrid post-processing system for offline handwritten Chinese script recognition. *Pattern Anal Appl* 8:272–286
30. Chaudhuri BB, Pal U (1997) An OCR system to read two Indian language scripts: Bangla and Devanagari. In: *Proceedings of the 4th international conference on document analysis and recognition*, Ulm
31. Chaudhuri BB, Pal U (1997) Skew angle detection of digitized Indian script documents. *IEEE Trans PAMI* 19(2):182–186
32. Pal U, Mitra M, Chaudhuri BB (2001) Multi-skew detection of Indian script documents. In: *Proceedings of the 6th conference on document analysis recognition*, Seattle
33. Agrawal P, Hanmandlu M, Lall B (2009) Coarse classification of handwritten Hindi characters. *Int J Adv Sci Technol* 10:43–54
34. Hanmandlu M, Agrawal P, Lall B (2009) Segmentation of handwritten Hindi text: a structural approach. *Int J Comput Process Lang* 22(1):1–20
35. Shaw B, Parui SK, Shridhar M (2008) A segmentation based approach to offline handwritten Devanagari word recognition. In: *Proceedings of the IEEE international conference information technology*, Tampa, FL
36. Chaudhuri BB, Bera S (2009) Handwritten text line identification in Indian scripts. In: *Proceedings of the 10th international conference on document analysis and recognition*, Barcelona

37. Roy P, Pal U, Lladós J (2008) Morphology based handwritten line segmentation using foreground and background information. In: Proceedings of the international conference on frontiers in handwriting recognition, Montréal, pp 241–246
38. Leedham G et al (2002) Separating text and background in degraded document images – a comparison of global threshholding techniques for multi-stage threshholding. In: Proceedings of the eighth international workshop on frontiers in handwriting recognition (IWFHR'02), Niagara-on-the-Lake. IEEE Computer Society, p 244
39. Shi Z, Setlur S, Govindaraju V (2004) Digital enhancement of palm leaf manuscript images using normalization techniques. In: 5th international conference on knowledge based computer systems, Hyderabad
40. Setlur S et al (2003) Creation of data resources and design of an evaluation test bed for Devanagari script recognition. In: Proceedings of the 13th international workshop research issues data engineering: multi-lingual information management (RIDE-MLIM), Hyderabad
41. Baker P et al (2004) Corpus linguistics and South Asian languages: corpus creation and tool development. *Lit Linguist Comput* 19(4):509–524
42. Kompalli S, Setlur S, Govindaraju V (2009) Devanagari OCR using a recognition driven segmentation framework and stochastic language models. *IJDAR* 12:123–138
43. Umesh RS, Pati PB, Ramakrishnan AG (2009) Design of a bilingual Kannada-English OCR. In: Govindaraju V, Setlur S (eds) Guide to OCR for Indic scripts. Springer, London, pp 97–124
44. Amengual JC, Vidal E (1998) Efficient error-correcting Viterbi Parsing. *IEEE Trans PAMI* 20(10):1109–1116
45. Liu CL, Jaeger S, Nakagawa M (2004) Online recognition of Chinese characters: the state-of-the-art. *IEEE Trans PAMI* 26(2):198–213
46. Liu C-L et al (2013) Online and offline handwritten Chinese character recognition: benchmarking on new databases. *Pattern Recognit* 46(1):155–162
47. Govindaraju V, Setlur S (ed) (2009) Guide to OCR for Indic scripts. Springer, London, 325p
48. Jayadevan R et al (2011) Offline recognition of Devangari script: a survey. *IEEE Trans Syst Man Cybern* 41(6):782–796

## Further Reading

This chapter provides an overview of the key issues pertaining to the recognition of Asian scripts. An introduction to the history of recognition of Chinese characters can be found in [2]. A benchmarking of effective techniques for offline and online Chinese OCR can be found in [45] and [46]. An overview of the research issues in Indic OCR (offline as well as online) and information retrieval can be found in [47]. All major research groups working in this area are represented in this book, which is divided into sections on recognition of Indic scripts and retrieval of Indic documents. A recent overview of Devanagari script recognition can be found in [48].

---

## Part D

# Processing of Non-textual Information

Until now, we have concentrated on documents conveying information through text. But if we come back to the general definition of a document we gave at the beginning of ►Part A (Introduction, Background, Fundamentals), it is evident that people can also use other kinds of representations to record facts and information for the purpose of communicating them in a legible way to other people. ►Part D (Processing of Non-textual Information) therefore extends the scope we have had so far of processing and recognizing images of textual documents to other kinds of documents currently printed on paper, archived, scanned, and used in human communication.

Of course, we could have addressed the general problem of documents containing pictures, paintings, photographs, and so on. But although a picture is worth a thousand words, it is still of another class. Documents can of course contain pictures, and a lot of efforts are devoted to natural image analysis, to retrieving information from photographs, to retrieving the right picture from a large set – the well-known problem of content-based retrieval. But covering all these techniques, questions, problems, and achievements would probably need one or two thousand pages more and is actually better documented in other handbooks. We therefore will basically restrict ourselves to our view of a document as a man-made symbolic representation of facts, and in this part we will address the specific questions arising from the analysis of documents where the information is only or mainly conveyed by *graphical representations*.

Although basic techniques for document image processing or page segmentation are still of great use for graphics-rich documents, they lead to specific segmentation problems, the two most prominent being the separation between text and graphics, and the conversion of graphics from the raster representation provided by digital imaging to a vector representation more suitable for many analysis and recognition tasks. These specific techniques are presented by Josep Lladós and Marçal Rusiñol in ►Chap. 15 (Graphics Recognition Techniques).

We have already studied in great detail the methods used for recognizing characters. In graphics-rich documents, the “alphabet” can be much larger, as hundreds of different *symbols* may be used. Salvatore Tabbone and Oriol Ramos Terrades

give us an overview, in ►Chap. 16 (An Overview of Symbol Recognition), of the main differences between character recognition and graphical symbol recognition. In addition to the large number of possible symbols, some of them being very close to each other, specific problems arise from the varying size of the symbols and from the fact that they are often intimately connected to lines and other graphics, which leads to the famous dilemma that you need to segment to correctly recognize, but you need to recognize to correctly segment.

But it is not enough to master basic segmentation and symbol recognition methods. Graphical documents rely heavily on a *context*; an engineering drawing is meant to be understandable by engineers and technical workers within the field it belongs to; a map relays information about features, locations, positions of various items, and is only understandable in close connection to an explicit or an implicit legend. In ►Chap. 17 (Analysis and Interpretation of Graphical Documents), Bart Lamiroy and Jean-Marc Ogier lead us through the numerous issues raised by the need to take this context into account in graphical document analysis systems.

The three next chapters go into the details of state-of-the-art solutions for three classes of problems which arise very often, where the graphical part of textual documents must absolutely be analyzed and understood for the recognition process to be successful as a whole.

Anastasios Kesidis and Dimosthenis Karatzas explore in ►Chap. 18 (Logo and Trademark Recognition) the important question of *logo and trademark recognition*; typically, this is a graphics recognition problem which often comes up in the midst of plain textual document analysis, as a logo or a trademark can be present in many documents, and it is necessary to correctly recognize them.

Tables and forms are also a very common way to organize information in plain documents. Again, although they are made of a lot of text, they also have a graphical framework whose analysis is vital for correct recognition and understanding of the document. Bertrand Coüasnon and Aurélie Lemaitre address this issue in ►Chap. 19 (Recognition of Tables and Forms).

Finally, in ►Chap. 20 (Processing Mathematical Notation), Dorothea Blostein and Richard Zanibbi explore the intricate question of *mathematical notation*, which at first glance appears to be close to impossible, as math can be very complex. But luckily, they also obey a strict and well-mastered syntax, so that math recognition systems become a very good illustration of the power of using the context expressed as syntactical rules to decipher complex formulas.

---

# Graphics Recognition Techniques

# 15

Josep Lladós and Marçal Rusiñol

## Contents

Introduction.....	490
History and Importance.....	491
Evolution of the Problem.....	491
Applications.....	492
Main Difficulties.....	493
Summary of the State of the Art.....	494
Outline of the Chapter.....	494
Text Recognition in Graphical Documents.....	495
Text-Graphics Separation.....	495
The Problem of Text Touching Graphics.....	503
Raster-to-Vector Conversion.....	504
Finding the Lines.....	505
Polygonal Approximation.....	510
Post-processing and Contextual Information.....	511
Arcs and Other Primitives.....	511
Recognition of Dimensions in Technical Drawings.....	512
Extraction of Texture-Based Graphical Primitives.....	513
Executive Summary.....	514
Conclusion.....	516
Cross-References.....	517
Notes.....	517
References.....	517
Further Reading.....	521

---

J. Lladós (✉) • M. Rusiñol

Computer Vision Center & Computer Science Department, Universitat Autònoma de Barcelona,  
Bellaterra, Spain

e-mail: [josep@cvc.uab.es](mailto:josep@cvc.uab.es); [marcal@cvc.uab.es](mailto:marcal@cvc.uab.es)

**Abstract**

This chapter describes the most relevant approaches for the analysis of graphical documents. The graphics recognition pipeline can be splitted into three tasks. The low level or lexical task extracts the basic units composing the document. The syntactic level is focused on the structure, i.e., how graphical entities are constructed, and involves the location and classification of the symbols present in the document. The third level is a functional or semantic level, i.e., it models what the graphical symbols do and what they mean in the context where they appear. This chapter covers the lexical level, while the next two chapters are devoted to the syntactic and semantic level, respectively. The main problems reviewed in this chapter are raster-to-vector conversion (vectorization algorithms) and the separation of text and graphics components. The research and industrial communities have provided standard methods achieving reasonable performance levels. Hence, graphics recognition techniques can be considered to be in a mature state from a scientific point of view. Additionally this chapter provides insights on some related problems, namely, the extraction and recognition of dimensions in engineering drawings, and the recognition of hatched and tiled patterns. Both problems are usually associated, even integrated, in the vectorization process.

**Keywords**

Dimension recognition • Graphics recognition • Graphic-rich documents • Polygonal approximation • Raster-to-vector conversion • Texture-based primitive extraction • Text-graphics separation

---

## Introduction

Graphical documents are instances of graphical (visual) languages, i.e., valid expressions of a diagrammatic notation. Graphical languages consist of terms, i.e., lexical units, and relational rules involving a certain type of grammar. The most common terms extracted from graphical documents are textual labels, lines and arcs, loops, solid regions, dotted lines, and hatched patterns. Relational rules define valid bidimensional relations between terms depending on the domain. Combined with domain-dependent semantics, terms and associated rules are interpreted giving rise to valid visual concepts. Most of graphic documents consist in line drawing structures. Because of that, primitive extraction in graphics recognition usually involves a vectorization process. Taking vectorization as the core problem, this chapter will be focused on the specific techniques to recognize graphical terms, namely, text, vectorial primitives, and repetitive patterns. These units are the basis of the subsequent tasks of the graphics recognition pipeline: symbol recognition and graphical document interpretation, studied in the next chapters ►Chaps. 16 (An Overview of Symbol Recognition) and ►17 (Analysis and Interpretation of Graphical Documents).

## History and Importance

The recognition of technical documents of different areas of engineering (mechanical, electrical, civil, etc.) became very popular in the 1990s. Document analysis had been mainly addressed the text recognition problem. Optical character recognition was one of the typical pattern recognition problems. But there was a growing industrial need to convert raster images of scanned drawings and maps to formats compatible with Computer-Aided Design (CAD) software and Geographic Information Systems (GIS). With the advent of software platforms for creating and editing such types of documents, the collateral challenge was the ingest of the existing large collections of engineering drawings and maps in paper format. The need was to convert these documents to a format compatible with such platforms for a subsequent edition. Hence, vectorization was the seed of an emerging graphics recognition activity that for decades has been, along with text recognition and layout analysis, the third leg of the document analysis field.

In addition to line primitives, graphical documents also contain textual labels (e.g., dimension labels or toponymy). Such labels are present mixed to graphics, in different orientation and font sizes. Classical OCR approaches are not able to recognize them. It leads to refer to the problem of text-graphics separation.

Methodologically, vectorization and text-graphics separation are considered two different problems; however, from an applied point of view, both can be considered as complementary in the task of converting technical drawings to a CAD or GIS format. As indicated above, their origin came from an industrial need, and a research community grew in the 1990s around it. Nowadays, many mature commercial applications exist and the interest of the scientific community has evolved to focus on problems related to the recognition and interpretation of graphical documents. The problem of text-graphics separation has nowadays a certain resurgence. In business document applications (e.g., processing of forms and bank checks), it is still a need to separate printed or handwritten text from rule lines.

## Evolution of the Problem

Many solutions have been proposed, either from the academia or the industry, and it seems to be an already mature problem. The evolution, in terms of performance, has been slow. Vectorization improvements have consisted in approaches that obtain more accurate approximations of lines with the minimum number of primitives. The problem of text-graphics separation has evolved improving the difficulty of text touching graphics and with methods more invariant to scale and orientation. In both cases, the major concern has been to get rid of the sensitiveness to a number of parameters.

Table 15.1 shows an interesting observation on the activity of the research evolution on graphics recognition techniques. It shows the percentage of papers presented for different topics in the graphics recognition workshop (GREC) series

**Table 15.1** Evolution of the graphics recognition topics in GREC workshop series

	GREC95 (%)	GREC97 (%)	GREC99 (%)	GREC01 (%)	GREC03 (%)	GREC05 (%)	GREC07 (%)	GREC09 (%)
Low-level processing	16	7	0	10	6	6	0	0
Vectorization and text extraction	16	13	10	16	6	19	17	14
Technical drawings maps	21	30	29	19	18	0	8	3
Layout analysis and diagrammatic notations	16	13	6	13	3	8	3	8
Applications, systems, and architectures	0	13	10	13	12	6	0	3
Symbol and shape recognition	11	13	23	6	18	25	14	14
Retrieval, indexing, and spotting	5	0	6	10	15	11	14	11
Sketching interfaces	0	0	3	10	18	8	11	8
Performance evaluation	16	10	13	3	6	6	17	8
Historical documents	0	0	0	0	0	11	14	11

over the years.<sup>1</sup> The activity on graphics recognition techniques is quite stable. Although in general the major challenges of the graphics recognition community have evolved, there is still room for improvements and applied research on basic techniques.

## Applications

The graphics recognition techniques described in this chapter are applied to technical diagrams belonging to different engineering disciplines. The most relevant application domains are:

- **Electrical and logic circuit diagrams.** It is one of the early areas of graphics recognition. Electronic schematics have a standardized notation that makes the processing easier, with rectilinear structures representing wires, and loop-like structures corresponding to electrical components.
- **Engineering drawings.** They can be seen as the assembly of low-level primitives such as arcs and straight lines, dashed lines, crosshatched, and solid areas. The recognition of these low-level primitives combined with domain-dependent knowledge gives meaning to the document and allows them to be converted to a CAD format. Thus, a hatched area combined with a semicircle could represent a screw, but a similar hatched area in another part of the same document could represent a section of a solid part. The recognition of dimensioning annotation plays an important role in such documents.
- **Architectural drawings.** A growing number of methods have been proposed for recognizing architectural drawings for conversion to CAD environments and subsequent actions as edition or 3D visualization. A particular focus of attention has been made on hand-drawn sketches, where actions such as vectorization and text detection are more imperfect due to the natural deformation of strokes. As in engineering drawings, the recognition of dimensioning annotation is very frequent.
- **Maps.** Map-to-GIS conversion has been an active application field over the last decades. Maps are one of the most difficult types of technical diagrams. Different classes of maps exist. Cadastral maps consist of polygonal shapes and hatched patterns, representing parcels, with circumscribed text annotations. Utility maps consist of lines and small symbols composed of geometric basic primitives (squares, circles, arrowheads, etc.). Finally, geographic maps with line objects that usually represent isolines and roads and small symbols whose meaning is given by a legend. In this kind of maps, color information plays an important role in the segmentation. The detection of symbols is usually legend-driven, so text-graphics separation is very important to read the legend.

## Main Difficulties

Although stable and robust vectorization methods exist, it is not clear what vectorization should be or should output. Vectorization is not an isolated step and its accuracy depends on the goal of the whole system. In some cases, the best the primitives fit into lines, the better is vectorization; however, in other cases, the goal is the minimum number of primitives although the vectors roughly approximate the lines. Usually vectorization systems provide a set of parameters that have to be set by the user or customized in terms of document categories.

One of the worst difficulties of text-graphics separation algorithms is the case of text touching graphics. Although this configuration can be detected, in some cases the reconstruction of characters for the subsequent recognition is very difficult. As for vectorization, the avoidance of parameters, or their customization in terms of document properties, is another difficulty of text-graphics separation methods.

## Summary of the State of the Art

Many solutions have been proposed, either from academia or industry, for the graphics recognition techniques reviewed in this chapter. Vectorization, i.e., raster-to-vector conversion, is a mature problem; however, the community has not yet reached stable methods because of the lack of a consensus on what a vectorization system ought to output. Vectorization consists in a kind of polygonal approximation of the medial axis of line images [32, 36]. Skeleton-based approaches tend to displace the junction points and to be sensitive to image irregularities. Another family of methods is based on line following or line contour matching [24]. These methods are more precise in junction finding but their performance decreases in complex images. Sparse-pixels approaches are considered to be a third family of methods [22]. These approaches do not analyze all the image pixels but detect vectors by analyzing key points in terms of local neighboring configuration. Vectorization approaches often involve the detection of other primitives than straight segments, but curve ones, in particular circular arcs [21].

Another classical problem in graphics recognition is text-graphics separation. Graphical documents often contain text annotations (names of streets in maps, dimensions in engineering drawings, beat in musical scores) that must be segmented before being recognized by an OCR module. Text can be segmented looking for small connected components following a regular path. However, text is sometimes connected to graphics. A baseline reference to solve it is the algorithm of Fletcher and Kasturi [25] and subsequent improvements such as Tombre et al. [68]. Other classical methods work at a multiresolution representation [64] which allows to detect text of different sizes and orientations. A recent approach proposed by Hoang and Tabbone [29] consists in an elegant formulation inspired by techniques from the signal processing field.

## Outline of the Chapter

This chapter addresses the different tasks of graphics recognition that are applied when analyzing line drawings. First in section “[Text Recognition in Graphical Documents](#),” the main algorithms for separating text from graphics are described. Although there is no standard pipeline, text-graphics separation is usually performed first, so afterwards a raster-to-vector process is applied to the graphical entities, and an OCR to the textual ones. In section “[Raster-to-vector Conversion](#),” vectorization algorithms are reviewed. Section “[Recognition of Dimensions in Technical Drawings](#)” provides a short overview on the recognition of dimensions, a classical entity appearing in engineering drawings. Section “[Extraction of Texture-Based Graphical Primitives](#)” is addressed to review the main approaches for the recognition of textured patterns (one-dimensional and bidimensional) such as dotted lines, or hatched areas. Finally, section “[Conclusion](#)” draws the conclusions and perspectives.

## Text Recognition in Graphical Documents

Graphical documents usually contain text labels (e.g., dimensions in mechanical drawings, place or river names in maps, room names in architectural drawings, labels in flowcharts). This kind of textual terms cannot be recognized following classical OCR approaches that assume a regular layout organized in columns, paragraphs, and lines (Manhattan layout). Common problems of text analysis in graphical documents are touching to graphics, multioriented or even following curvilinear paths, reduced lexicons, etc. In this section, the state-of-the-art algorithms for detecting text strings in graphical documents will be described.

### Text-Graphics Separation

Graphical documents at large rely on a diagrammatic notation dependent of the domain. Each individual entity such as an arc, a dashed line, a hatched area, or a symbol, has a meaning in a given context and according to a syntax. Text labels are used to give attributes to the graphical entities. For example, a numerical string in an engineering drawing or an architectural plan informs the reader about the dimension of an element or a text label in an isoline of a map informs about the altitude in the geographical location represented by the isoline. A key step in the conversion or raster images to CAD formats requires the segmentation and recognition of text labels, and their interpretation jointly with the graphical parts.

The text-graphics separation process aims at segmenting the document into two layers: a layer assumed to contain text characters and annotations and a layer containing graphical objects. As introduced before, the recognition of text strings may be very relevant for the interpretation of the graphical artifacts. Most approaches use to perform text separation at early stages in the processing pipeline, usually using image processing techniques and limited knowledge about the configuration of the image in terms of higher-level objects.

As Lu suggests in [46], the differentiation between text and graphics in a graphical document is sometimes subjective if only the local distribution of pixels is considered, without taking into account the context. An isolated small line can be part of an “I” character but also part of a dashed line, the end of a dimension component, etc. Although there is some overlapping due to the mentioned higher-level interpretation, a common definition of both categories may be stated as follows:

- **Text:** symbols or strings for the interpretation of the document parts, including letters, words, digits, and/or special symbols
- **Graphics:** non-textual components having a domain-dependent meaning according to a diagrammatic notation, including all kinds of lines, curves, solid, or textured areas

The use of low-level information for discriminating text in graphical documents requires the consideration of a number of geometric and topological properties at

pixel or region level. These properties will drive the heuristics for the process. According to Lu [46], in technical drawings, some geometric features that differ between textual and graphical components can be observed:

- The size of text characters is often much smaller than that of graphics components. Changes in text size or shape are within a narrow range.
- Text characters usually appear in the form of short strings, having uniform size, inter-character distance and a “smooth path” (usually rectilinear and oriented horizontally, vertically, or slanted at an angle of 45°).
- The local stroke density of text regions is often much higher than that of graphics.
- The length of the linear components included in strokes of text strings is much shorter than that of graphics.

It is quite straightforward to implement algorithms that filter the image parts (connected components or segments) according to the above features. Figure 15.1 illustrates the expected output of this process. However, the detection of text components in graphical documents has a number of difficulties that require more sophisticated steps. The main difficulties are the following:

- Graphical components include lines of any length, thickness, orientation, and pattern (continuous, dashed, dotted). Closed curves of different order (circles, ellipses, etc.) or polylines can be unfilled, filled with solid areas, hatched, tiled.
- Text and graphic parts appear mixed, sometimes touching or overlapped. In some cases, there is no clear geometric difference between them (e.g., small components of dotted or dashed lines can be confused with characters or punctuation symbols).
- Text can be of any font, size, and orientation. The problem of multioriented text is an important difficulty that makes unusable traditional OCR techniques. Engineering drawings use to present text strings in vertical, horizontal, or slanted rectilinear orientation; however, in other documents like maps, text may appear in curvilinear orientations.

Tombre et al. in [68] proposed three families of text-graphics separation methods. Recently, Hoang and Tabbone [29] slightly reformulated this classification. According to the previous works, the following taxonomy of text-graphics separation approaches is proposed:

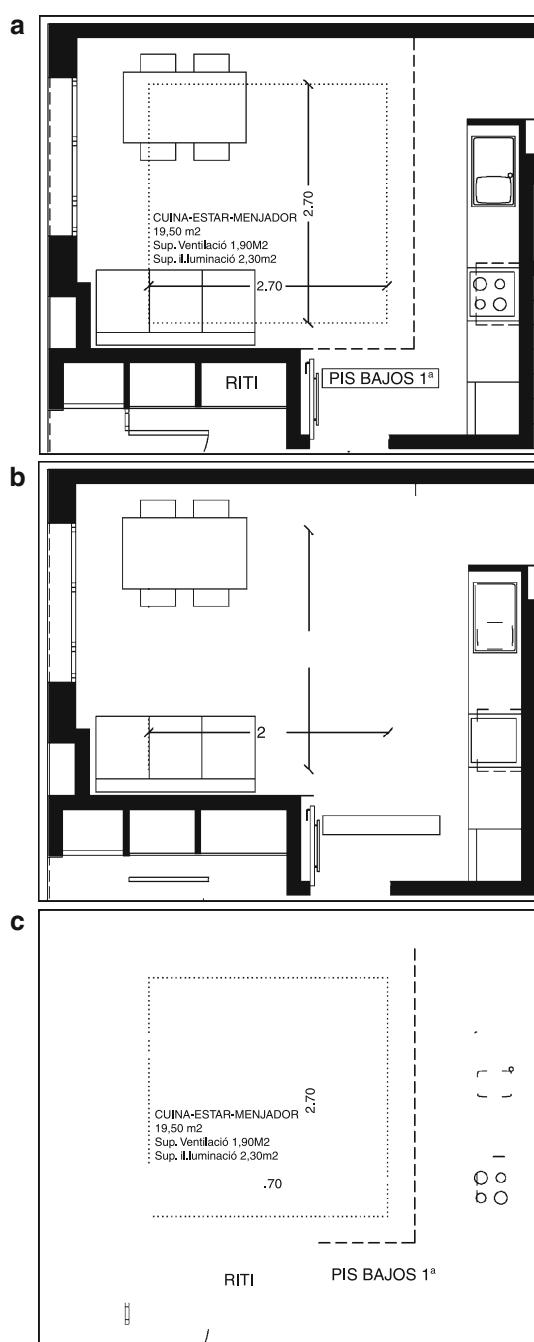
- Morphology analysis
- Connected component analysis
- Line (vector)-based segmentation
- Multiresolution analysis
- Signal processing
- Ad hoc methods (forms, music)

In the next subsections the different families will be further described.

## **Methods Based on Morphology Analysis**

A number of methods use morphological operators with the assumption of that the text is what remains after applying iterative openings to the original image with structuring elements designed to eliminate rectilinear objects. The method proposed by Wahl et al. [70] is one of the first methods based on morphological

**Fig. 15.1** Text-graphics separation example. (a) Original image, (b) graphical layer, (c) textual layer



filtering. It uses Run-Length Smoothing Algorithm (RLSA) to detect vertical and horizontal text strings. RLSA can be seen as morphological closing (or opening) operations with vertical and horizontal structuring elements of length according to the text size and graphical lines width. The method of Lu [46] uses RLSA too. The main improvement of this work is that it allows to detect slanted lines by performing a stretching operation at different angles. The main drawback of these approaches is that they tend to wrongly label text as graphics. RLSA has proven to be efficient in separating rule lines in forms, but its use in graphics documents is less frequent.

### **Methods Based on Connected Component Analysis**

This family of methods is one of the most commonly used. A pioneer and well-known work was proposed by Fletcher and Kasturi [25]. The basic idea is to segment text based on basic perceptual grouping properties. Thus, simple heuristics on font size, inter-character, word and line spacing, and alignment are used. The method requires many thresholds, but the good point is that they are extracted from the image object properties and are not manually set *a priori*. The main steps of this method are:

1. Connected component generation
2. Filtering of connected components based on area and size
3. Connected component grouping in terms of area and size to cluster those that are likely to belong to the same font size, so are candidate to be in the same string
4. Hough Transform applied to the centroids of connected components (text strings are supposed to have a rectilinear arrangement)
5. Logical grouping of strings into words and phrases. This step intends to capture those components kept aside by the Hough step, but that fall into the potential text area (in terms of interline spacing, intercharacter gaps, etc.). For example, a period at the end of a string or an accent
6. Text string separation

This method combines simplicity with good performance and scalability to different text properties. This is probably the reason that most of the methods are based on the Fletcher and Kasturi one, with small variations and adaptations to different contexts. The weakness of this method is that it does not cope with text touching graphics. Tombre et al. [68] proposed an improved approach able to separate text touching to graphical parts. In addition, they introduced some more heuristics allowing to improve the performance. Many commercial systems nowadays use approaches inspired by the ones described above. Figure 15.2 illustrates the main steps of the methods based on the grouping of connected components.

### **Methods Based on Vectorial Representations**

There are some approaches where the segmentation of text is performed at vector level, i.e., after the image is vectorized (see section “[Raster-to-Vector Conversion](#)”), instead of pixel level [17, 18, 35]. The main idea consists in recursively grouping short line segments in the vectorial image. Thus, after a continuous short segment is selected as a seed, touching segments are iteratively merged. This procedure allows to obtain a coarse detection of character bounding boxes that are then grouped in

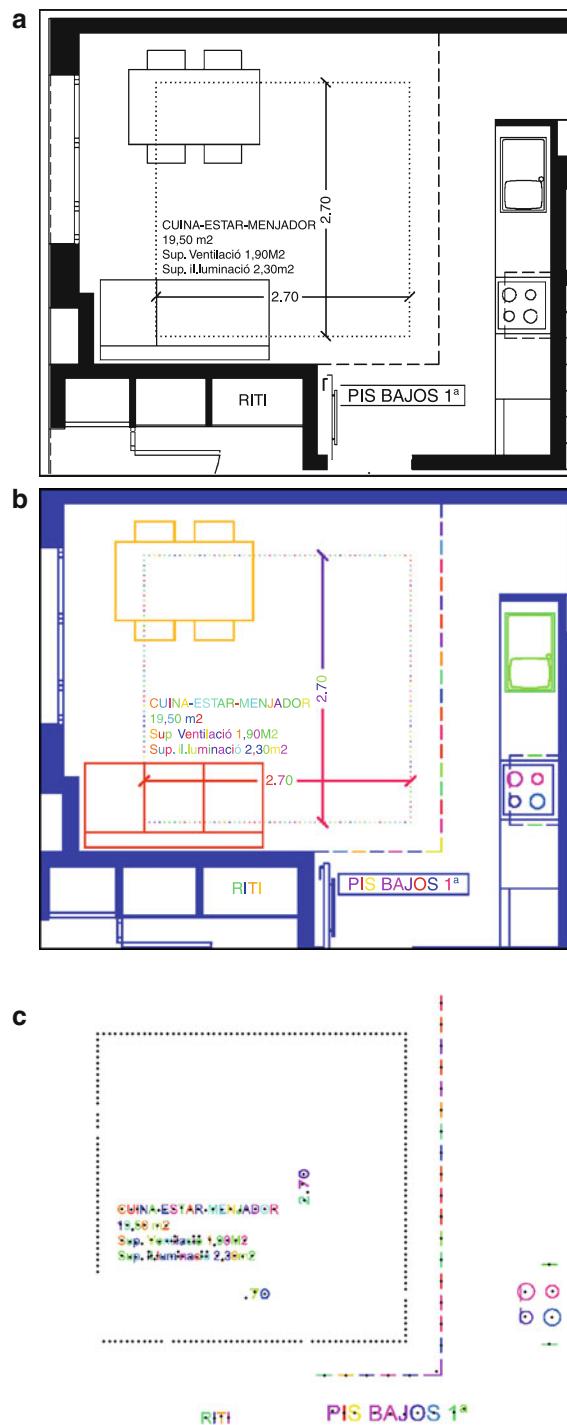
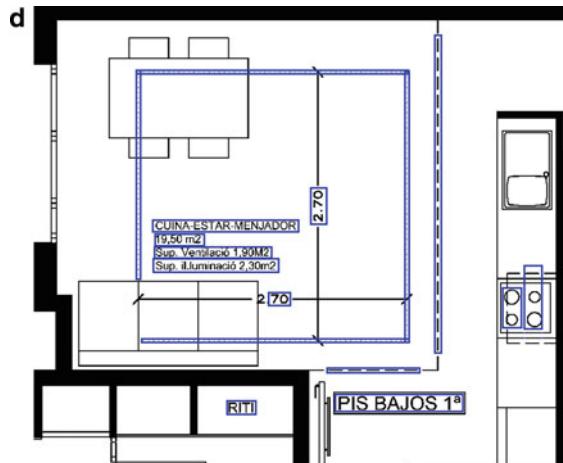


Fig. 15.2 (continued)



**Fig. 15.2** Text-graphics separation steps. (a) Original image, (b) connected components, (c) filtering of connected components, (d) result of string grouping

terms of their alignment. The basic process of this method does not differ from the methods based on connected component grouping. It is a bottom up process that instead of grouping pixels groups vectors.

### Methods Based on Multiresolution Representations

Multiresolution approaches are based on visual perception principles. Each resolution allows to highlight a specific category of information. Tan and Ng proposed a multiresolution approach in [64] based on the construction of a regular pyramid over the image. A pyramid representation is a well-known multi-scale signal representation in which an image is subject to repeated smoothing and subsampling. Historically, the pyramid representation is a predecessor to scale space representation and multiresolution analysis. The main assumption of this approach is that at a certain level of the image pyramid, a text line consists of a sequence of regularly placed components. However, the same text line at a higher (coarser) level looks like a long component. Hence, text is detected at the pyramid level where the granularity of the cells is “tuned” with the size of the text components. Loo and Tan in [45] proposed an improved approach based on irregular pyramids. The main advantage is that it does not require the detection of connected components. The main hypothesis of this approach is to view a text image as a combination of multiple irregular smaller regions. A word group is an aggregation of smaller regions holding fragments of a word and empty regions. The grouping process of lower-level regions into higher-level ones in the pyramid is determined by a concept of closeness. Thus, regions of the pyramid are merged in the next level if they contain word fragments with uniformity properties in area, mass, and density. The irregular pyramid structure contains the layout information of the entire document such that layout analysis is straightforward by navigating the pyramid levels. An advantage of this algorithm is that it is able to extract word groups with varying sizes, fonts,

alignments, and orientations. However, when text and graphics components touch, this approach induces wrong detection results.

### Methods Based on Signal Processing

Recently, Hoang and Tabbone [29] proposed a novel approach inspired in the techniques used in the signal processing field. In this approach, the image is seen as a bidimensional signal built as a mixture of two separated bidimensional signals of the same size (text and graphics) but containing morphologically different features. They propose a text-graphics separation method based on sparse representations. Sparse representations are compact representations that account for most or all information of a signal with a linear combination of a small number of elementary signals called atoms. Popular transforms are the discrete Fourier transform, the wavelet transform, and the singular value decomposition. The general problem of finding a representation with the smallest number of atoms from an arbitrary dictionary has been shown to be NP-hard.

In the approach presented in [29], the two signals, namely, text and non-text, are represented in terms of two discriminative dictionaries based on undecimated wavelet transform and curvelet transform. Morphological component analysis is employed for the separation of sparse representation of text and graphics components in these two dictionaries. Once text image is obtained, the final text string localization step follows a classical approach of connected component grouping in terms of heuristic spatial properties. The main advantage of this method is that it overcomes the problem of text touching graphics, showing an improved performance regarding the classical state-of-the-art approaches.

There exist some methods in the literature that use texture segmentation techniques. The basic assumption underlying these approaches is that the text and the graphics parts are considered as two different textured regions. Hence, the segmentation is seen as a binary text/non-text classification problem. The analysis is usually done in the frequential domain using filters that have maximum responses at some frequencies corresponding to text. Examples are Gabor filters [31], wavelet transform [2], or directional filters [34]. These approaches usually assume that documents are structured in a Manhattan layout, where text is organized in columns, paragraphs, and lines, and graphics are contained in figures inserted in the text parts. Since this is a layout segmentation problem rather than text detection in graphical documents, the reader is referred to Chaps. ▶5 (Page Segmentation Techniques in Document Analysis) and ▶8 (Text Segmentation for Document Recognition) for further details.

### Ad hoc Methods

There are some types of documents like forms, tables, or musical scores that due to their particular layout have some text separation rules specifically designed. The graphical parts usually consist in horizontal or vertical lines.

Common methods under this category are those for ruling-lines separation from handwritten text in forms [1, 6] or staff removal in musical scores [10]. These methods assume that lines have an horizontal configuration. The detection is

performed with projection profiles, RLSA, or line following algorithms. For further details, the reader is referred to Chaps. ►19 (Recognition of Tables and Forms) and ►22 (Analysis and Recognition of Music Scores).

## Summary

In the above subsections, the most relevant methods for the segmentation of text strings in graphical documents have been reviewed. A comparative summary of the different categories in terms of the main desired features is provided in Table 15.2. The categories of methods have been evaluated depending on the ability of segmenting text touching the graphical parts, if they are able to cope with different fonts and sizes, and the ability to detect multioriented text, even if it is not in a rectilinear layout.

The segmentation of text touching graphics is one of the main difficulties. In this classification, a method meets this feature if it is able to discriminate the text component at pixel level and is also able to reconstruct the segmented strokes. The baseline method based on grouping connected components [25] does not solve this problem. However, the improvement proposed by Tombre et al. [68] overcomes this problem proposing a method inspired by the vector-based approaches. These approaches achieve the best performances. The reason is that, since they work at skeleton level, they can reconstruct broken strokes from the bifurcation points of the skeleton. Since it is a problem that in the literature is addressed specifically, the solutions are further described in the next subsection.

The segmentation invariant to multiple fonts and sizes is solved working at different scales. It is implicitly done in methods that analyze pyramids of spatial bins at different scales (multiresolution methods) or methods that model the types of components in terms of frequencies (signal processing). The rest of the methods are able to detect text of different sizes but involving some thresholds defining the size of the structuring element, or the connected components or vectors that are grouped. Such thresholds are not necessarily set manually, but they can be inferred from the properties of the image being analyzed.

Concerning the property of multioriented text detection, the main difficulty is to cope with text that does not follow a rectilinear orientation but a curvilinear one. Typical examples are text strings appearing in maps that follow the orientation of an isoline, a river, or a road. Most of the methods cope with rectilinear orientations by ad hoc procedures like repeating the process at different orientations

**Table 15.2** Strong and weak points of text-graphics separation approaches

	Text touching graphics	Multiple fonts and sizes	Multioriented text
Morphology analysis	—	+	-/+
Connected components	+	+	-/+
Vector-based segmentation	++	+	-/+
Multiresolution analysis	—	++	+
Signal processing	+	++	+

(morphology analysis) or using Hough-based methods when grouping components. The methods based on multiresolution or frequential analysis achieve in general better performance; however, they do not really define text strings as output artifacts. Roy et al. [55] recently proposed an approach addressing the problem of segmenting multioriented curvilinear text strings in graphical documents.

## The Problem of Text Touching Graphics

As stated in section “[Text-Graphics Separation](#),” one of the main difficulties of the separation of text and graphics is when text components are connected to graphical ones. Although some of the methods reviewed in the previous section can deal with text characters which touch the graphics, a better performance is achieved with specific methods. Such methods usually separate text from graphics by detecting touching lines.

When there is a priori knowledge about the structure of the graphical parts like long vertical and horizontal lines in forms, or the width of the lines, it can be modeled and ad hoc graphics detectors can be used. Otherwise, there are two main assumptions which are used. First, text characters connected to the graphics use to generate a local configuration of small strokes connected to a long line. Second, it is assumed that some isolated neighboring characters have already been detected and define a certain context.

Cao and Tan [5] base their method on the observation that the constituent strokes of characters are usually short segments in comparison with those of graphics. They work on the skeletons of large connected components. Using the properties of smooth continuation and line width, small strokes are grouped to reconstruct long segments underlying the region of intersection. Afterwards, the rest of the strokes connected to the reconstructed long line are grouped in candidate characters depending on their proximity and adjacency. Finally, character strings are extracted by merging connected components of a certain size likely to be isolated characters and the characters connected to graphics. Tombre et al. [68] proposed a similar approach. The main difference is that they assume that isolated characters have been previously segmented and grouped into strings. For each string, continuation zones are defined as prolongation of its ends. Then potential characters touching graphics are searched grouping connected short skeleton strokes included in these regions.

Song et al. [62] as part of a raster-to-vector conversion system proposed an improvement of the method of Dori and Liu [20]. The vectorization approach progressively simplifies the raster image by erasing recognized graphic objects to eliminate their interference with subsequent recognition. This iterative simplification allows to implicitly identify intersections between small vectors belonging to text characters and the erased graphical parts and gather nearby vectors in subsequent steps.

In the approach of Lu [46] mentioned in section “[Methods Based on Morphology Analysis](#),” the problem of text touching graphics is solved by an opening

morphological operation once text strings are detected in a given direction. It eliminates possibly existing connections but can provoke a side effect of breaking down connected components corresponding to single characters. A similar approach based on directional morphological filtering was proposed by Luo and Kasturi [47].

---

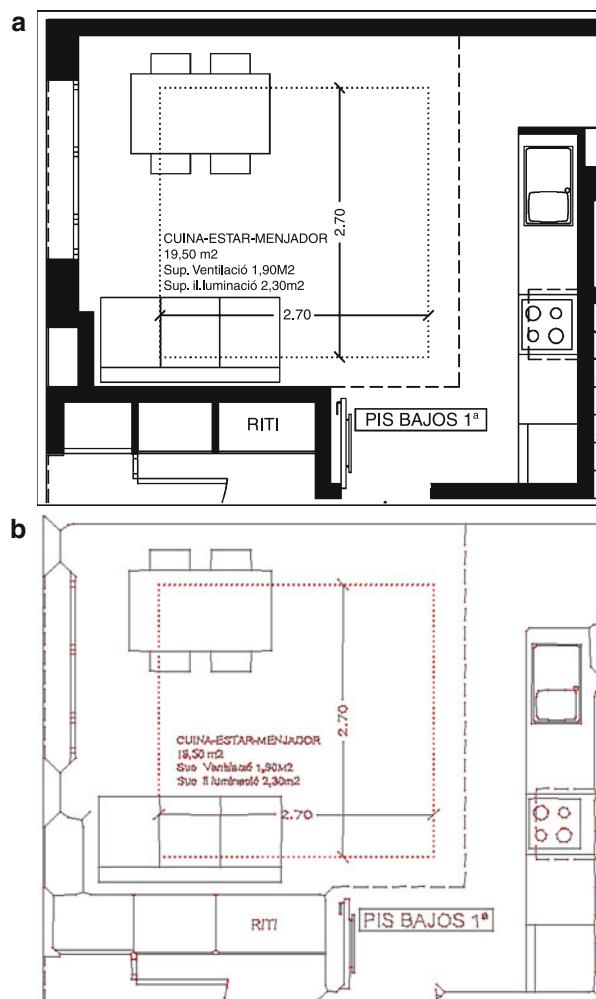
## Raster-to-Vector Conversion

Vectorization, also known as raster-to-vector conversion, has been one of the most studied problems since the beginnings of graphics recognition. The vectorization process can be defined as the automated conversion of scanned document images, having a pixel representation, into a set of vectors and other simple geometrical primitives. An example of the result of a raster-to-vector conversion algorithm is shown in Fig. 15.3. By representing graphic-rich documents, such as line drawings, in vectorial format its edition is easier and the storage much more efficient. In addition, vectorial data can be more suitable for further analysis than working directly with the raw pixel values in certain domains. Within the graphics recognition field, vectorization is a mature research topic. Many vectorization methods have been developed and implemented throughout the years and a number of software packages are available. Most of the methods provide good enough results to be used as input data for higher-level recognition and analysis methods. In that sense, raster-to-vector conversion seems to be an already solved problem. There are very few new scientific contributions to this specific topic, as vectorization methods have reached maturity and are assimilated by researchers as state of the art. However, all the methods present their specific weaknesses and, as stated by Tombre in [65], one cannot say that perfect raster-to-vector conversion is available. Several surveys that review the existing methods, presenting the strong and weak points of each of them, can be found in the literature, for example, Doermann's overview on the vectorization and segmentation problems presented in [14], Wenyin and Dori's survey of non-thinning vectorization methods [73], or Tombre et al.'s discussion on which are the most stable and robust vectorization methods presented in [67].

Focusing only on line features, the vectorization processes can be decomposed in three consecutive steps. The first step is to find the straight lines in the original raster image. The next step is to approximate these found lines into a set of vectors. After this approximation, a post-processing step is often required in order to merge some vectors, find more accurately the junction positions, etc. Each of these three steps will be overviewed in sections “[Finding the Lines](#),” “[Polygonal Approximation](#),” and “[Post-processing and Contextual Information](#),” respectively.

However, in order to be useful for higher-level interpretation phases, a raster-to-vector process should not be limited to the recognition of straight lines and should be able to deliver other geometric primitives. One of the most prolific research areas

**Fig. 15.3** Raster-to-vector conversion example.  
**(a)** Original drawing.  
**(b)** vectorial representation



within the vectorization problem is the circular arc detection. The state of the art in arc detection will be overviewed in section “[Arcs and Other Primitives](#).”

## Finding the Lines

The first step in any raster-to-vector scheme is to extract a set of lines, i.e., chain of pixels, from the raster image to be processed.

Historically, the state-of-the-art methods have been categorized in several families depending on how they perform the line finding step. Roughly one can summarize these families into the following taxonomy:

- **Skeleton-based approaches** encode the shapes' topology by reducing them to their skeleton.
- **Contour matching methods** extract the vectors by tracking pairs of points from parallel edges.
- **Line-fitting methods** use a parametric line model to detect the lines that are present in the image.
- **Sparse-pixel approaches** avoid having to examine all the pixels in the image by using subsampling methods that give a broader view of the line.

### **Skeleton-Based Methods**

Computing the skeleton of the shapes from the raster image is the most common approach for vectorization [32, 36].

There are two well-known paradigms for extracting the skeleton from an image that are commonly used for vectorization purposes. The first one is based on an iterative thinning of the original image by a boundary erosion process [40]. The iterative process removes pixels until a unit-wide pixel chain remains. It is usually known as using a “peeling an onion” approach. On the other hand, the skeleton of an image can be computed by identifying the ridge lines formed by the centers of all maximal disks included in the original shape. For this second approach, usually some distance transform [51] is used for efficiency. As studied by Tombre and Tabbone in [66], iterative approaches need multiple passes before reaching the final result, so that computational time might be too high. In addition, they are sensitive to noise. On the other hand, distance-based approaches provide their results efficiently (only two passes are needed) and yield stable skeletons. It is therefore the preferred choice for the skeleton-based methods, used in many works as for instance in [28, 38].

Skeleton-based methods yield good precision regarding the positioning of the line. However, they tend to give lots of barbs when the image is irregular; therefore, post-processing steps to prune these barbs are needed. In addition, skeleton-based methods are weak with respect to the correct restitution of the junction and extremity points, compared with the position the draftsman wanted to be. This effect can be appreciated by looking at Fig. 15.4b. This drawback is due because skeletons follow the centers of maximal disks within the shape, whereas the correct junction or extremity position is not on those centers. Recovering the original line width is straightforward when using distance-based approaches, but it is not so when using iterative ones.

### **Pairwise Contour Matching Methods**

Contour matching approaches extract the vectors by tracking pairs of points from parallel edges. Once the contours are extracted from the raster image, parallel edges are matched together. From a pair of parallel contours, the medial line is generated and taken as the chain of pixels that represent the straight lines in the drawing. A final step that extends the medial lines until the intersections are found is performed in most of the cases. Several examples of raster-to-vector methods that follow this approach can be found in the literature [3, 24, 27].

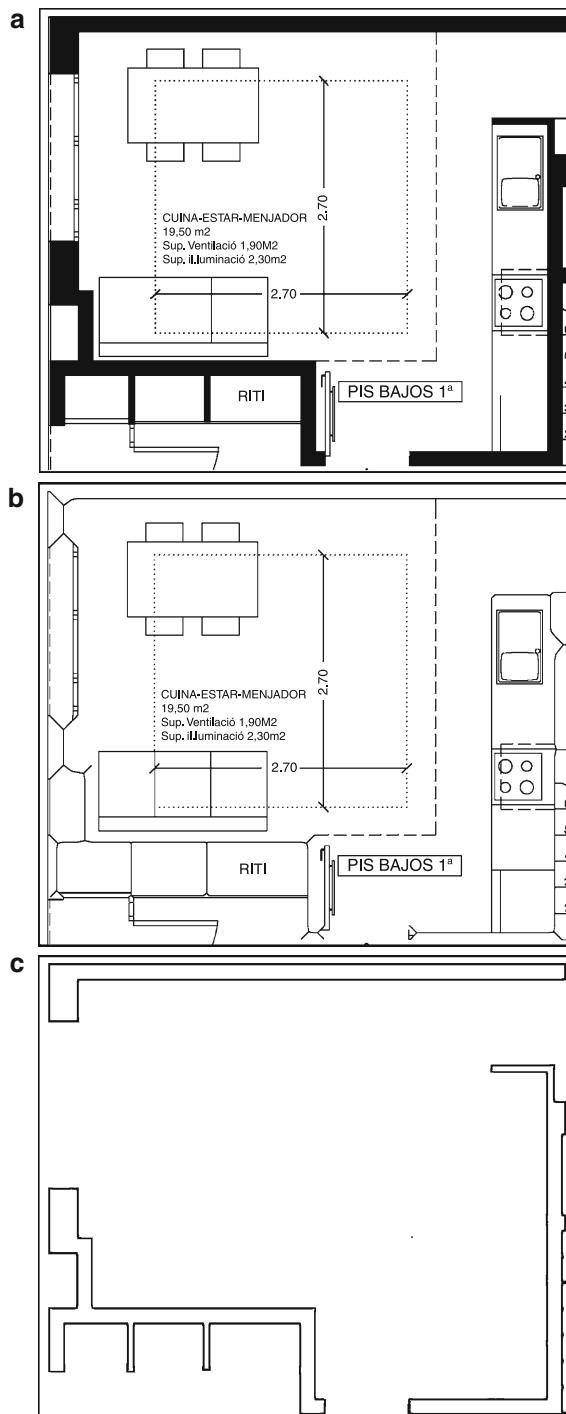
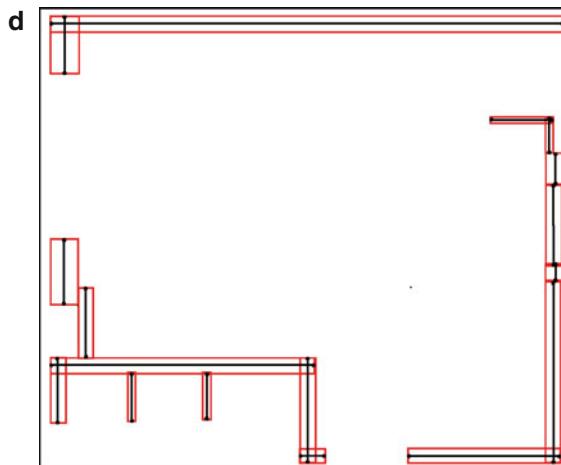


Fig. 15.4 (continued)



**Fig. 15.4** Line finding examples. (a) Original image, (b) its skeleton, (c) contours of thick elements, (d) result of the pairwise contour matching

The methods based on matching parallel contours are better at positioning the junction points than the skeleton-based ones. In addition, it is simple to add a line width recovery step, in order that the lines in the vectorial result have the original widths. However, the main drawback of these approaches comes from the difficulty to handle one-to-many and many-to-many contour matches when the original drawing presents complex patterns. See example in Fig. 15.4c,d.

In order to get the best of both approaches, some authors as Hori and Tani-gawa [30] or Shimotsuji et al. [60] propose hybrid methods that use a combination of the skeleton and the contours in order to find the lines from the raster images.

### Line-Fitting Approaches

Line-fitting approaches consist in using a parametric line model to detect the lines present in the image. The most common and well-known technique to do so is the Hough transform. The main advantage of such methods is that the polygonal approximation step is avoided since the Hough transform already delivers vectorial data. However, very few attempts using the Hough transform have been proposed. Some examples are the following works [16, 61]. The main disadvantage of Hough-based methods is its memory inefficiency. In addition, since the parameter space is discretely sampled, the localization accuracy might be hindered.

### Sparse-Pixel Approaches

A number of sparse-pixel approaches have also been proposed. The general idea is to avoid having to examine all the pixels in the image, by using appropriate subsampling methods which give a broader view of the line. Examples are the orthogonal zigzag method presented by Dori and Wenyin in [22], the sparse-pixel tracking approach by Wenyin and Dori in [72], or the mesh-based approaches by Lin et al. [44] and by Vaxivière and Tombre [69]. The sparse-pixel approaches are

time-efficient due to the sparse sampling of the image data; however, their main limitation is that they are prone to double detections.

### Other Approaches

Other approaches that do not fall into any of the above families can also be found in the state of the art.

Some algorithms base the line detection on a run-based encoding of the raster image and then an analysis step of such runs. Examples of this technique are the following works [4, 13, 50]. The main problem of such techniques is again the inaccuracy to detect junction points, the sensitivity to noisy data, and the dependence on the scanning direction.

Other line extraction techniques are based on directly tracking lines in the original image itself. For example, Fukada presented in [26] a method using a line sensor that follows a line by keeping a tracking window perpendicular to the line direction. Or, in [59], Shih and Kasturi present a method which encodes the image in terms of maximum black squares. The lines are then extracted by tracking the centers of aligned maximum squares.

### Summary

Summarizing, skeleton-based methods are good at positioning the lines with minimum displacement from the original (localization) but are sensitive to noise (false-positive lines are detected) and are weak with respect to the correct restitution of junction and extremity points.

Contour matching methods perform well at both localizing and detecting the correct pixel chains but rely on complex matching schemes that can derive in detecting two lines where there is only one (multiple response). They perform well at the junction positioning and are able to recover the original line width.

Sparse-pixel approaches work well regarding localization but tend to miss small details and are prone to double detections. They also tend to provide junction points where there are none.

Finally, Hough-based methods are too dependent on the sampling of the parameter space provoking line merging and localization errors. They perform nonetheless quite good at detection of true lines and junction points.

All these strengths and weaknesses for each of the line finding families described above are presented in Table 15.3 (inspired by the summary by Tombre et al. [67]).

**Table 15.3** Strong and weak points of line-finding approaches

	Skeleton	Contour	Sparse	Hough
Localization	++	+	+	+/-
Detection	--	+	+/-	+/-
Single response	++	-	-	-
Junctions	-	+	-	+
Width	+/-	+	+	-

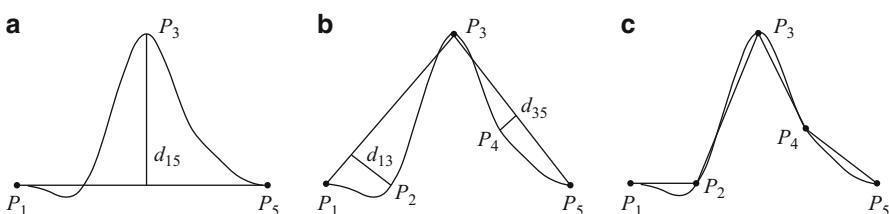
## Polygonal Approximation

The chain of pixels extracted from the previous step have to be represented by a set of vectors. Many methods exist to perform this polygonal approximation step. The interested reader is referred to Rosin's paper on benchmarking of polygonal approximation algorithms [53].

Among the most common approaches that are used for the specific purposes to build a raster-to-vector conversion system, the method proposed by Rosin and West [54] has been often the preferred one [23, 67].

The method by Rosin and West [54] is based on a recursive split-and-merge technique. The principle is to recursively split the pixel chain into smaller and smaller segments, until the maximum deviation is zero or there are three or less points left. When performing this splitting phase, a tree structure is built. Then, a merging step traverses the segment tree upwards by keeping those segments that maximize a measure of significance. This measure of significance is defined as the ratio between the maximum deviation and the segment's length. Rosin and West's method tends to split up the lines around junction points into many small segments but has the advantage that is nonparametric and thus very generic.

Following the example of Fig. 15.5, a list  $L_{ij}$  of skeleton pixels is hypothesized as being a straight line passing through its end points  $P_i$  and  $P_j$ , the point  $P_n$  corresponding at the point of maximum deviation  $d_{ij}$  to the straight line segments the list  $L_{ij}$  in two lists  $L_{in}$  and  $L_{nj}$ , and the process is repeated recursively on each of the two lists. The recursive process is stopped when a line segment is smaller than four pixels long or the deviation is less than three pixels. The result of the recursive process is a multilevel tree where the description of the list of skeleton pixels at each level is a finer approximation of the level above. The tree is then traversed back up to the root. At each level, if any of the line segments passed up from the lower level are more significant than the line segment at the current level, they are retained and passed up to the next higher level as candidate line segments. If this is not the case, the line segment at the current level is passed up. In Fig. 15.5, an example of the first steps of the algorithm in approximating a curve by a set of straight lines is presented.



**Fig. 15.5** Three levels of the straight line approximation. (a) First iteration; (b) second iteration; (c) third iteration

## Post-processing and Contextual Information

After the polygonal approximation, a post-processing step is often required in order to merge some vectors, prune small barbs, find more accurately the junction positions, etc. Since these low-level post-processing steps might enhance significantly the quality of the vector description, almost all the authors propose their own post-processing. Some examples can be found in [7, 28, 32].

When the documents to vectorize come from a very specific domain, it makes sense to propose ad hoc post-processing steps that model the domain-specific constraints. Until now the presented steps of line finding and polygonal approximation do not use any specific knowledge about how vectors are supposed to be in the original drawings. The addition of contextual knowledge favors a given method for a specific application area whereas it hinders its generality. Some examples are the work of Chhabra et al. [8] aimed to process drawings from a telephone company, the system from Dosch et al. [23] processing architectural drawings, or the work of Lee et al. [42] where they use a knowledge base to refine the vectorization of cartographic maps.

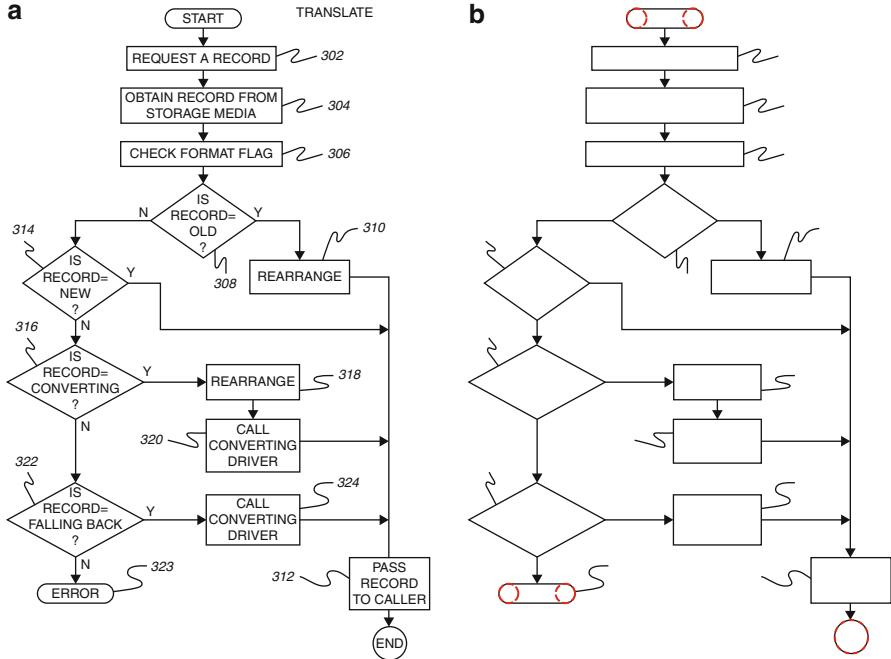
## Arcs and Other Primitives

Until now the vectorization methods were just focused on dealing with straight lines; however, in order to be useful for higher-level interpretation phases, a raster-to-vector process should not be limited to the recognition of segments and should be able to deliver other geometric primitives.

In the literature, several raster-to-vector methods that deliver other geometric primitives such as dashed lines [7, 36] or halftone dots [37] can be found. However, the most prolific research area related to the vectorization problem is the circular arc detection.

According to Dori and Wenyin [21], the arc detection methods can be separated in two families. The first family is based on the circular Hough transform and directly works on the original pixels of the image. An example of such arc detection is presented in Fig. 15.6. Some examples are the works presented in [9, 12, 52]. The main difficulty with these approaches is that they often are of considerable complexity and quite sensitive to the image quality. On the other hand, the second family of methods work on the pixel chains or the resulting segments from the polygonal approximation in order to estimate their edge curvature and to fit an arc model to these digital curves. Some examples of this family are presented in [41, 54, 63]. The main inconvenient of such methods is that they rely on a previous step of line extraction and polygonal approximation instead of working with the original bitmap. Potential errors can be introduced in these preliminary stages and thus the accuracy and reliability of the system might be damaged.

Finally, it is worth to mention that for more than a decade, arc segmentation contests have been held in ICPR conferences and GREC workshops. The reports of



**Fig. 15.6** Arc detection example. (a) Original drawing, (b) arc detection by circular Hough transform after separating text and computing the skeleton of the original image

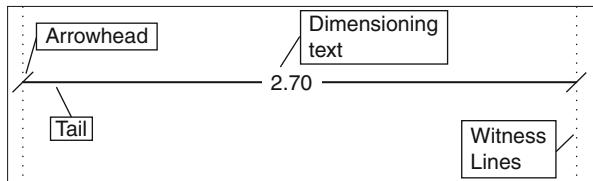
these contents (cf. [58] as an example) provide a clear insight on the evolution of the state of the art in arc segmentation and the remaining challenges.

## Recognition of Dimensions in Technical Drawings

Dimensions are graphical entities appearing in technical documents, usually engineering and architectural drawings. Since this kind of graphical documents is drawn to fit some ANSI or ISO drafting specifications, the dimension sets have a particular structure and follow standard rules. Dimension sets consist of several elements (see Fig. 15.7), namely, arrowheads and tails, witness lines, and text components. In the literature, several system examples dealing with the recognition of dimensioning sets [11, 15, 39] can be found. Usually, the systems dealing with dimensions recognition follow three main steps.

The first step deals with the separation of text from graphics. Usually, state-of-the-art techniques as the ones reviewed in section “Text-Graphics Separation” are used in this step. However, some works that propose text detection methods for the specific case of dimensions can be found, as the one presented by Dori and Velkovitch in [20].

**Fig. 15.7** Dimensions example



Once the text and the graphic elements are separated, the next step is focused on the detection of the dimensioning lines. In order to properly detect these lines, several works that propose to first detect and recognize the arrowheads can be identified. The arrowheads are used as anchor points to then extract the complete witness lines. Instead of using an off-the-shelf symbol recognition method aimed at locating these arrowheads, usually an ad hoc designed arrowhead model is preferred. Some examples of the arrowhead recognition and localization are presented in [39, 43, 71]. A final association of the text with the extracted dimensioning lines is performed. This association is not always straightforward since for every consecutive pair of arrowheads several situations are possible, whether two arrowheads correspond to a dimension set or not and have or do not have some text lying in between.

Since the dimension sets follow some standard rules, a final syntactic validation step is often used. Bidimensional grammars are used to specify how all the possible dimension sets can be generated and describe all the various possible dimension types. Web grammars have been commonly used for this purpose by many authors [15, 19, 49].

---

### Extraction of Texture-Based Graphical Primitives

Some graphical documents contain primitives characterized by a repetitive pattern. For example, isolines representing points of equal value in geographical maps or some dimensions in technical drawings are drawn using dotted lines. On the other hand, technical drawings use to have regions filled by hatched patterns (e.g., walls in architectural drawings or parcels in cadastral maps).

In the recognition of repetitive patterns, Perceptual Organization (PO) plays an important role. PO operates at the intermediate vision level to identify some particular emergent patterns that are used in end tasks as indexing and recognition. Saund et al. [57] studied the application of the PO framework to the field of graphics recognition. In this work, they noticed that PO allows to extract salient patterns with weak prior models and also to represent them compactly making explicit many of their features. In graphical documents, signatures formulated in terms of *proximity*, *continuation*, and *parallelism* of primitives are the basis for the detection of repetitive patterns.

One-dimensional patterns usually consist of dashed or dashed-dotted lines. Human perception gives the ability to “fill the gaps” between the small segments and hence perceive the dashed line as a unique entity. Machines simulate this perceptual

cue by a sequential stepwise recovery of components that meet certain continuity conditions.

Bidimensional patterns consist in structured textures, like hatching or tiling, filling some areas of the drawing. A structured texture can be informally described as a set of primitives regularly arranged following some placement rules. Two major approaches can be used to recognize these structures, namely, Hough-based and syntactic methods.

Classical Hough transform maps each image point  $(x, y)$  into a set of points  $(\theta, \rho)$  that fulfill the equation  $\rho = x \cos \theta + y \sin \theta$ . Detection of peaks in the parameter space constitutes a powerful method to detect straight lines in the input image. A configuration of several straight segments results in a set of corresponding peaks in the Hough space. These peaks have a regular arrangement when the original segments form a hatched or tiled pattern. As can be seen in Fig. 15.8a, a hatched pattern generates a set of aligned peaks at regular step in the Hough space. It is straightforward to detect it with a double Hough transform, i.e., detecting straight lines in the Hough space. The principle that parallel lines in the image generate aligned peaks in the Hough space can be extended to tiled patterns. Fig. 15.8b,c show the Hough transform for two types of tiled patterns. In both cases, lines of peaks appear in the Hough space. In addition, since tiled patterns consist of regular polygons, the lines of peaks in the Hough space are parallel and with an interline distance corresponding to the inner angles of the polygons.

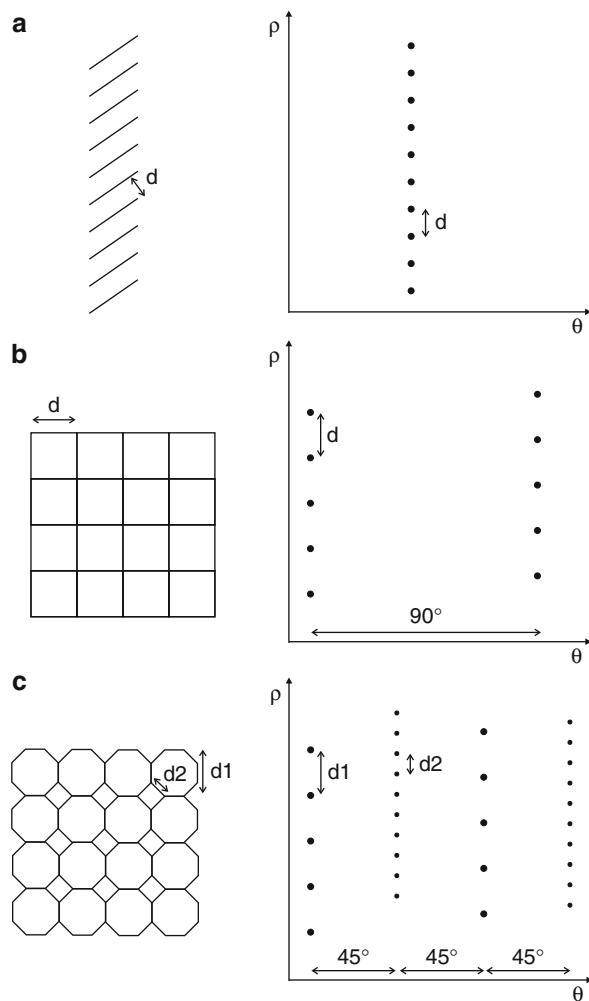
Grammars are a suitable tool to represent repetitive structures in graphic documents, either linear structures, as dashed lines [33], or bidimensional structures, as tiled patterns [48, 56]. Among the different types of syntactic models existing in the literature, graph grammars are the most common one used in graphic documents. Informally speaking, a graph grammar models patterns by a set of rules hierarchically describing patterns in terms of subpatterns. Primitive patterns, called terminal vocabulary in the terminology of grammars, are represented by graphs corresponding to the lines of the document. A grammar is useful to describe repetitive primitives, as hatching or tiling patterns, or syntactic rules that a diagrammatic notation involves, as for dimension symbols or musical scores. The recognition process is hence carried on by a parser. Grammars are complex models supported in a robust theory (syntactic pattern recognition). This is out of the scope of this chapter. The reader is referred to [56] for a further description.

---

## Executive Summary

Graphics recognition concerns on the analysis of the non-textual entities of document images (tables, symbols, line drawings, etc.). It has experienced a growing interest since the 1980s when there was an industrial need for converting raster images of engineering designs and maps to CAD and GIS platforms for subsequent edition, printing, and reflowing. Specific graphics recognition techniques were addressed for the low-mid level stages of the whole interpretation pipeline. In this chapter, the relevant approaches to convert raster images into basic primitives

**Fig. 15.8** Hough space configuration for (a) hatched patterns, (b) (c) tiled patterns



that compound meaningful objects have been described. Such basic primitives are basically vectors (analytic descriptions of lines of pixels as segments, arcs or higher order curves), and text labels (usually associated to dimensions, object labels, georeferences or toponyms). Therefore, the two main operations that are the core of the graphics recognition techniques are text-graphics separation and vectorization.

Having a look to the literature, both text-graphics separation and vectorization can be considered mature techniques from a scientific point of view. Commercial software includes them at reasonable levels of performance. Hence, one can not expect groundbreaking scientific innovations in the future. The challenges have to be foreseen in a more global view, having vectorization and text-graphics separation

---

modules integrated in systems and platforms for massive processing and integral interpretation of documents.

---

## Conclusion

In this chapter, the state of the art in the techniques addressed to extract the compounding units of graphical documents has been reviewed. The needs of the market have been the leading force behind a huge amount of research and development in graphics recognition techniques. Hence, there was an industrial need of converting engineering drawings and maps to CAD and GIS platforms that originated a growing applied research activity in the 1990s on vectorization and text detection in graphical documents. A community on graphics recognition has been consolidating since then including other challenges such as symbol recognition and document interpretation, topics that are reviewed in the following chapters.

Vectorization, or raster-to-vector conversion, is the problem that has more mature solutions. There is still the open question of what vectorization has to be, but this might be a rhetoric question that will never be answered. Vectorization is not a stand-alone functionality, but it is integrated in bigger systems. Hence, the desired performance of a vectorization process depends on the purpose of the entire system. If vectorization is an intermediate step in the process of recognition, it is not needed that vectors perfectly fit into the lines of pixels, but it should be stable in terms of invariance to variations of shapes. On the other hand, when the purpose of the system is just converting to a DXF or SVG format compatible with a CAD system for reprinting or storing the document, then it is better a higher number of primitives to get a perfect fitting into the original lines. It leads to vectorization algorithms with different parameters where the user or the document conditions determine the settings. One of the current challenges of vectorization is to develop systems that learn the configuration of the parameters in terms of the user feedback in previous documents or from the features of the document contents.

Currently, engineering drawings are digitally born documents, so the classical need of vectorization for the sake of conversion from raster to CAD format is quite obsolete. However, there is a latent need for vectorization in novel applications appearing in the market. An example of this is CAD software for nonprofessional use where the user can scan a plan of his/her house to render a simple 3D view augmented with some furniture objects, changed colors of the walls, etc., for decoration purposes. Whatever is the application, vectorization is not usual to be a stand-alone process, but it is combined with the recognition of graphical entities.

The main objective of text-graphics separation is not only to detect the text strings in a graphical document, independently of the font size and the orientation, but to be able to reconstruct the characters touching graphical lines. The baseline methods are the ones of Fletcher and Kasturi [25] and Tombre et al. [68]. These methods are based on the iterative grouping of connected components having some spatial properties (similar size, alignment, regular inter-character distance).

The segmentation invariant to multiple fonts and sizes is solved working at different scales.

Nowadays, the problem of text-graphics separation is having a resurgence but in new applications as form processing in digital mail room platforms for the automation of incoming mail processing. Among the milliards of documents being processed, forms are a big subset. Forms have vertical and horizontal lines forming tables, text boxes, and rule lines. Classical text-graphics separation techniques cannot be applied to solve the segmentation of text in forms because it is usually handwritten. It opens new perspectives and reformulates to some extend the problem of text-graphics separation making closer the areas of graphics recognition and handwriting analysis.

Text-graphics separation is also of special relevance in maps. Not for the classical raster to GIS conversion, but for massive map processing, georeferencing, and retrieval maps using textual queries. In this case, although being a different application, the classical techniques reviewed in this chapter can be applied.

---

## Cross-References

- ▶ [Analysis and Interpretation of Graphical Documents](#)
  - ▶ [Analysis and Recognition of Music Scores](#)
  - ▶ [An Overview of Symbol Recognition](#)
  - ▶ [Page Segmentation Techniques in Document Analysis](#)
  - ▶ [Recognition of Tables and Forms](#)
  - ▶ [Text Segmentation for Document Recognition](#)
- 

## Notes

<sup>1</sup> GREC is the workshop of the technical committee on graphics recognition (TC10) of the International Association of Pattern Recognition (IAPR).

<sup>2</sup> <http://www.qgar.org/>

---

## References

1. Abd-Almageed W, Kumar J, Doermann D (2009) Page ruleline removal using linear subspaces in monochromatic handwritten arabic documents. In: Proceedings of the 12th international conference on document analysis and recognition, Barcelona, pp 768–772
2. Acharyya M, Kundu MK (2002) Document image segmentation using wavelet scale-space features. IEEE Trans Circuits Syst Video Technol 12(12):1117–1127
3. Antoine D, Collin S, Tombre K (1992) Analysis of technical documents: the REDRAW system. In: Structured document image analysis. Springer, Berlin, pp 385–402
4. Boatto L, Consorti V, Del Buono M, Di Zenzo S, Eramo V, Esposito A, Melcarne F, Meucci M, Morelli A, Mosciatti M, Scarci S, Tucci M (1992) An interpretation system for land register maps. Computer 25(7):25–33

5. Cao R, Tan CL (2002) Text/graphics separation in maps. In: Graphics recognition algorithms and applications. Lecture notes in computer science, vol 2390. Springer, Berlin/New York, pp 167–177
6. Chen J, Lopresti D, Kavallieratou E (2010) The impact of ruling lines on writer identification. In: Proceedings of the 2nd international conference on frontiers in handwriting recognition, Kolkata, pp 439–444
7. Chen Y, Langrana NA, Das AK (1996) Perfecting vectorized mechanical drawings. *Comput Vis Image Underst* 63(2):273–286
8. Chhabra AK, Misra V, Arias J (1996) Detection of horizontal lines in noisy run length encoded images: the FAST method. In: Graphics recognition methods and applications. Lecture notes in computer science, vol 1072. Springer, Berlin/Heidelberg, pp 35–48
9. Chiu SH, Liaw JJ (2005) An effective voting method for circle detection. *Pattern Recognit Lett* 26(2):121–133
10. Dalitz C, Droettboom M, Pranzas B, Fujinaga I (2008) A comparative study of staff removal algorithms. *IEEE Trans Pattern Anal Mach Intell* 30:753–766
11. Das AK, Langrana NA (1997) Recognition and integration of dimension sets in vectorized engineering drawings. *Comput Vis Image Underst* 68(1):90–108
12. Davies ER (1988) A modified Hough scheme for general circle location. *Pattern Recognit Lett* 7(1):37–43
13. Di Zenzo S, Cinque L, Levialdi S (1996) Run-based algorithms for binary image analysis and processing. *IEEE Trans Pattern Anal Mach Intell* 18(1):83–89
14. Doermann D (1998) An introduction to vectorization and segmentation. In: Graphics recognition algorithms and systems. Lecture notes in computer science, vol 1389. Springer, Berlin/New York, pp 1–8
15. Dori D (1992) Self-structural syntax-directed pattern recognition of dimensioning components in engineering drawings. In: Structured document image analysis. Springer, Berlin/New York, pp 359–384
16. Dori D (1997) Orthogonal zig-zag: an algorithm for vectorizing engineering drawings compared with Hough transform. *Adv Eng Softw* 28(1):11–24
17. Dori D, Liu W (1996) Vector-based segmentation of text connected to graphics in engineering drawings. In: Advances in structural and syntactical pattern recognition. Lecture notes in computer science, vol 1121. Springer, Berlin/New York, pp 322–331
18. Dori D, Liu W (1999) Automated CAD conversion with the machine drawing understanding system: concepts, algorithms, and performance. *IEEE Trans Syst Man Cybern Part A: Syst Hum* 29(4):411–416
19. Dori D, Pnueli A (1988) The grammar of dimensions in machine drawings. *Comput Vis Image Underst* 42(1):1–18
20. Dori D, Velkovitch Y (1998) Segmentation and recognition of dimensioning text from engineering drawings. *Comput Vis Image Underst* 69(2):196–201
21. Dori D, Wenyin L (1998) Stepwise recovery of arc segmentation in complex line environments. *Int J Doc Anal Recognit* 1(1):62–71
22. Dori D, Wenyin L (1999) Sparse pixel vectorization, an algorithm and its performance evaluation. *IEEE Trans Pattern Anal Mach Intell* 21(3):202–215
23. Dosch P, Tombre K, Ah-Soon C, Masini G (2000) A complete system for the analysis of architectural drawings. *Int J Doc Anal Recognit* 3(2):102–116
24. Fan KC, Chen DF, Wen MG (1998) Skeletonization of binary images with nonuniform width via block decomposition and contour vector matching. *Pattern Recognit* 31(7): 823–838
25. Fletcher LA, Kasturi R (1988) A robust algorithm for text string separation from mixed text/graphics images. *IEEE Trans Pattern Anal Mach Intell* 10(6):910–918
26. Fukada Y (1984) A primary algorithm for the understanding of logic circuit diagrams. *Pattern Recognit* 17(1):125–134
27. Han CC, Fan KC (1994) Skeleton generation of engineering drawings via contour matching. *Pattern Recognit* 27(2):261–275

28. Hilaire X, Tombre K (2006) Robust and accurate vectorization of line drawings. *IEEE Trans Pattern Anal Mach Intell* 28(6):890–904
29. Hoang TV, Tabbone S (2010) Text extraction from graphical document images using sparse representation. In: Proceedings of the 9th IAPR international workshop on document analysis systems, Boston, pp 143–150
30. Hori O, Tanigawa S (1993) Raster-to-vector conversion by line fitting based on contours and skeletons. In: Proceedings of the 2nd international conference on document analysis and recognition, Tsukuba, pp 353–358
31. Jain A, Bhattacharjee S (1992) Text segmentation using gabor filters for automatic document processing. *Mach Vis Appl* 5(3):169–184
32. Janssen RDT, Vossepoel AM (1997) Adaptive vectorization of line drawing images. *Comput Vis Image Underst* 65(1):38–56
33. Jonk A, van den Boomgaard R, Smeulders A (1999) Grammatical inference of dashed lines. *Comput Vis Image Underst* 74(3):212–226
34. Journet N, Eglin V, Ramel JY, Mullot R (2005) Text/graphic labelling of ancient printed documents. In: Proceedings of the 8th international conference on document analysis and recognition, Seoul, pp 1010–1014
35. Kaneko T (1992) Line structure extraction from line-drawing images. *Pattern Recognit* 25(9):963–973
36. Kasturi R, Bow ST, El-Masri W, Shah J, Gattiker JR, Mokate UB (1990) A system for interpretation of line drawings. *IEEE Trans Pattern Anal Mach Intell* 12(10):978–992
37. Kawamura K, Watanabe H, Tominaga H (2004) Vector representation of binary images containing halftone dots. In: Proceedings of the IEEE international conference on multimedia and expo, Taipei, pp 335–338
38. Kolesnikov AN, Belekhov VV, Chalenko IO (1996) Vectorization of raster images. *Pattern Recognit Image Anal* 6(4):786–794
39. Lai CP, Kasturi R (1994) Detection of dimension sets in engineering drawings. *IEEE Trans Pattern Anal Mach Intell* 16(8):848–854
40. Lam L, Lee SW, Suen CY (1992) Thinning methodologies – a comprehensive survey. *IEEE Trans Pattern Anal Mach Intell* 14(9):869–885
41. Lamirov B, Guebbas Y (2010) Robust and precise circular arc detection. In: Graphics recognition. Achievements, challenges and evolution. Lecture notes in computer science, vol 6020. Springer, Berlin/Heidelberg, pp 49–60
42. Lee KH, Cho SB, Choy YC (2000) Automated vectorization of cartographic maps by a knowledge-based system. *Eng Appl Artif Intell* 13(2):165–178
43. Lin SC, Ting CK (1997) A new approach for detection of dimensions set in mechanical drawings. *Pattern Recognit Lett* 18(4):367–373
44. Lin X, Shimotsuji S, Minoh M, Sakai T (1985) Efficient diagram understanding with characteristic pattern detection. *Comput Vis Image Underst* 30(1):84–106
45. Loo PK, Tan CL (2001) Detection of word groups based on irregular pyramid. In: Proceedings of the 6th international conference on document analysis and recognition, Seattle, pp 200–204
46. Lu Z (1998) Detection of text regions from digital engineering drawings. *IEEE Trans Pattern Anal Mach Intell* 20(4):431–439
47. Luo H, Kasturi R (1998) Improved directional morphological operations for separation of characters from maps/graphics. In: Graphics recognition algorithms and systems. Lecture notes in computer science, vol 1389. Springer, Berlin/New York, pp 35–47
48. Matsuyama T, Saburi K, Nagao M (1982) A structural analyzer for regularly arranged textures. *Comput Graph Image Process* 18:259–278
49. Min W, Tang Z, Tang L (1993) Using web grammar to recognize dimensions in engineering drawings. *Pattern Recognit* 26(9):1407–1416
50. Monagan G, Roosli M (1993) Appropriate base representation using a run graph. In: Proceedings of the 2nd international conference on document analysis and recognition, Tsukuba, pp 623–626
51. Niblack CW, Gibbons PB, Capson DW (1992) Generating skeletons and centerlines from the distance transform. *CVGIP: Graph Models Image Process* 54(5):420–437

52. Olson CF (1999) Constrained Hough transforms for curve detection. *Comput Vis Image Underst* 73(3):329–345
53. Rosin PL (2003) Assessing the behaviour of polygonal approximation algorithms. *Pattern Recognit* 36(2):505–518
54. Rosin PL, West GA (1989) Segmentation of edges into lines and arcs. *Image Vis Comput* 7(2):109–114
55. Roy PP, Pal U, Lladós J (2012) Text line extraction in graphical documents using background and foreground information. *Int J Doc Anal Recognit* 15(3):227–241
56. Sánchez G, Lladós J (2004) Syntactic models to represent perceptually regular repetitive patterns in graphic documents. In: *Graphics recognition. Recent advances and perspectives. Lecture notes in computer science*, vol 3088. Springer, Berlin/New York, pp 166–175
57. Saund E, Mahoney J, Fleet D, Larner D (2002) Perceptual organization as a foundation for graphics recognition. In: *Graphics recognition: algorithms and applications*. Springer, Berlin/New York, pp 139–147
58. Shafait F, Keysers D, Breuel TM (2008) GREC 2007 arc segmentation contest: evaluation of four participating algorithms. In: *Graphics recognition. Recent advances and new opportunities. Lecture notes in computer science*, vol 5046. Springer, Berlin/New York, pp 310–320
59. Shih CC, Kasturi R (1989) Extraction of graphic primitives from images of paper based line drawings. *Mach Vis Appl* 2(2):103–113
60. Shimotsuji S, Hori O, Asano M, Suzuki K, Hoshino F, Ishii T (1992) A robust recognition system for a drawing superimposed on a map. *Computer* 25(7):56–59
61. Song J, Lyu MR (2005) A Hough transform based line recognition method utilizing both parameter space and image space. *Pattern Recognit* 38(4):539–552
62. Song J, Su F, Tai CL, Cai S (2002) An object-oriented progressive-simplification-based vectorization system for engineering drawings: model, algorithm, and performance. *IEEE Trans Pattern Anal Mach Intell* 24:1048–1060
63. Song J, Lyu MR, Cai S (2004) Effective multiresolution arc segmentation: algorithms and performance evaluation. *IEEE Trans Pattern Anal Mach Intell* 26(11):1491–1506
64. Tan CL, Ng PO (1998) Text extraction using pyramid. *Pattern Recognit* 31(1):63–72
65. Tombre K (1998) Analysis of engineering drawings: state of the art and challenges. In: *Graphics recognition algorithms and systems. Lecture notes in computer science*, vol 1389. Springer, Berlin/New York, pp 257–264
66. Tombre K, Tabbone S (2000) Vectorization in graphics recognition: to thin or not to thin. In: *Proceedings of the 15th international conference on pattern recognition*, Barcelona, pp 91–96
67. Tombre K, Ah-Soon C, Dosch P, Massini G, Tabbone S (2000) Stable and robust vectorization: how to make the right choices. In: *Graphics recognition recent advances. Lecture notes in computer science*, vol 1941. Springer, Berlin/New York, pp 3–18
68. Tombre K, Tabbone S, Pelissier L, Lamiroy B, Dosch P (2002) Text/graphics separation revisited. In: *Document analysis systems V. Lecture notes in computer science*, vol 2423. Springer, Berlin/New York, pp 615–620
69. Vaxivière P, Tombre K (1994) Subsampling: a structural approach to technical document vectorization. In: *Structure and pattern recognition. Proceedings of the IAPR Workshop on syntactic and structural pattern recognition*, Haifa, Israel, pp 323–332
70. Wahl F, Wong K, Casey R (1982) Block segmentation and text extraction in mixed text/image documents. *Comput Graph Image Process* 20(4):375–390
71. Wendling L, Tabbone S (2004) A new way to detect arrows in line drawings. *IEEE Trans Pattern Anal Mach Intell* 26(7):935–941
72. Wenyin L, Dori D (1996) Sparse pixel tracking: a fast vectorization algorithm applied to engineering drawings. In: *Proceedings of the 13th international conference on pattern recognition*, Vienna, pp 808–812
73. Wenyin L, Dori D (1998) A survey of non-thinning based vectorization methods. In: *Advances in pattern recognition. Lecture notes in computer science*, vol 1451. Springer, Berlin/New York, pp 230–241

## Further Reading

For further reading, the reader is addressed to the classic literature contributions of the field. The key references in text-graphics separation are the work of Fletcher and Kasturi [25] and the subsequent improvements proposed by Tombre et al. [68]. Concerning the problem of raster-to-vector conversion, recommended readings are the surveys and analysis papers of the QGAR group led by K. Tombre [66, 67]. Practitioners interested in the implementation of the methods can download the open source QGAR software library.<sup>2</sup> Finally, to be kept informed of the progress on graphics recognition techniques and the area at large, the reader is referred to the publications arising from the IAPR graphics recognition workshop (GREC series), held every two years since 1995, and the post workshop book with selected papers that is published by *Springer* in the *Lecture Notes in Computer Science* series.

---

# An Overview of Symbol Recognition

# 16

Salvatore Tabbone and Oriol Ramos Terrades

## Contents

Introduction.....	524
History.....	524
Evolution of the Research Field.....	526
Features.....	528
Polar Representation.....	530
Invariance to Similarities.....	530
Pixel Descriptors.....	531
Multi-scale/Resolution Decomposition.....	534
Structural Descriptors.....	535
Recognition Methods.....	538
Distance and Similarity Measures.....	539
Embedding Methods.....	540
Structural Classification.....	540
Statistical Classification.....	541
Symbol Spotting.....	543
Conclusion.....	546
Cross-References.....	547
References.....	547
Further Reading.....	551

---

### Abstract

According to the Cambridge Dictionaries Online, a symbol is a sign, shape, or object that is used to represent something else. Symbol recognition is a subfield of general pattern recognition problems that focuses on identifying, detecting,

---

S. Tabbone (✉)

Université de Lorraine – LORIA, Vandœuvre-lès-Nancy Cedex, France  
e-mail: [tabbone@loria.fr](mailto:tabbone@loria.fr)

O.R. Terrades

Universitat Autònoma de Barcelona – Computer Vision Center, Bellaterra, Spain  
e-mail: [oriolrt@cvc.uab.es](mailto:oriolrt@cvc.uab.es)

and recognizing symbols in technical drawings, maps, or miscellaneous documents such as logos and musical scores. This chapter aims at providing the reader an overview of the different existing ways of describing and recognizing symbols and how the field has evolved to attain a certain degree of maturity.

---

### Keywords

Pattern recognition • Shape descriptors • Structural descriptors • Symbol recognition • Symbol spotting

---

## Introduction

In any symbol recognition process, the following operations are usually performed (not necessarily in the same order): segmentation, feature extraction, invariance to similarities, and comparison. Traditionally, these operations are seen as a two-step process consisting of feature extraction and symbol recognition. Most of the features are extracted on segmented symbols. However, symbols are often embedded in technical documents, connected to other symbols, and associated with text. Moreover, it is usually very difficult to perform symbol recognition by simply assuming that they have been cleanly extracted. Therefore, “symbol spotting” methods have been proposed to localize symbols without the need of a full segmentation step.

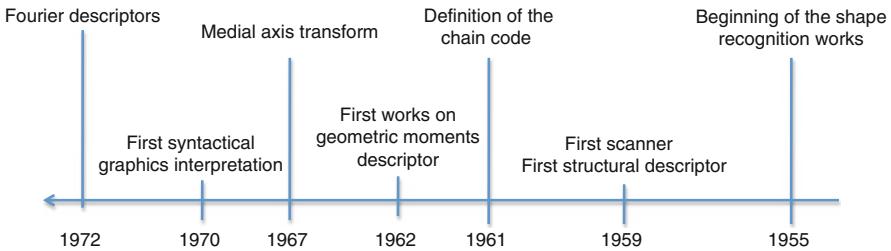
Some of the methodologies proposed in the forthcoming sections have been developed to deal with particular problems coming from technical documents. The evolution of symbol recognition has a close link with text recognition because characters can also be considered as symbols. Several methods in symbol recognition have been inspired from character recognition. Specialized techniques for OCR are described in details in ►Part C (Text Recognition) of this book, and this chapter will focus on symbol recognition only.

This chapter starts with a brief history of pattern recognition methods related to symbol recognition from the 1950s until today. This historical review will be used as a basis for the structure of the forthcoming sections.

## History

The first works on pattern recognition appeared in the late 1950s and early 1960s (Fig. 16.1). These studies were modest in terms of objectives and data used in experimentation, but at that time authors of these works felt the potential of applications behind the pattern recognition field. Forty years later, much progress has been made but there is still room for researches.

From today’s perspective where everyone has a camera and can transmit images worldwide instantly, one of the main difficulties that researchers had 40 years ago was image acquisition and manipulation. For instance, the Lincoln Laboratory was



**Fig. 16.1** Chronological history related to shape recognition purposes

one of the few laboratories in the world that could read and process digital images through the “Computer Memory Test” (CMT) in 1955. At that time, Dinneen [25] studied whether the averaging and edge operators could be used for simple shape recognition purposes (images of A’s, O’s, triangles, and squares). In 1959, Bomba [10] proposed the first structural descriptor for character recognition and based on a set of features (straight lines at different orientations, four orientations of T- and L-junctions, and some selected V-junctions). To extract those features, the averaging operator studied by Dinneen was applied to reduce the noisy pattern and to normalize line width. These features were computed from *sliding windows* where local features were computed by aggregating pixels at different orientations. As a result of this process, characters were decomposed (segmented) into different layers and recognition was performed with a decision tree. In 1961, Freeman [34] proposed the *Chain Code*, an encoding method for the representation of arbitrary planar curves, by a sequence of integers ranging from 0 to 7, largely used in many applications and until now.

These previous works by Dinneen, Bomba, and Freeman assumed that patterns (squares and triangles) and characters were isolated inside the image, perfectly oriented, and of the same scale. In 1962, Hu [40] defined a first set of invariant features, namely, geometric moments, to deal with shapes at different positions, scales, and orientations. The theory of moment invariants was based on the algebraic invariant theory developed during the second half of the nineteenth century by Cayley, Sylvester, and Boole. In these works, Hu observed that the more the number of moments were used, the higher the discrimination capacity of his method would be.

All these methods proposed a set of features for describing shapes based on authors intuition about which kind of shape information was relevant for recognition purposes. At that time, a set of features inspired by psychological and psychophysical works [4] was proposed in [9, 82]. Based on those psychophysical knowledge, Blum [9] proposed in 1967 the Medial Axis Transform (MAT). The MAT is defined as the locus of points that are equidistant to the shape contour, and it has largely been used in many other recognition systems, sometimes under the name *Shape Skeleton*. Blum used the MAT to represent pattern’s symmetrical

lines and observed some properties that make this transform useful for recognition purposes.

Zahn and Roskies [82] proposed a set of descriptors based on shape contours in 1972. By construction, the amplitude of the Fourier coefficients are invariant to shape's scale, position, and rotation. However, the phase of the Fourier coefficients lacks invariance properties because it depends on the starting point used to parameterize the curve. Thereby, Zahn and Roskies proposed a family of invariant functions of the phase, regardless the starting point.

The aforementioned contributions are typical examples of methods in which authors supposed that symbols had previously been segmented from the document. On the contrary, some works assumed that symbols were connected to other parts of the document. For instance in 1969, Shaw [71] proposed the Picture Description Language (PDL) to describe graphics for the interpretation of electrical circuits. In this case the symbols representing each of the circuit elements were not segmented from the document, and their recognition was performed during the interpretation of the document through a grammar. Shaw represented circuits by directed graphs where each node was a segment of the circuit and the nodes were connected based on adjacency segment relations [35]. This chapter will not go into details about the use of grammar as an interpretation method since it is discussed in ►Chap. 17 (Analysis and Interpretation of Graphical Documents) and it only focuses on the representation way.

In these early works, from a statistical point of view, it was necessary that symbols were segmented and the number of classes was quite limited (rather printed capital letters), and from structural one, the described relationships were narrowed to adjacent relationships (regardless of inclusion or overlap). It should also be noted that, at that time, different algorithms were not available in common programming languages as debugging tools or image processing libraries like today (e.g., C language was developed in the late 1960). Therefore, the development of any new method had a high cost in time and resources. Later on, with the advent of the IBM 704 and the construction of the first scanner, researchers were able to process images in a more similar manner to what is done today. The majority of the most competitive state-of-the-art methods nowadays rely on concepts and ideas that could be found in these early works. The next section will show how different contributions by adding more processing layers will refine and improve these early works.

## Evolution of the Research Field

Symbol recognition has become a fertile field where several methods have been proposed in many directions. That methods introduced in the 1960s and 1970s are still relevant today, despite some improvements. For instance, the Angular Radial Transform (ART) descriptor [53], which is included in the MPEG-7 standard, is an evolution of descriptors based on Zernike moments, which in turn evolved from the Hu moments [40]. Likewise, based on the works of Zahn and Roskies, the univariate [63, 79] and bivariate Fourier descriptors [47, 84] have been widely used

and reused to describe shapes for different applications. The current technology makes that methods that were impossible to apply 50 years ago, due to their computation complexity, become possible. Early works in these years sowed the seeds for the growth of many of today's methods.

For the following, to illustrate the evolution of symbol recognition methods, two axes of changes are selected. The first axe will show how structural methods have evolved into representations of complex data, which has led to increasingly efficient matching algorithms. The second one will show how the problem of deformation of symbols was addressed whatever the used method.

Structural methods focus on describing relations among elementary parts of symbols, namely, *primitives*, that make up symbols. Primitives are, in most of the cases, vectors and arcs, and the relations between them could be their adjacency or inclusion/overlap. Trees and graphs are data structures that best represent this type of information, and the recognition process is to find patterns of symbols in these structures. Therefore, structural approaches focus on developing matching methods that are efficient enough to find substructure within graphs or trees. It is important however to use suitable data structures so that matching algorithms give the expected results. For instance, directed graphs have been used for the representation of electronic circuits [71]. In addition, some attempts to use associative graphs to describe electrical diagrams [22] and shapes [62] will also be mentioned later in this chapter. In 1982, Bunke [11] used attributed graphs for the interpretation of electrical diagrams. This method was then extended as a basis for the proposal of region adjacency graphs (RAG) for similar purposes in 2001 [48].

Despite many improvements, matching algorithms are still so computationally expensive that they cannot be applied to large data sets. For this reason, the interest in kernel embedding methods has increased recently. The goal of such methods is to transform graphs in feature vectors to apply machine learning and statistical pattern recognition techniques. Further details on these methods are discussed in section “[Embedding Methods](#).”

These structures are able to represent any kind of technical documents by establishing simple relations among primitives. However, symbols can also be deformed or show local distortions near boundaries, and existing methods at that time, such as the *Chain Code*, were not able to deal with them. In the late 1970s, these difficulties were known and relaxation techniques were used to find approximative correspondences between shapes with, or without, incomplete information [22]. The edit distance is another technique used for the same kind of problem [48, 55].

A slightly different approach to the previous ones are elastic, or deformable, models. These methods consider symbols as objects that can be deformed by adding enough energy to warp one shape into another. Thus, a vectorial field based on curve equations was used by Burr to transform one symbol into another [14]. In the same vein, deformable models based on active contours were created to calculate similarities between handwritten symbols [77].

In the 1980s and 1990s, the use of graphics tablets with CAD software allowed the acquisition of images of diagrams and freehand sketches of architectural

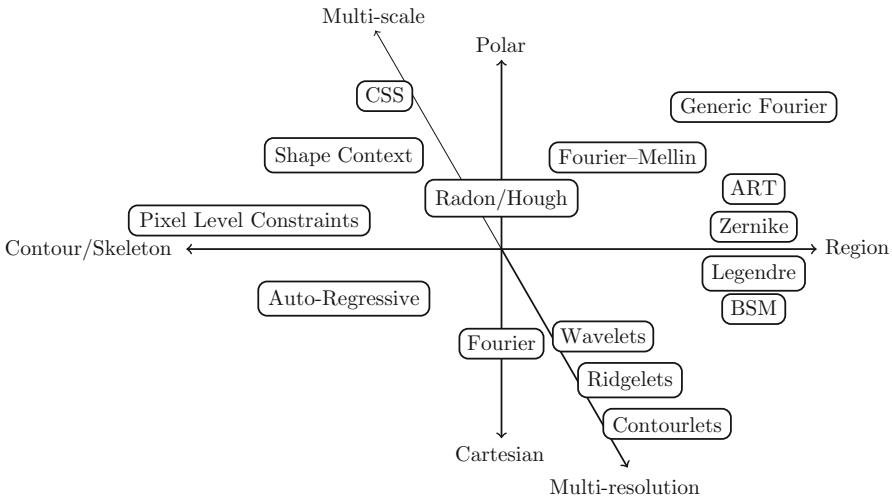
symbols directly in addition to digitized documents. These new devices allowed to explore new symbol recognition applications such as the recognition of mathematical expressions [52]. In addition, online acquisition added dynamic information of symbol drawing, which was used for segmentation and recognition tasks as well. However, the use of this dynamic information introduced new challenges to the symbol recognition field. For instance, the time of response of the system became an important issue to be taken into account, and the order for which the strokes of a symbol were drawn affected the recognition results. In 2010, an adjacency grammar, which can be generated either on- or off-line, was proposed in [54]. Depending on the source of data, a set of primitives were taken and used as terminal elements. The derived adjacency grammar is built from region adjacency graphs [48], but unlike the grammars proposed by Shaw [71] or Bunke [11], inclusion and neighborhood relations between terminal elements and between terminal and nonterminals elements were defined.

These methods are just a few examples of how the problem of nonrigid deformations of the symbols has been tackled. In general, structural descriptors are more flexible than statistical ones because the adopted metrics and matching algorithms allow greater variability between symbols of the same class. For structural descriptors, relationships between primitives are usually independent of position, scale, and rotation since they are defined locally. The matching algorithms are, as with nonrigid transformations, in charge of finding symbols regardless of the position, scale, and rotation. Statistical descriptors work differently from the structural ones to be invariant to similarities. In statistical methods, relationships between primitives are not made explicitly. Images of symbols are seen as surfaces or contours defined in the plane, and features are obtained as the result of applying mathematical transformations. Thus, to be invariant to these deformations, the possible deformations of symbols are taken into account [5, 6] in the comparison stage and not in the statistical descriptor. Generally speaking, statistical descriptors can achieve invariance just before feature extraction by using some kind of normalizations. Ideally, the similarity invariance should be achieved during the process of feature extraction. More details on descriptors invariance will be found in the next section.

---

## Features

This section focuses on feature extraction methods for the construction of symbol descriptors. Several surveys have been proposed in the literature to summarize advances in shape descriptors [61, 85]. However, due to the large number of methods, and since many of them are combinations of the previous ones which could be of different types, it is difficult to establish a clear categorization of symbol recognition methods. Broadly speaking, descriptors are divided into four main groups. On the one hand, descriptors are categorized according to their structural or statistical properties. On the other hand, descriptors are classified depending on whether they are extracted from the regions or the contours. A slightly different terminology had been used by Pavlidis [61] in 1978. Pavlidis divided *algorithms*



**Fig. 16.2** A taxonomy of pixel feature extraction methods

for shape analysis in several binary classes: *external* methods refer to methods defined over the local boundary, whereas *internal* methods are defined over the whole shape. Pavlidis also made another distinction between *scalar* and *domain* transforms. Domain methods transform one image to another, whereas scalar methods compute scalar features from input images. According to these criteria, he defines the following four classes of algorithms: *external scalar* transforms, *internal scalar* transforms, *external space domain* techniques, and *internal space domain* techniques. Scalar features are simple descriptors like area, compactness, rectangularity, and ellipticity.

Nevertheless, the distinction between contours and regions is somehow confused. Indeed, many transforms can be applied to both *contours* and *regions* (e.g., Fourier transform, wavelet transform, Radon transform, regression methods), and many of their properties do not depend on whether they are applied on one-dimensional or two-dimensional data. Moreover, the notion of contour also depends on the topology of a symbol. Full symbols such as logos are different from the ones like wire diagrams. Extracting contours in the latter case means that the skeleton, or the MAT (see ▶Chap. 15 (Graphics Recognition Techniques)), is considered, whereas for full symbols, the contour means the external shape. Moreover, as symbols are often represented in black and white, for noiseless full symbols, describing a symbol by its region or external contour is equivalent. On contrary, for noisy full symbols, region descriptors are more robust than contour ones. Following these considerations, Fig. 16.2 proposes a taxonomy of the most representative pixel feature methods following their representation (polar or Cartesian), their decomposition in a multi-scale/resolution space, and the domain of applicability (contour/skeleton vs. region).

## Polar Representation

Shapes are usually expressed in Cartesian coordinates but sometimes descriptors are based on polar coordinates. In the polar representation, the description of a shape is more concise and therefore less sensitive to noise and shape variations. However, the main drawback of this representation is the definition of the coordinate origin. The change of Cartesian to polar, and also polar to Cartesian, is based on the distance of points from the origin. The same shape can be represented in a very different manner depending on the definition of the origin, leading to instability when the shape is noisy. Examples of methods to determine the origin coordinates are the center of gravity, the center of the bounding box, or the center of the minimal enclosing circle. Each of these methods will result in a different polar description of the shape. Another drawback is the effect of shift in a polar description. While a shift in Cartesian coordinates follows a linear map, a shift in polar coordinates follows a sinusoidal function. This fact causes difficulty in getting invariance to shape translation. The change of Cartesian-to-polar coordinates is also a time-consuming process, and methods on pseudo-polar transform have been proposed using concentric squares instead of concentric circles to represent shapes [64] to speed up the change. However, this transform introduces geometric distortions due to the approximation of circles by squares.

Although most of the descriptors using this kind of representation are built from pixel images, some examples of structural descriptors coded in polar coordinates are also found [42]. Hough and Radon transforms describe straight lines in terms of slope angle and distance to the origin which provides a polar description that has been used in structural and statistical descriptors. The  $\mathcal{R}$ -transform, which is an integral function of the Radon transform in the radial parameter, gives rise to a descriptor called  $\mathcal{R}$ -signature [38, 74]. Ridgelets descriptor [64] is defined by performing a wavelet transform on the Radon space or combination of several transforms (Radon, Fourier, and wavelets [16]).

Polar Fourier transform simply consists in computing 2D Fourier transform in polar coordinates. The projection in the polar space provides the rotation invariance. An example is the generic Fourier descriptor [85]. The Fourier–Mellin transform is defined by using the Fourier and Mellin transforms, respectively, on the angular and radial parameters [1, 39]. Trace transform generalizes the Radon transform by applying other functionals on a set of lines [43].

ART [53] decomposes a shape in a basis defined by the multiplication of a radial and an angular function. Both functions, angular and radial, are defined by a parameter that determines the ART coefficients. Finally, Zernike moments [17] are defined by the same angular function as ART descriptors, but the radial function is a real-valued polynomial.

## Invariance to Similarities

Usually, in invoice documents, trademark logos are not rotated and thus, descriptors do not need to be rotation invariant. On the contrary, in architectural or electronic

documents, symbols can be found in almost any orientation. Such documents could be scanned at different resolutions or, in case of architectural maps, drawn at different scales. In such cases, descriptors have to be invariant to scale. For instance, the descriptor in Fig. 16.3 is invariant to scale and translation but not rotation. In the latter case, the descriptor is shifted horizontally and circularly. Also, when the document is nonplanar such as thick-bound book pages or for documents captured by mobile devices (see ▶Chaps. 2 (Document Creation, Image Acquisition and Document Quality) and ▶4 (Imaging Techniques in Document Analysis Processes)), images are often warped and this deformation can be approximated by an affine transform.

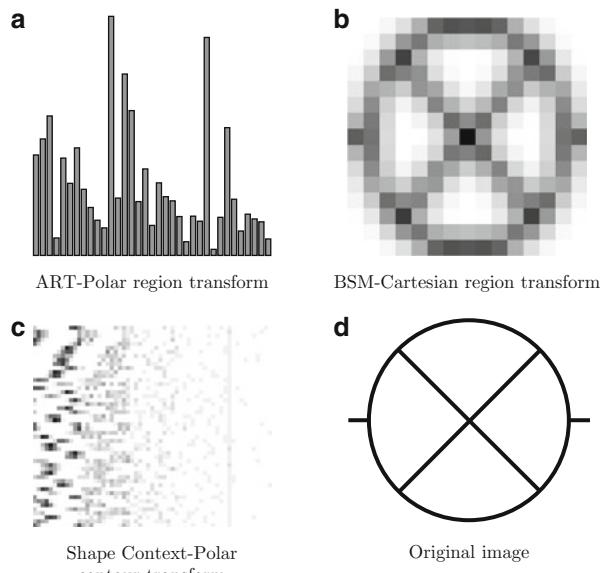
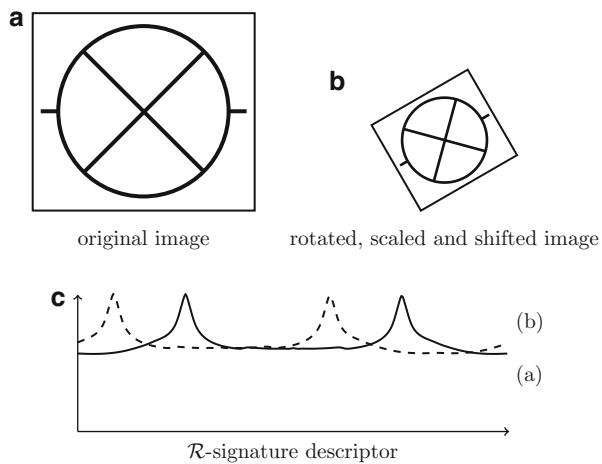
There are different ways of achieving invariance to similarities. The first one is to extract directly from symbols an invariant descriptor. The next section will show a set of different pixel descriptors that are intrinsically invariant to similarities. In other cases, symbol is normalized by estimating its center, scale, and orientation. Having done this, these changes can be reversed to obtain the *original* one. Symbol normalization is sensitive to wrong segmentations, document noise, and partial occlusions or distortions. The estimation of these parameters is done before applying any feature extraction method. When symbols are completely segmented, this normalization is achieved as it is done for polar representation. Thus, the symbol center is defined either as the center of gravity of the symbol or as the center of bounding boxes, convex hull, or minimal enclosing circle. Therefore, invariance to translation and scaling is achieved by shifting the symbol center to the coordinate center and rescaling the symbol to a fixed value. Achieving invariance to symbol rotation by normalization is more critical because sometimes symbols do not have a clear orientation, as it is the case for characters. A typical technique to recover symbol rotation is to use the angle of the main axis defined by the second-order moment, but this technique is sensitive to noise and distortions and not robust when the eccentricity value is near to 1, meaning that the symbol is quite circular. Consequently, when possible, it is better to achieve similarity invariance by means of an invariant feature extraction method since they do not require the estimation of symbol position, scale, and rotation.

The last way of taking into account the invariance is to incorporate the invariance directly into the measure of similarity. For instance, to recover warping deformation, several works use the property of elasticity of the dynamic time warping (DTW) distance [5, 32]. Until now, affine invariance has only been slightly addressed in the symbol recognition community, but with the popularity of mobile devices, the interest on invariance to affine transformation will increase substantially in the future.

## Pixel Descriptors

Pixel descriptors are features directly computed from raw images (Fig. 16.4). These type of descriptors usually have been named *statistical* since traditionally they have been used as input of statistical classifiers. Some examples of the most simple pixel features are the Euler number, the number of connected components, the area,

**Fig. 16.3** Example of similarities invariance



**Fig. 16.4** Example of pixel descriptors

the perimeter, the compactness, the rectangularity, and the symbol ellipticity. These scalar features can be enough to solve very simple recognition problems, or they can also be used as attributes in graph descriptors. For more complex tasks, more complex transforms are needed like those reviewed below.

**The Fourier transform** is probably the most popular extraction method in pattern recognition problems, both in one and two dimensions. There are several

ways of applying the Fourier transform to a planar curve. As explained in the section “[History](#),” the first time the Fourier transform was applied to recognition problems was in 1972 [82]. The phase of the Fourier coefficients computed from shape contours was used to compute similarity between shapes [5]. Moreover, the Fourier transform was applied to the  $\mathcal{R}$ -signature, obtained from the Radon transform of the image, to get invariance to rotation [74]. The strength of Fourier descriptors is that they permit to get a global description of curves without requiring a large number of coefficients. However, they lose their discriminant capability when the similarity between shapes is important because slight shape differences are confused with noise. Bivariate Fourier transform is also used for symbol recognition after applying other transforms like contourlets [15] or Radon transform [16, 39]. The generic Fourier descriptor computed as bivariate Fourier transform in polar coordinates has proved to be more robust to symbol distortions [85].

**Geometric moments** were introduced in 1962 by Hu [40] to build descriptors invariant to similarity transforms. If, instead of using monomials of order  $p$ ,  $x^p$ , orthogonal polynomials like Zernike and Legendre are used over the unitary disk we obtain the Zernike and Legendre moments. Moreover, the moment order is defined as the polynomial degree. The related mathematical theory proves that we can express any bivariate function defined over the unitary disk as a Legendre or Zernike polynomial of infinite degree. Then, an approximation of the original symbol is obtained by truncating these infinite polynomials. Zernike moments have proved to be more discriminant and robust to noise than geometric and Legendre moments. However, they are more computationally expensive. Some comparative studies have been carried out in this direction in [17]. Some other works concerning Legendre and Zernike moments are found in [75, 81], just to mention a few. Based on the geometric invariant theory [57], several wavelet invariant descriptors have been proposed in [28, 76], but most of the proposed invariant functions require contour-based description of symbols and their extension to regions is not straightforward [33, 45].

**Local norm** methods compute the norm over a set of features. This type of descriptors was formalized in [19]. Zoning descriptors are the most basic kind of local norm descriptors where an image is divided into cells and the area is computed on each cell. In general, such descriptors are not invariant to similarities unless symbols are centered and resized before computing local norms. These descriptors are useful since shape description is compact and the size of descriptor is reduced. However, the discrimination capability is less, especially for tasks with a lot of different classes of symbols. The blurred shape model (BSM) is a sophisticated zoning descriptor which is robust to symbol deformations [29]. The  $\mathcal{R}$ -signature is invariant to shift and scale because the signature is computed along the radial parameter of the Radon transform [74].

**Auto-regressive** (AR) methods consist of computing the parameters of closed curves using regression techniques like least squares. These methods are usually applied to contour curves. Coefficients fitting contour curves are then used to derive descriptors invariant to similarity transforms. Bivariate AR models were proposed in [21, 70] to overcome some shape representation problems instead of univariate

function representing the shape boundary [44]. With a bivariate function, convex and non-convex shapes are treated in the same way. One of the drawbacks of stochastic methods is that the number of coefficients required to describe the shape is high for complex shapes and is usually chosen empirically.

**Curvature function** is based on the second derivative, describes a planar curve (except its position and orientation), and is invariant to shift and rotation in shape. Changes of curvature in shapes are considered to be dominant features, and they have been the focus of detailed studies since the beginning of the 1980s. It has been shown that this kind of descriptor usually has good performance for general shape description purpose. However, the required computation of the second derivative makes this descriptor sensitive to noise. Curvature function has largely been used as symbol descriptor. The concept of *curvature primal sketch*, which is in fact a Curvature Scale Space, was introduced in [3]. Besides, maxima curvature points were proposed in [7]. The curvature function was computed and local maxima points were extracted to construct a structural descriptor. Finally, the Curvature Scale Space (CSS) descriptor [56] is based on a multi-resolution description of the curvature function and was included in the MPEG-7 standard [53].

**Directional** methods compute shape gradients in several directions [49]. They are essentially based on the implementation of discrete derivatives, and, as a result, their strengths and weaknesses are strongly influenced by this fact. In general, these descriptors are extremely sensitive to contour distortions and local occlusions of shapes, but they can be easily applied and adapted. Therefore, these descriptors have been used for both machine-printed and handwritten non-Latin characters with different degrees of success since the end of 1970s. In these works, directional information is extracted by means of different masks like Kirsh or Sobel masks. A different approach has been proposed by Kimura et al. in 1997 based on *Chain Codes* for handwritten Japanese character recognition [46] and successfully applied for mathematical symbols recognition [52]. An advantage of this type of descriptors in comparison with the gradient-based approach is the computation time. However, the Chain Code-based descriptor usually shows lower accuracy compared to gradient-based descriptors.

**Histogram descriptors** are empirical approximations of probability density functions of features. These descriptors are useful because they allow us to reduce the feature space into a small set of bins where feature information is accumulated, like directions or variations in gradient modulus. Their use is not restricted to shape description. For instance, histograms based on color features [73], computed from global, or local, shape features [36, 80], have been proposed in the literature (Fig. 16.4).

## Multi-scale/Resolution Decomposition

Some of the previous descriptors are defined in multi-scale or multi-resolution frameworks. The underlying hypothesis is that the most relevant features are

preserved at rough scales. The multi-scale decomposition of a shape basically consists in smoothing by convolving it with a *scale* function, which is most of the time the Gaussian function. The scale function depends on a parameter,  $\sigma$ , which is usually referred to as the *scale parameter*. An inherent drawback of multi-scale decomposition is that the size of the shape is always the same, in spite of a reduction in the number of features. This means that the size of the descriptor is the same, regardless of the scale used. For this reason, descriptors based on this method usually extract features from a pyramidal decomposition of shape from the roughest scale to the finest one to complete the shape description. Contour-based scale space descriptors are obtained by convolving the contours of the shape by a Gaussian kernel (first and second derivative of a Gaussian filter). The Curvature Scale Space [56] and the “primal sketch curvature” [3] are defined in a multi-scale context where local maximal curvatures are extracted, leading to a reduction in the size of the descriptor. Region-based scale space descriptors are obtained by applying Gaussian kernels over the whole image. For instance, the SIFT descriptor [49] is defined by selecting key points at extrema of the difference of Gaussian function in a scale space where at each scale and at each selected point a directional descriptor is computed.

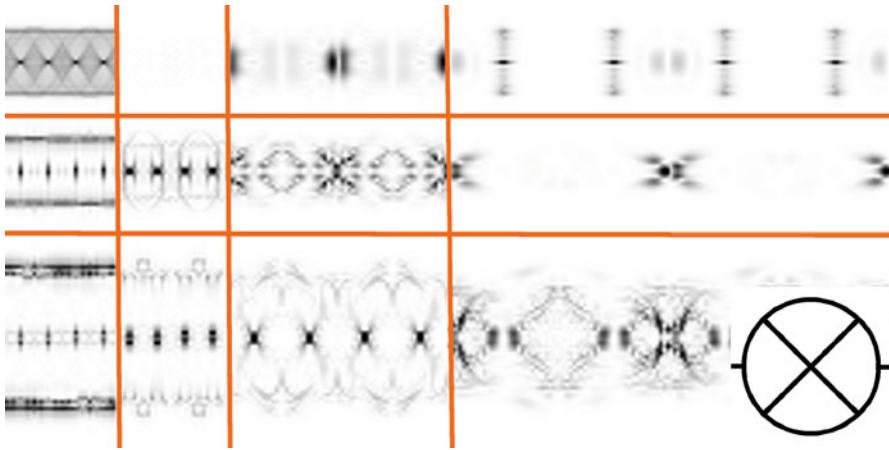
Different from multi-scale decomposition, multi-resolution decomposition is derived from the Multi-Resolution Analysis (MRA) theory [51]. Wavelets were the first MRA methods used as symbol descriptors and applied to shape contours in combination with affine invariants [28]. Moreover, wavelet transform has also been used directly on images to detect horizontal or vertical lines and corners. As symbols are composed of lines oriented in any direction, the performance of bivariate wavelets descriptors in symbol recognition tasks is not high. Therefore, other MRA methods like Gabor wavelets [78], ridgelets [64], and contourlets [15] have been used to overcome this problem of line orientation (Fig. 16.5).

## Structural Descriptors

*Structural* descriptors consider the shape structure in their definition. Shape structures are the logical relations (perpendicularity, adjacency, crossing, and so on) between the *primitive* elements composing the shape. These types of descriptors are usually stored in *graph* or *grammar* structures.

This section is dedicated in the descriptors themselves or, more specifically, in the existing structures to represent the relations between shape entities. The next section “[Structural Classification](#)” will consider main techniques for matching algorithms and methods to reduce their high complexity. Structural descriptors for graphics recognition can be broadly divided into two classes: syntactic and prototype-based descriptors.

Syntactic descriptors are determined by a grammar, which is based on the formal language theory introduced by Chomsky in the middle of the 1950s [37], for graphic document interpretation (see ▶ [Chap. 17](#) (Analysis and Interpretation of Graphical Documents) for further details). A grammar is a condensed representation of a



**Fig. 16.5** Example of a multi-resolution descriptor based on ridgelet decomposition

large set of prototypes. From a finite set of elements and a set of rules, a large set of prototypes are produced in a similar way as in human language, in which alphabets and language grammar rules allow us to produce words. This kind of representation is suitable when the number of prototype patterns is big, when common substructures among patterns are large, and when the available knowledge about the structure facilitates the grammar inference. When any of these factors is not held, it will be better to use a prototype-based descriptor.

A prototype is a class representative usually represented by using strings and graphs. Using the graph theory (graph and subgraphs isomorphisms), it is possible to compare and to classify shapes that can even be partially occluded. The use of prototypes, instead of all graph descriptors, reduces the complexity of matching algorithms, but the computation of graph prototypes may be computationally expensive also. However, graph prototypes are computed only once during the learning phase, and matching is performed each time at the query level. There are several definitions of prototypes and researches in learning graph prototypes are still a subject of interest. For instance, a recent work [65] has proposed to use a genetic algorithm for learning graph prototypes (generalized median set, generalized discriminative set).

In addition to the above, the use of embedding techniques (see section “[Embedding Methods](#)”) allows to replace the computation of the graph-edit distance by the computation of a  $L_p$  distance in a  $n$ -dimensional space, thus reducing the computational complexity. The most representative graph structures used in symbol recognition are *labeled graphs*, *attributed graphs*, and *associative graphs*.

A **labeled graph** is a set of nodes, edges, and labeling functions. It is one of the simplest graph structures still used which can be constructed from a graphic

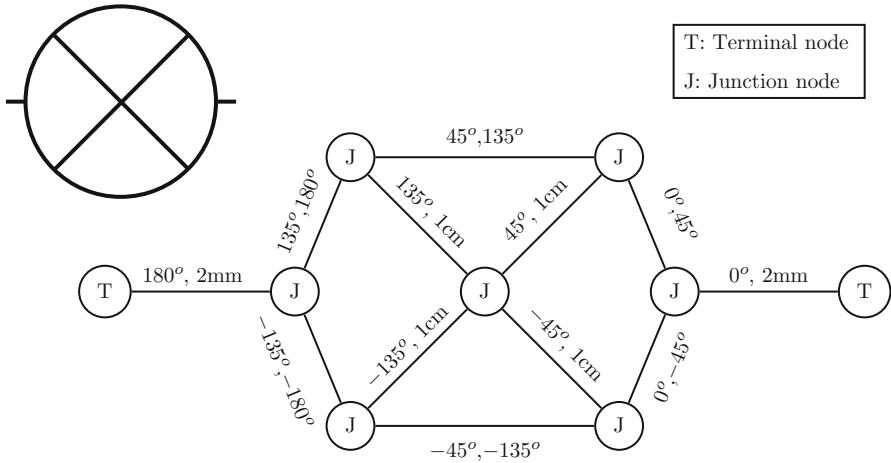
document after a vectorization process. The formal definition of a labeled graph can be found in any structural pattern recognition textbook with slight differences in notation and names. However, the set of nodes is usually composed of vectors, which play the role of *primitives*, and the set of edges is composed of pairs of nodes representing touching vectors. The definition of the labeling function depends on each particular method. It has been shown before that a grammar was used in 1969 to create the PDL. Later on, labeled graphs were used as a basis to build more sophisticated representations like, for instance, attributed graphs. In 1999, a labeled graph, called “shock graph,” was proposed in [72]. Here, the set of nodes is composed of terminal and junction points obtained after applying the MAT to symbols.

An **attributed graph** is a labeled graph (Fig. 16.6) with two more functions that assign a set of attributes to nodes and edges [11]. Graphs without attributes represent the structural information of symbols, while attributes add semantic information for the interpretation of schematic diagrams. In addition to symbol recognition, attributed graphs have also been used in other related applications such as handwriting recognition (see ▶ Chaps. 11 (Handprinted Character and Word Recognition) and ▶ 12 (Continuous Handwritten Script Recognition)) and general structural pattern recognition (see ▶ Chap. 15 (Graphics Recognition Techniques)). A particular case of attributed graphs is the RAG where the minimal closed loops (regions) are extracted within their adjacency relations [48].

An **associative graph** is a different graph-based representation. Each node is an association between local descriptions of a symbol model and partial descriptions of the document being processed. Then, each association is evaluated by a matching function which is defined locally. Therefore, in this representation, two nodes are connected if the matching value is below a preset threshold. With this representation, graphs do not represent simple primitive connections but incorporate information related to the sought primitive symbol models. This representation, which provided a definition of associated graph for trees, was used [22, 62] for comparing shock graphs.

Regardless of the graph structure used for symbol description, structural relations can be grouped taking into account the number of primitives:

- **Unitary** relations are commonly used as node attributes in graphs. For instance, attributed graphs based on the MAT have been proposed in [24, 72]. Another set of structural information that is classically used is the area, perimeter, eccentricity, length, and the number of holes of a region.
- **Binary** relations are the most used set to capture structural information from shapes including for instance, parallelism, angle of intersection, or inclusion between lines [59]. These relations are classically represented by means of edge attributes in graph structures or in a constraint set for adjacency grammars [54].
- **Ternary** relations are less often used than binary ones but may be more important in some applications like text detection. For instance, a ternary descriptor in a Markov random field was proposed in [83] to measure the text alignment using adjacent regions as primitives.



**Fig. 16.6** Example of structural descriptor based on an attributed graph representation

## Recognition Methods

In 50 years of publications in the field of pattern recognition, many textbooks [8, 27] have been proposed. While some of them are dedicated to specific techniques [31], many others have been applied to symbol recognition. Therefore, a thorough review of them is not only unworkable but is also beyond the scope of this chapter. In this perspective, only the most used and well-known methods in the field are discussed in this section.

Pattern recognition process is divided into two stages: feature extraction and classification. As seen, feature extraction involves the construction of symbol descriptors which can be pixel or structural, invariant or not to transformations. Classification indicates the set of methods that will allow to recognize symbols. In general, the majority of symbol recognition methods fit within supervised learning methods (i.e., classification), and therefore, a set of learning symbols is given.

When the training set is lacking or there is no need for supervised learning, the classification step reduces to mere calculation of distance or similarity measure between symbols to be recognized and a set of models. Therefore, the first part of this section will be first dedicated to similarity and distance measures for both structural and statistical methods. The computational cost is one of the motivation for the introduction of embedding and kernel techniques that allow to move from a graph description to a feature one. The second part gives an overview of the respective techniques in structural and statistical classification that have been used to recognize symbols.

## Distance and Similarity Measures

The easiest way to recognize a symbol is to compare it to a reference set, namely, models or prototypes, and assign to it the label of the most similar model. If the set of models is not very large, this comparison can be done sequentially using a *similarity* measure. A variety of similarity measures such as distances, correlation, inner product, trigonometric functions, and integral operators, all of which can be applied to symbols recognition, can be found in the literature.

Generally speaking, any functional defined on two elements that returns a scalar value can be interpreted as a similarity measure. Of course, it is preferable that this value has a meaning. An example of a similarity measure used in some symbol recognition methods is the Kullback–Leibler (KL) divergence, which is used as a measure for comparing two probability function densities  $q$  and  $p$  [86].

A special case of similarity measure is the distance, or metric. A set  $X$  with a distance  $d$  is called a metric space. From a formal viewpoint, if  $d$  is a distance, a set of metric spaces properties can be directly applied. The definition of the distance that one can find in any book of elementary geometry verifies the three well-known properties: positivity, symmetry, and triangle inequality. Examples of the most common distances are:

1. Real vectorial space  $\mathbb{R}^n$  with any of the  $L_p$  distance:  $d(x, y) = (\sum_{i=1}^n (x_i - y_i)^p)^{1/p}$ . For  $p = 1, 2$ , and  $\infty$ , it is, respectively, the *Manhattan*, the *Euclidean*, and the *supremum* distance. The sum operator is replaced by the max operator for the supremum distance.
2. The space of real functions  $L^p(\mathbb{R})$  composed of  $p$ -integrable functions:  $d(f, g) = (\int_{\mathbb{R}} (f(x) - g(x))^p dx)^{1/p}$ . For  $p = 1$ , it is the Banach space and for  $p = 2$  the Hilbert space.

In general, for any vectorial space with norm  $N$ , a distance is defined as  $d(x, y) = N(x - y)$ . When the chosen symbol descriptor is a feature vector, it can be useful to consider one of the distances in the previous examples. In such case, the feature vector is embedded into a vectorial space of finite dimension  $n < \infty$  where all distances are topologically equivalent. That is, given two distances  $d_1$  and  $d_2$  defined in the metric space  $X$ , for all values  $x$  and  $y$ , two real positive values  $A$  and  $B$  exist such that

$$Ad_1(x, y) \leq d_2(x, y) \leq Bd_1(x, y).$$

In practice, this means that given a symbol described by means of feature vectors, regardless of the distances from the Example 1 used, differences in classification rates are insignificant. In other words, the performance of a given feature vector will not change much if the Manhattan distance is used instead of the Euclidean, but the complexity will decrease.

If the  $L_p$  distances are the most widely used in finite-dimensional vector spaces, the edit distance, which is in a broad sense a similarity measure, is most often used to compare structural descriptors. It was initially defined to compare strings

and, later, extended to trees and graphs. The distance calculation is obtained by adding the costs of edit operations: insertion, deletion, and substitution needed to transform one string to another. The costs associated with each operation depend on the application, and if they are chosen properly, the edit distance is a true distance which satisfies the three properties required for a distance.

The edit distance is a measure that is robust to errors obtained during the extraction of primitives, but it is computationally expensive to be calculated accurately. To overcome this problem, an algorithm allowing an estimation of this distance by means of a bipartite graph was proposed in [66].

## Embedding Methods

The goal of kernel and embedding methods is to apply statistical methods to structural descriptors. The reason here is twofold. On the one hand, they take benefit from structural descriptors, which allow a richer representation of symbols by using graphs and trees than feature vectors. On the other hand, they extend the range of classification methods that can be used in classification problems and reduce the order of complexity of some operations, for example, the calculation of the *generalized median graph* [30].

Embedding methods are categorized formally as implicit or explicit. Explicit methods transform a graph into a feature vector. Thus, we can apply any statistical method: dimension reduction such as PCA by Fisher's discriminant analysis and classifiers such as KNN, boosting, neural network, and SVM. In all cases, the difficulty of embedding methods is to find suitable embedding functions.

Rather than seeking explicit transforms, implicit embedding is based on graph kernels. A kernel is a bivariate function that performs two operations at once. It first embeds structural descriptors into a vectorial space and then performs the dot product in such a space. The advantage of using kernel functions is that the embedding transformation is not needed to be known and it is much easier to define kernel functions than embedding functions. Further details on how to apply this framework for graphs are found in [12, 13, 58].

## Structural Classification

Classification methods with structural descriptors consist mostly in finding substructures in global representations of documents. One advantage of these methods, in contrast to statistical ones, is that they do not require a learning phase. However, an expert is needed to set the parameters and the heuristics to have good performance, either in execution time or recognition rate. Basic programming techniques such as dynamic programming [5, 14] or branch-and-bound techniques [48] were applied for searching subgraphs. For an overview of these algorithms, not only for symbol recognition but also for any field of applications in structural pattern recognition, some overviews can be found in [13, 18].

Ideally, matching algorithms look for exact matches between the object to be recognized and a list of known patterns and models. This means that, if data are represented by graphs, both structures must have at least the same number of nodes. On the contrary, in statistical methods distances between two feature vectors are required to have the same dimension. The first component of the first vector has to be compared to the first component of the second vector and so on until last component. In contrast, in structural approaches, there is no canonical order of nodes, and a priori all nodes are compared between them, making sure that all relations between nodes of the first graph are the same as the defined relations between nodes in the second graph. The complexity of these algorithms, in the worst case, grows exponentially with the number of nodes. This, in practice, makes these approaches intractable, especially when graphs have many nodes. Matching algorithms seek to perform these searches in a more intelligent way, with heuristics to reduce the search space. Structural descriptors obtained after a feature extraction process are not free of errors. Thus, two descriptors extracted from different images of the same object can have different representations in the number of nodes and edges. Matching algorithms have to be able to deal with this source of errors in descriptors, and therefore, the graph matching problem in symbol recognition is rather a problem of subgraph matching. Furthermore, a technique to provide error tolerance due to the primitive extraction process is achieved through the edit distance. Other possibilities are relaxation or elastic techniques [22] and active contours [77].

Finally, relations between nodes of trees and graphs can also be represented by adjacency matrices. The values of these matrices depend on the type of trees or graphs, but in any case they are real or even complex numbers. Thus, eigenvalues and eigenvectors of these matrices are computed to compare two different graphs [72]. One advantage is that the order of complexity of the matching algorithm is similar to the complexity of any given distance but graphs should have the same number of nodes and two similar eigenvectors do not necessarily correspond to similar graphs.

## Statistical Classification

Statistical methods use information from labeled data to learn classifiers for symbol recognition purposes. The aim of these methods is to learn decision rules to classify with the minimum possible risk of being wrong. In its simplest formulation, the decision rule is defined from the a posteriori probabilities of classes. That is, it maximizes the a posteriori probability (MAP rule) that the class is  $\omega_m$  given a descriptor  $x$ :

$$P(y = \omega_m|x) > P(y = \omega_p|x) \text{ for all } p \neq m. \quad (16.1)$$

This formulation is useful for a classifier, the response of which approximates the a posteriori probabilities of classes. In contrast, for other classifiers it is necessary to introduce the concept of loss function and risk of error:

$$R(y = \omega_p | x) = \sum_q \mathcal{L}(\omega_p, \omega_q) P(\omega_q | x) \quad (16.2)$$

A loss function,  $\mathcal{L}$  evaluates the *loss* suffered when a mistake in classification is made. When the loss function is 0/1, MAP rule is recovered, Eq.(16.1). The *risk error*  $R(y|x)$  is an operator that indicates the risk of error given a descriptor  $x$ , and it is obtained from the loss function and the a posteriori probability of the class. In symbol recognition, the classifier traditionally used is the nearest neighbors. Other classifiers such as the  $k$ -th nearest neighbors (KNN), support vector machine (SVM), boosting methods, or genetic algorithms become to be used since benchmarks are available.

Thus, the KNN is the simplest and most used classifier in symbol recognition. The distance between the unknown symbol and all the elements of the data set is computed, and the labels of the closest elements are counted. Then, the label of the majority class is assigned to the unknown symbol. When  $k$  increases the ratio obtained by dividing the number of votes with  $k$  estimates the posteriori probability [41].

Other state-of-the-art classifiers, like boosting and SVM methods, are two-class classifiers that have been extended to multi-class classifiers to perform symbol recognition. That extension will depend on the strategy used to learn. In classical recognition problems, one possibility is to follow a strategy *1 vs. 1* in which we take  $m - 1$  classifiers for each of the  $m$  classes. The predictions of these  $m - 1$  classifiers can be combined using any of the techniques described in [2]. Another possibility is to use a *1 vs. all* strategy in which  $m$  classifiers, one per class, are trained. In this case, one class is composed of all the elements of the same class (*positive class*), and the other class (*negative class*) is composed of elements taken randomly from the other classes.

SVM are a family of classifiers looking for the optimal hyperplane separating two classes. When this hyperplane does not exist (because classes overlap), the least bad hyperplane that best separates two classes is sought. To find the optimal solution, the problem is formulated as a quadratic optimization problem with boundary constraints. The method of Lagrange multipliers is used to determine the vector  $w$ , perpendicular to the separation hyperplane, and a scalar  $b$  which is the offset of the hyperplane from the origin. Hence, for classifying a new element  $x$ , the sign operator is applied:

$$y = \langle w, x \rangle + b \quad (16.3)$$

To better fit data, instead of looking for planes, surfaces are sought. In other words, data are transformed so that the optimal surface which separates two classes becomes flat. That's the idea behind the *kernel trick* and which has inspired kernel methods for structural descriptors. In this case, the dot product of Eq.(16.3) is replaced by a kernel function  $k$  which is in charge of performing the dot product of  $x$  and  $w$ , transformed by the embedding function  $\phi$ , in another vectorial space, usually of higher dimension than the original data:

$$y = k(w, x) + b = \langle \phi(w), \phi(x) \rangle + b \quad (16.4)$$

Choosing the appropriate kernel function is one of the difficulties to solve. Examples of kernel functions are polynomial of order  $p$ ,  $(\langle x, y \rangle + 1)^p$ ; radial basis functions,  $\exp\left(-\frac{\|x-y\|^2}{2\sigma^2}\right)$ ; and the hyperbolic tangent,  $\tanh(\kappa\langle x, y \rangle - \delta)$ . Once the kernel function is chosen, optimal parameters can be obtained by a cross validation on the training set. However, learning SVM for large data sets (millions of support vectors) is still a challenging problem [26].

Boosting methods seek to build up a good classifier reinforcing the learning of what is called a *weak classifier*. A weak classifier makes slightly better than the object class chosen randomly. That is to say, for a two-class classification problems, it is a classifier with a recognition rate slightly more than 50 %. Boosting methods assigns a weight to each learning sample, which is used to learn a new classifier. This new classifier is also evaluated so that the weights of samples well classified are reduced while the weight of misclassified samples is increased. The resulting classifier is an additive model consisting of the sum of all weak classifiers  $\{f_p\}$  that have been trained so far, and  $c_p$  is a weight that is obtained from the error of classification for  $f_p$  on the training set:

$$y = \sum_p c_p f_p(x) \quad (16.5)$$

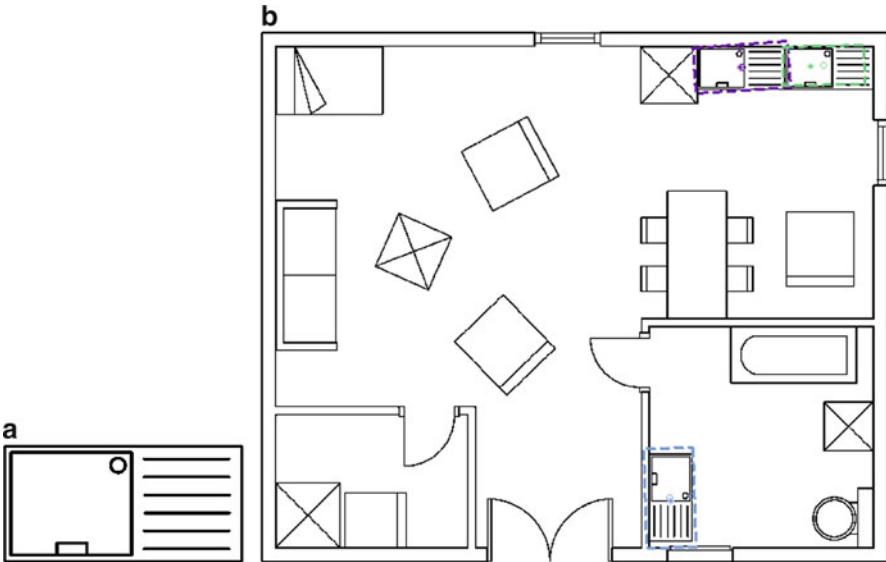
Both SVM and boosting methods have proven to be good classifiers in many applications of pattern recognition and are often used and proposed as reference classifiers too. For instance, directional features were used in [52] for SVM classification. A descriptor based on a combination of contourlets and Fourier coefficients, which were trained using an AdaBoost classifier, was proposed in [15]. However, recent experiments using these two kinds of classifiers on zoning-like descriptors show some degree of success whose differences are not statistically significant [29].

---

## Symbol Spotting

All symbol descriptors seen so far assume that symbols are cleanly segmented. However, when symbols are embedded in documents, the well-known paradox rises: to correctly recognize the symbols, one should be able to segment the input data, but to correctly segment them, one need to recognize the symbols! Generally speaking, symbol spotting is a kind of “strategy” used to break down this paradox since one does not try to recognize a symbol in a document as a whole but as a set of primitives.

In a spotting system, the user submits a query he or she wants to retrieve from the document database, and the system retrieves zones in the documents that are likely to contain the query (see Fig. 16.7). The query defined by the user is usually a cropped image of a document or a hand-sketched one belonging to the database. The retrieval stage is done online and, of course, short delay of response is expected.



**Fig. 16.7** (a) The query. (b) Retrieved zones in the document that are likely to contain the query

The analogy to the computer vision community is that a spotting system looks like a *content-based image retrieval* (CBIR) application that focuses on subpart retrieval of the image. The main difference here is that technical documents have different photometrical conditions. Documents are usually in black and white and descriptors based on color or texture features cannot be used. Thus, although the underlying strategies are somewhat similar, the features are different since the structural representation is very important for a document.

Symbol spotting is an emerging topic and few works have been proposed so far. As a manuscript on symbol spotting in digital libraries [68] has been published recently, the state of the art will be short in this section. Broadly speaking, symbol spotting methods are decomposed into two main steps. The first one describes the document and is related to the feature extraction method (pixel or structural). This description could be done locally with focus on points or regions of interest, or globally through a structural representation. The second one is the decision step that occurs in the retrieval stage. Since the query is decomposed into a set of primitives like the document, a strategy is needed to organize these primitives to generate a list of location hypotheses in the document that are likely to contain the queried symbol. If the number of symbols and documents is high, a sequential search is not realistic. In this perspective, hashing structures have shown their efficiency in quickly generating hypotheses. In [68], a relational indexing scheme is proposed where a hash table stores the adjacency matrix of proximity graphs and a hash function is designed for feature vectors allowing, in fact, to combine numerical and structural descriptions of symbols. Other structures have been proposed based on

---

hierarchical organization or inverted file combined with the vectorial model. These structures, however, require more time complexity. As in CBIR applications, the list of hypotheses is ranked from the most to the least likely based on similarity measures or voting strategies.

When the number of documents is small and the same symbols have a low variability in size and orientation, a brute force solution could be applied. In this case the spotting works like a correlation filter. The document is decomposed into regions defined on either the connected components, a grid partition, or a sliding window, and then the similarity (like a correlation function) with the query is measured. For instance, a method to localize mechanical objects in grayscale images was proposed in [50]. The approach does not require any preprocessing step and is based on a pyramidal decomposition. The first step is to search for potential positions of the query symbol in the document as maxima of a normalized correlation surface. These positions are then propagated level by level towards the lowest level of the pyramid in order to precisely locate the corresponding objects. The approach defined in [29] is based on a sliding window strategy and a descriptor (circular blurred shape model) describing the spatial arrangement around a centroid point of the object region in a correlogram structure. These methods based on correlation principle locate accurately positions of a query symbol and are robust enough to be applied on real applications, but the invariance to symbol variations (size and rotation) remains a bottleneck. In this perspective, to avoid the scanning of the whole document, a variant is to focus on interest points, usually corner points, and to use these points to locally describe the document by means of visual words, as it is done in information retrieval where documents are indexed by textual words, or to organize these points spatially to validate hypotheses in order to find a symbol.

Lines remain one of the most used and simplest primitives [60]. An extension to polyline primitives was proposed to take into account circular shapes and segment fragmentation due to the sensitivity to noise of raster-to-vector algorithms. In [68, 69], regions are considered as being circular polylines and symbols are supposed to be defined by closed contours but it is not necessary the case in electrical drawings. Some structural approaches encode primitives in terms of strings [69] or dendrograms [87] representation; however, graphs remain the most popular data structure since it offers a more powerful representation to encode structural relations between different parts of symbols. In some approaches, graph is a means to extract a vectorial signature [68] using a windowing or bucket decomposition of the document for correlation filter. The main limitation of these approaches is that they are not robust to line fragmentation. They could, however, be used to pre-segment the document into zones of interest, which offer a richer description than interest points. Others [48] consider graph as a whole and the spotting is seen as a problem of subgraph matching between the query and document, both represented by graphs. However, these approaches suffer from the classical drawbacks of subgraph isomorphism in pattern recognition such as tolerance to noise and high time complexity, as discussed in section “[Structural Classification](#).”

Spotting methods are strongly related to the adopted symbol descriptors as well as the subsequent treatments, since the heart of the methods relies on the

description of document. Usually the used primitives are either pixel or structural. Pixel descriptors are less dependent on the quality of prior segmentations but are less robust to partial occlusions and provide more false responses because they do not take into account the structural configuration of a symbol. In contrast, structural descriptors offer a more powerful description of the document (less false alarms are generated), and they are more flexible to occlusions but they are dependent on the quality of the preprocessing step needed to feed the structural descriptor. Therefore, the compactness and precision of the representation are very important because they have an impact on the performance of the spotting method, on the indexing efficiency of the system and on the delay of response.

---

## Conclusion

The field of symbol recognition has reached its maturity with many descriptors which have been proposed so far. Some descriptors are variants of the previous ones, and several descriptors have reached very high recognition rate even when symbols are noisy, partially occluded, and transformed under similarities. However, the problem of obtaining descriptors robust to severe deformations remains a topic of interest. To achieve a high level of performance, several assumptions on symbols related to their number of models, variability, and segmentation quality are required. Thus, some specific problems still remain for future research.

When the number of symbol models or the complexity of the symbols increases, the confusion between different classes increases. For instance, in an aircraft electrical wiring diagrams, symbols are numerous and are complex entities that associate a graphical representation, a number of connection points, and associated text annotations. The main challenge here is to discriminate symbols not on their global shape but on small details (e.g., the number of connections). Combining outputs of classifiers, descriptors, or selecting features is one of the strategies used to improve the recognition rate. Although it is relatively easy to adapt these strategies to statistical descriptors, the extension to structural ones remains difficult. Recent works on embedding methods [12, 30] have been proposed with promising results.

Performance evaluation campaigns on massive data collection to test the scalability of the proposed approaches are lacking (see ►Chap. 30 (Tools and Metrics for Document Analysis Systems Evaluation)). However, contrary to a huge number of documents managed, technical documents are rare and few data are available as benchmarks to the image analysis community. Several efforts have been made to create database in graphical contest, but these databases are often small and synthetic, in spite of the effort to generate synthetic documents that look like real ones [23]. The probable reason is the difficulty in getting real documents and the cost to associate the ground truth to real documents since the amount of time to generate it manually would require a huge effort.

With the growing popularity of digital input devices like tablet PCs and smartphones, there is an increasing interest in designing systems that can recognize automatically hand-sketched or camera-captured symbols

(see ►Chap. 28 (Sketching Interfaces)). These types of symbols are warped and thus have a high variation and distortion. Even if general de-warping document methods can be applied to the particular domain of symbols, new descriptors robust to shape deformations or affine transformations are needed.

The spotting methods are still in their early years, the proposed methods so far are only applied on synthetic documents. Spotting methods need to reach a high level of maturity to consider applications to real documents. Methods combining hash tables and voting strategies seem to be efficient in terms of time and indexing complexity when we face large collections of documents. Moreover, although performance measures [67] have been proposed for symbol spotting methods, these measures are more dedicated to synthetic documents, and their applicability on real documents has not been really verified. In addition, we believe that even if symbol spotting is closely related to symbol recognition, it can be seen as a particular case of document mining and the next step to symbol spotting would be new methods for symbol mining, and there are still many things to do to reach this goal. In this vein, recent works [20] use an original adaptation of a Galois lattice which has shown good performances in the field of data mining. The use of Galois lattices is a means to get a “symbolic representation” from a numeric one in the perspective to narrow the semantic gap.

---

## Cross-References

- [Analysis and Interpretation of Graphical Documents](#)
- [Asian Character Recognition](#)
- [Continuous Handwritten Script Recognition](#)
- [Document Creation, Image Acquisition and Document Quality](#)
- [Graphics Recognition Techniques](#)
- [Imaging Techniques in Document Analysis Processes](#)
- [Language, Script, and Font Recognition](#)
- [Middle Eastern Character Recognition](#)
- [Sketching Interfaces](#)
- [Text Segmentation for Document Recognition](#)
- [Tools and Metrics for Document Analysis Systems Evaluation](#)

---

## References

1. Adam S, Ogier MJ, Cariou C, Mullot R, Labiche J, Gardes J (2000) Symbol and character recognition: application to engineering drawings. *Int J Doc Anal Recognit* 3:89–101
2. Allwein EL, Schapire RE, Singer Y (2000) Reducing multiclass to binary: a unifying approach for margin classifiers. *J Mach Learn Res* 1:113–141
3. Asada H, Brady M (1986) The curvature primal sketch. *IEEE Trans PAMI* 8(1):2–14
4. Attneave F, Arnoult MD (1956) The quantitative study of shape and pattern perception. *Psychol Bull* 53(6):452–471

5. Bartolini L, Ciaccia P, Patella M (2005) Warp: accurate retrieval of shapes using phase of fourier descriptors and time warping distance. *IEEE Trans PAMI* 27(1):142–147
6. Belongie S, Malik J, Puzicha J (2002) Shape matching and object recognition using shape contexts. *IEEE Trans PAMI* 24(24):509–522
7. Berretti S, del Bimbo A, Pala P (2000) Retrieval by shape similarity with perceptual distance and effective indexing. *IEEE Trans Multimed* 2(4):225–239
8. Bishop C (2006) Pattern recognition and machine learning. Springer, New York
9. Blum H (1967) A transformation for extracting new descriptors of shape. In: Wathen-Dunn W (ed) Models for the perception of speech and visual form. MIT, Cambridge, pp 362–380
10. Bomba JS (1959) Alpha-numeric character recognition using local operations. Papers presented at the eastern joint IRE-AIEE-ACM (Eastern) computer conference IRE, 1–3 Dec 1959. ACM, New York, pp 218–224
11. Bunke H (1982) Attributed programmed graph grammars and their application to schematic diagram interpretation. *IEEE Trans PAMI* 4:574–582
12. Bunke H, Riesen K (2011) Improving vector space embedding of graphs through feature selection algorithms. *Pattern Recognit* 44(9):1928–1940. Computer analysis of images and patterns
13. Bunke H, Riesen K (2011) Recent advances in graph-based pattern recognition with applications in document analysis. *Pattern Recognit* 44(5):1057–1067
14. Burr DJ (1981) Elastic matching of line drawings. *IEEE Trans PAMI* 3(6):708–713
15. Chen GY, Kégl B (2010) Invariant pattern recognition using contourlets and adaboost. *Pattern Recognit* 43(3):579–583
16. Chen GY, Bui TD, Krzyzak A (2009) Invariant pattern recognition using radon, dual-tree complex wavelet and fourier transforms. *Pattern Recognit* 42(9):2013–2019
17. Chong C-W, Raveendran P, Mukundan R (2003) A comparative analysis of algorithms for fast computation of zernike moments. *Pattern Recognit* 36:731–742
18. Conte D, Foglia P, Sansone C, Vento M (2004) Thirty years of graph matching in pattern recognition. *Int J Pattern Recognit Artif Intell* 18(3):265–298
19. Costa LdFR, Cesar RM Jr (2009) Shape analysis and classification, 2nd edn. CRC Press, Boca Raton, Florida
20. Coustaty M, Bertet K, Visani M, Ogier J-M (2011) A new adaptive structural signature for symbol recognition by using a galois lattice as a classifier. *IEEE Trans Syst Man Cybern* 41(4):1136–1148
21. Das M, Paulik MJ, Loh NK (1990) A bivariate autoregressive technique for analysis and classification of planar shapes. *IEEE Trans PAMI* 12(1):97–103
22. Davis LS (1979) Shape matching using relaxation techniques. *IEEE Trans PAMI* 1(1):60–72
23. Delalandre M, Valveny E, Pridmore T, Karatzas D (2010) Generation of synthetic documents for performance evaluation of symbol recognition and spotting systems. *Int J Doc Anal Recognit* 13(3):187–207
24. Di Ruberto C (2004) Recognition of shapes by attributed skeletal graphs. *Pattern Recognit* 37:21–31
25. Dinneen GP (1955) Programming pattern recognition. In: Proceedings of the western joint computer conference, AFIPS'55 (Western), Los Angeles, 1–3 Mar 1955. ACM, New York, pp 94–100
26. Dong J-X, Krzyzak A, Suen CY (2005) Fast svm training algorithm with decomposition on very large data sets. *IEEE Trans PAMI* 27(4):603–618
27. Duda RO, Hart PE, Stork DG (2000) Pattern classification. New York
28. El Rube M, Ahmed I, Kamel M (2006) Wavelet approximation-based affine invariant shape representation functions. *IEEE Trans PAMI* 28(2):323–327
29. Escalera S, Fornés A, Pujol O, Lladós J, Radeva P (2011) Circular blurred shape model for multiclass symbol recognition. *IEEE Trans Syst Man Cybern B Cybern* 41(2):497–506
30. Ferrer M, Valveny E, Serratosa F, Riesen K, Bunke H (2010) Generalized median graph computation by means of graph embedding in vector spaces. *Pattern Recognit* 43(4):1642–1655

31. Flusser J, Zitova B, Suk T (2009) Moments and moment invariants in pattern recognition. Wiley, Chichester
32. Fornés A, Lladós J, Sánchez G, Karatzas D (2010) Rotation invariant hand-drawn symbol recognition based on a dynamic time warping model. *Int J Doc Anal Recognit* 13:229–241
33. Forsyth D, Mundy JL, Zisserman A, Coelho C, Heller A, Rothwell C (1991) Invariant descriptors for 3d object recognition and pose. *IEEE Trans PAMI* 13(10):971–991
34. Freeman H (1961) On the encoding of arbitrary geometric configurations. *IRE Trans Electron Comput* 10(2):260–268
35. Fu K-S, Bhargava BK (1973) Tree systems for syntactic pattern recognition. *IEEE Trans Comput* 22(12):1087–1099
36. Fujisawa H, Liu C-L (2003) Directional pattern matching for character recognition revisited. In: 7th international conference on document analysis and recognition, Washington, DC. IEEE Computer Society, p 794
37. Groen F, Sanderson A, Schalag F (1985) Symbol recognition in electrical diagrams using probabilistic graph matching. *Pattern Recognit Lett* 3:343–350
38. Hoang T-V, Tabbone S (2012) The generalization of the R-transform for invariant pattern recognition. *Pattern Recognit* 45(6):2145–2163
39. Hoang T-V, Tabbone S (2012) Invariant pattern recognition using the rfm descriptor. *Pattern Recognit* 45(1):271–284
40. Hu M-K (1962) Visual pattern recognition by moments invariants. *IRE Trans Inf Theory* 8:179–187
41. Jain AK, Duin RP, Mao J (2000) Statistical pattern recognition: a review. *IEEE Trans PAMI* 22(1):4–37
42. Josep L, Gemma S (2004) Graph Matching Versus Graph Parsing In Graphics Recognition - A Combined Approach. *IJPRAI* 18(3):455–473
43. Kadyrov A, Petrou M (2001) The trace transform and its applications. *IEEE Trans PAMI* 23(8):811–828
44. Kashyap R, Chellappa R (1981) Stochastic models for closed boundary analysis: representation and reconstruction. *IEEE Trans Inf Theory* 27(5):627–637
45. Khalil MI, Bayoumi MM (2001) A dyadic wavelet affine invariant function for 2d shape recognition. *IEEE Trans PAMI* 23(10):1152–1164
46. Kimura F, Wakabayashi T, Tsuruoka S, Miyake Y (1997) Improvement of handwritten japanese character recognition using weighted direction code histogram. *Pattern Recognit* 30(8): 1329–1337. Oriental character recognition
47. Lin C-S, Hwang C-L (2010) New forms of shape invariants from elliptic fourier descriptors. *Pattern Recognit* 20(5):535–545
48. Lladós J, Martí E, Villanueva JJ (2001) Symbol recognition by error-tolerant subgraph matching between region adjacency graphs. *IEEE Trans PAMI* 23(10):1137–1143
49. Lowe DG (2004) Distinctive image features from scale-invariant keypoints. *Int J Comput Vis* 60(2):91–110
50. MacLean W, Tsotsos K (2008) Fast pattern recognition using normalized grey-scale correlation in a pyramid image representation. *Mach Vis Appl* 19(3):163–179
51. Mallat S (1999) A wavelet tour of signal processing. Academic Press, San Diego, California
52. Malon C, Uchida S, Suzuki M (2008) Mathematical symbol recognition with support vector machines. *Pattern Recognit Lett* 29(9):1326–1332
53. Manjunath B, Salembier P, Sikora T (2002) Introduction to MPEG-7: multimedia content description interface.
54. Mas J, Lladós J, Sánchez G, Jorge JAP (2010) A syntactic approach based on distortion-tolerant adjacency grammars and a spatial-directed parser to interpret sketched diagrams. *Pattern Recognit* 43(12):4148–4164
55. Messmer BT, Bunke H (1998) A new algorithm for error-tolerant subgraph isomorphism detection. *IEEE Trans PAMI* 20(5):493–504
56. Mokhtarian F, Abbasi S (2002) Shape similarity retrieval under affine transforms. *Pattern Recognit* 35(1):31–41

57. Mundy JL, Zisserman A (1992) Geometric invariance in computer vision. MIT, Cambridge, pp 02142
58. Neuhaus M, Bunke H (2007) Bridging the gap between graph edit distance and kernel machines. World Scientific Publishing, River Edge
59. Park JH, Umm BS (1999) A new approach to similarity retrieval of 2-d graphic objects based on dominant shapes. *Pattern Recognit Lett* 20(6):591–616
60. Park GB, Lee MK, Lee US, Lee HJ (2003) Recognition of partially occluded objects using probabilistic arg (attributed relational graph)-based matching. *Comput Vis Image Underst* 90(3):217–241
61. Pavlidis T (1978) Survey: a review of algorithms for shape analysis. *Comput Graph Image Process* 7(7):243–258
62. Pelillo M, Siddiqi K, Zucker SW (1999) Matching hierarchical structures using association graphs. *IEEE Trans PAMI* 21(11):1105–1120
63. Pratt I (1999) Shape representation using fourier coefficients of the sinusoidal transform. *J Math Imaging Vis* 10:221–235
64. Terrades OR, Valveny E (2006) A new use of the ridgelets transform for describing linear singularities in images. *Pattern Recognit Lett* 27(6):587–596
65. Raveaux R, Adam S, Héroux P, Trupin E (2011) Learning graph prototypes for shape recognition. *Comput Vis Image Underst* 115(7):905–918. Special issue on graph-based representations in computer vision
66. Riesen K, Bunke H (2009) Approximate graph edit distance computation by means of bipartite graph matching. *Image Vis Comput* 27(7):950–959
67. Rusiñol M, Lladós J (2009) A performance evaluation protocol for symbol spotting systems in terms of recognition and location indices. *Int J Doc Anal Recognit* 12(2):83–96
68. Rusiñol M, Lladós J (2010) Symbol spotting in digital libraries. Springer, London
69. Rusiñol M, Lladós J, Sánchez G (2010) Symbol spotting in vectorized technical drawings through a lookup table of region strings. *Pattern Anal Appl* 13(3):321–331
70. Sekita I, Kurita T, Otsu N (1992) Complex autoregressive model for shape recognition. *IEEE Trans PAMI* 14(4):489–496
71. Shaw AC (1969) A formal picture description scheme as a basis for picture processing systems. *Inf Control* 14(1):9–52
72. Siddiqi K, Shokoufandeh A, Dickinson SJ, Zucker SW (1999) Shock graphs and shape matching. *Int J Comput Vis* 35(1):13–22
73. Swain MJ, Ballard DH (1991) Color indexing. *Int J Comput Vis* 7(1):11–32
74. Tabbone S, Wendling L, Salmon JP (2006) A new shape descriptor defined on the radon transform. *Comput Vis Image Underst* 102(1):42–51
75. Teh C-H, Chin RT (1988) On image analysis by methods of moments. *IEEE Trans PAMI* 10(4):496–513
76. Tieng QM, Boles WW (1997) Wavelet-based affine invariant representation: a tool for recognizing planar objects in 3d space. *IEEE Trans PAMI* 19(8):846–857
77. Valveny E, Martí E (2003) A model for image generation and symbol recognition through the deformation of lineal shapes. *Pattern Recognit Lett* 24(15):2857–2867
78. Wu X, Bhani B (1997) Gabor wavelet representation for 3-d object recognition. *IEEE Trans Image Process* 6:47–64
79. Yadava RB, Nishchala NK, Gupta AK, Rastogi VK (2007) Retrieval and classification of shape-based objects using fourier, generic fourier, and wavelet-fourier descriptors technique: a comparative study. *Opt Lasers Eng* 45(6):695–708
80. Yang S (2005) Symbol recognition via statistical integration of pixel-level constraint histograms: a new descriptor. *IEEE Trans PAMI* 27(2):278–281
81. Yap P-T, Paramesran R (2005) An efficient method for the computation of legendre moments. *IEEE Trans PAMI* 27(12):1996–2002
82. Zahn CT, Roskies RZ (1972) Fourier descriptors for plane closed curves. *IEEE Trans Comput* 21(3):269–281

83. Zhang DQ, Chang SF (2004) Learning to detect scene text using a higher-order MRF with belief propagation. In: Proceedings of the 2004 conference on computer vision and pattern recognition workshop (CVPRW'04). IEEE Comput Soc. Washington, DC, USA **6**:101–108
84. Zhang D, Lu G (2003) A comparative study of curvature scale space and fourier descriptors for shape-based image retrieval. *J Vis Commun Image Represent* 14(1):41–60
85. Zhang D, Lu G (2004) Review of shape representation and description techniques. *Pattern Recognit* 37:1–19
86. Zhang W, Wenyin L, Zhang K (2006) Symbol recognition with kernel density matching. *IEEE Trans PAMI* 28(12):2020–2024. Senior Member-Liu Wenyin
87. Zuwala D, Tabbone S (2006) A method for symbol spotting in graphical documents. In: Bunke H, Spitz A (eds) Document analysis systems VII. Volume 3872 of lecture notes in computer science. Springer, Berlin/Heidelberg, pp 518–528

## Further Reading

This chapter does not claim to be exhaustive, and therefore, interested readers are referred to broaden their knowledge with the following books [19, 68] for shape descriptions and symbol spotting considerations and [8, 27] for more global skills on shape classification and machine learning.

---

# Analysis and Interpretation of Graphical Documents

17

Bart Lamiroy and Jean-Marc Ogier

## Contents

Introduction.....	554
Graphical Document Analysis.....	554
History: Evolution of the Problem.....	555
Structure of This Chapter.....	556
State of the Art and Classification of Graphical Document Analysis.....	557
Overview of Graphical Document Analysis Problems.....	557
Classification of Graphical Document Analysis Systems.....	560
Relations Between Machine Perception and Image Analysis Systems.....	564
Examples of Analysis Approaches and Interpretation Knowledge Representations.....	565
Classical Strategies: Bottom-Up Approach.....	565
Knowledge-Based Approaches.....	571
Hybrid Approaches for Graphics Analysis.....	573
Discussion and Limitations.....	581
Conclusion.....	582
Recent Evolutions: Learning, Spotting, and Indexing for Navigation into Graphical Document Repositories.....	582
Challenges in Graphics Interpretation.....	586
Cross-References.....	587
References.....	588
Further Reading.....	590

---

B. Lamiroy (✉)

Université de Lorraine – LORIA, Vandœuvre-lès-Nancy Cedex, France  
e-mail: [Bart.Lamiroy@loria.fr](mailto:Bart.Lamiroy@loria.fr)

J.-M. Ogier

Université de La Rochelle, La Rochelle Cédex 1, France  
e-mail: [Jean-Marc.Ogier@univ-lr.fr](mailto:Jean-Marc.Ogier@univ-lr.fr)

**Abstract**

This chapter is dedicated to the analysis and the interpretation of graphical documents and, as such, builds upon many of the topics covered in other parts of this handbook. It will therefore not focus on any of the technical issues related to graphical documents, such as low-level filtering and binarization, primitive extraction and vectorization as developed in ►Chaps. 4 (Imaging Techniques in Document Analysis Processes) and ►15 (Graphics Recognition Techniques), or symbol recognition, for instance, as developed in ►Chap. 16 (An Overview of Symbol Recognition). These tools are put in a broader framework and threaded together in complex pipelines to solve interpretation questions. This chapter provides an overview of how analysis strategies have contributed to constructing these pipelines, how specific domain knowledge is integrated in these analyses, and which interpretation contexts have been contributed to successful approaches.

**Keywords**

Graphical document interpretation • Heuristics • Image analysis • Interpretation pipeline • Interpretation process • Interpretation systems • Knowledge management • Ontology

---

## Introduction

### Graphical Document Analysis

Graphical documents are basically documents containing a significant (if not exclusive) part of line drawings. They are usually considered a separate class between full text documents and photo-realistic documents. Many applications or research areas, related to the interpretation of documents in general, usually consider it good practice to segment composite documents into at least text, line drawings, and photo-realistic subparts, to which more specialized treatments are then applied.

There exists a very rich literature on projects and systems related to graphical document interpretation. Some of them are inspired from concepts which were developed in broader image interpretation problems. From a conceptual point of view, the scientific objectives of these systems consist in trying to implement generic and adaptable strategies, based on knowledge modeling, adaptive interpretation scenarios, etc. Most of the systems are related to projects within organizations managing huge amounts of graphical documents, and for which these documents are operational or organizational assets, the value of which needs to be optimized. Among the organizations/research groups who have presented some realistic systems, it is noteworthy to mention those projects related to the management of network maps and data, such as telephone networks, electric power grids, or waste water networks, for instance. Since efficiently operating these networks raises many management questions and is a major financial issue, many companies have investigated ways of leveraging the use of their digital maps with automated tools that are capable of extracting relevant interpretations.

While there are many references in the state-of-the-art literature to various domains, applications, and categories of documents, one of the most successful approaches concerns the area of electronic diagram interpretation for aircraft, done by Baum et al. at Boeing in the late 1990s. Their goal was to scan paper versions of wiring diagrams and other engineering drawings in order to convert them into a digital and operational maintenance tool that would allow hyper-referenced access to assembly parts, logic diagrams, etc. The illustrations in Fig. 17.2 were reported in [4] and subsequent publications and give an idea of how it was able to cross-reference text-based part numbers and graphical entities.

The main challenge of interpretation systems is to implement flexible and adaptive strategies, being able to produce knowledge compatible with its interpretation domain, and being able to automatically detect and solve semantic ambiguities.

## History: Evolution of the Problem

The history of automated document analysis and its related problems finds its origin and initial scope in the corporate business domain. In order to optimize information management in big public or private institutions, which was initially totally paper based, new solutions and document management applications were expected to emerge from focused research. In this context, during the last decades, mainly due to the huge increase and reduction of costs of storage capacities and to the continuous progress in image analysis techniques, there has been a considerable evolution in the various strategies of large institutions when dealing with document analysis problems.

Initially, because of the lack of relevant automatic interpretation systems, most of the organizations decided to only digitize their documents, in order to obtain a representation of data (and therefore, mistakenly, information) which could be easily stored, shared, duplicated, and transmitted and would therefore respond to elementary data management problems. This massive digitizing cannot be considered as part of document analysis per se, but it marks the beginning of an era where automatic document management will progressively attract more and more economic interest.

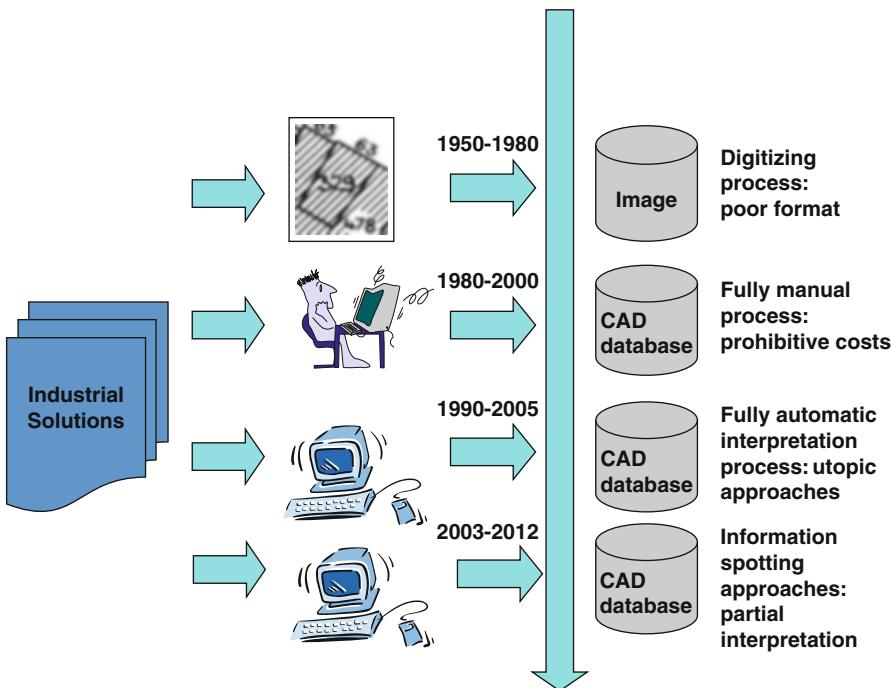
However, it became rapidly clear that considering digitization only was producing data with too poor a level of information, and most organizations decided it was necessary to improve the level of usability by trying to obtain higher-level interpretations from the digitized data by structuring the contained information through analysis processes. At first, regarding the fact that automatic processes were in their infancy, most of companies and organizations adopted full manual processes for reverse document interpretation. Many low-cost manual digitizing and interpretation projects were started in countries where human labor was cheap. With respect to the definitions and classifications mentioned in the previous section, this is an interesting benchmark, since it fully relies on human document analysis. It remains interesting to have a deeper look into the notions of context, interpretation, and analysis in this particular case. Indeed, the analysis is done by human collaborators who are not necessarily entirely knowledgeable of the

interpretation context, since they are externally hired to process the documents and provide the interpretation. This required the context to be documented and a quality process to be instituted in order to make sure that the produced interpretations were conforming to the expected context. However, facing prohibitive costs of the manual interpretation resources and considering quality problems related to human digitizing, many of them tried to implement interactive processes, coupling reliable image-processing tools and human correction interfaces. Then, during the 1990s full automatic interpretation systems were presented in the literature, based on complex approaches, integrating sophisticated strategies. The complexity of these approaches essentially came from the inability of the developers to fully capture the interpretation context, who therefore resorted to compensating this (often unconsciously) by embedding them into the algorithms themselves. Because of this, the produced analysis programs were often very focused on specific interpretation contexts, without offering any satisfactory hints to whether they could be adapted to other contexts without significant re-engineering. Considering that fully automatic systems that would also be generic and usable over a wide range of interpretation context were considered as quite Utopian by some authors, and given the fact that this seemed to be confirmed by the observation of the state of the art, a significant paradigm shift appeared suggesting to develop alternatives to full interpretation systems by only partially interpreting document contents and by using “spotting techniques” integrating “contextual information.” Generally speaking, the *information spotting* problem can be defined as the location of a set of regions of interest from a document image, which are likely to contain an instance of a certain queried object without explicitly recognizing it. The most famous applications of this kind of concepts are word spotting on manuscript documents on the one hand and symbol spotting in the context of graphical documents on the other hand. One of the main applications for information spotting methods is its use in large collections of documents. In a sense, this is a very pragmatic answer to the previously mentioned inability to capture interpretation context. Spotting techniques implicitly admit that full interpretation seems to be out of reach and focuses either on very generic (“low-level”) analyses or on very broad interpretation contexts, leaving it to a later stage process step to combine this partially complete information to achieve full interpretation.

This explains why more recent literature does much less focus on complete interpretation systems. Figure 17.1 illustrates these different alternatives, as well as corresponding milestones during the last five decades.

## Structure of This Chapter

As will become gradually clear through the development of this chapter, there is no precise or generally adopted approach to graphical document analysis. Furthermore, there is a significant overlap with the more general Machine Perception domain, and it will sometimes be necessary to digress and illustrate some approaches that are appropriate in that area.



**Fig. 17.1** Main periods of the graphic interpretation history

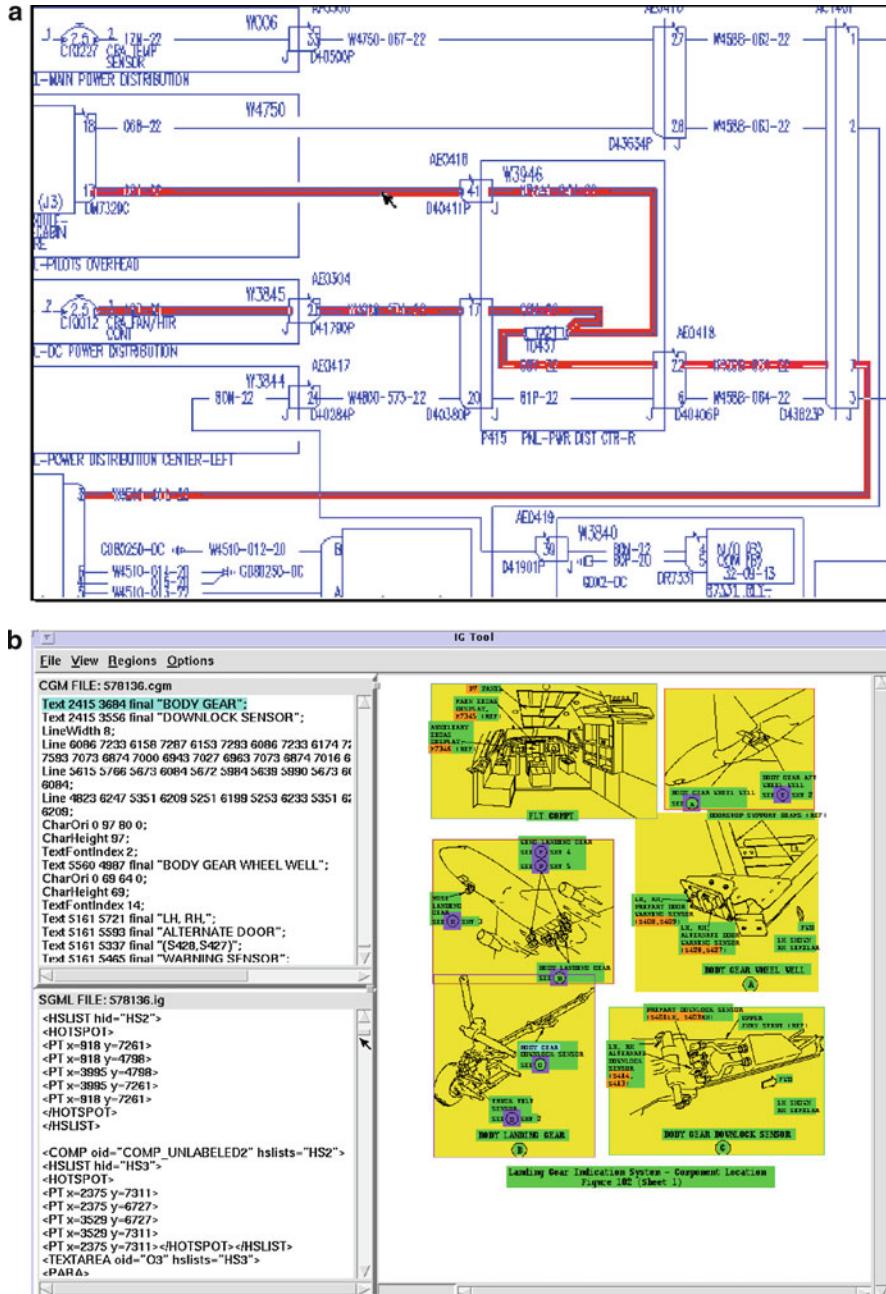
However, it is possible to identify broad categories of approaches to graphical document analysis. These are highlighted in section “[Structure of This Chapter](#).” Section “[Examples of Analysis Approaches and Interpretation Knowledge Representations](#)” will give a quick state-of-the-art overview of some of the most significant approaches for each category.

## State of the Art and Classification of Graphical Document Analysis

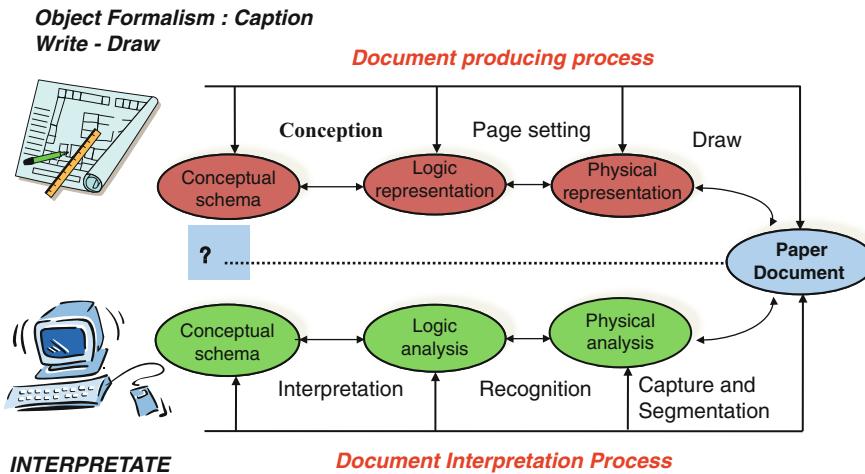
### Overview of Graphical Document Analysis Problems

The most generally adopted approach to graphical document analysis is related to the document reverse production process. This essentially means that the analysis aims at extracting information from a 2D representation. The principle of document production and the corresponding reverse interpretation is represented in Fig. 17.3.

As a consequence, a classical analysis process includes several stages, which try to reconstruct higher-level information from a 2D representation, on the basis of a progressive analysis, going from low-level information i.e., graphical primitives) to



**Fig. 17.2** Illustrations from [4] illustrating the results of graphical analysis tools linking textual and graphical data to aircraft maintenance semantic information



**Fig. 17.3** From the production of the document to its interpretation

a conceptual (semantic) level. Figure 17.3 gives an overview of this analysis process. The upper part of the figure (in red) describes the production process where an author within the context of her own conceptual schema will express her ideas using a logic (or otherwise formalized) representation language. This expression will then be transcribed and made physically available (as pixels, ink strokes, etc.).

Interpretation of the initial intentions of the author can only be achieved through the appropriate reverse analysis (in green, at the bottom of Fig. 17.3) provided that each process stage shares enough of the initial context to analyze the data. Consequently, the implementation of an analysis system requires the sequencing of many processes allowing to reach this goal and covering all the aspects required for reaching a semantic level of information, usually starting from a 2D digital image. The way these processes interact and the choice of which process to execute next, as well as the operational parameters they are fed, heavily depend on the interpretation context. Many of these processes include low-level image-processing techniques, primitive extractions, structural and statistical pattern recognition techniques, and semantic analysis. Each of these processes has already been described in great detail in the various chapters of this handbook.

The semantic analysis is principally driven by the knowledge related to the representation of symbolic objects on documents, which, in its turn, drives the different processes, tunes their parameters, stores the progressively extracted knowledge, and performs specific processing when semantic ambiguities are detected.

The main differences between the existing contributions related to analysis systems come from the way processes are organized and knowledge (and context) is (or is not) managed, from how semantic constraints are handled, and from how users are integrated in the analysis loop to achieve the required interpretation.

The state-of-the art literature contains a large variety of alternatives addressing these issues. Part of them is detailed in section “[Examples of Analysis Approaches and Interpretation Knowledge Representations](#).” The next section tries to establish a classification of the available approaches.

A special mention should be made for hand-sketched graphics recognition, some techniques of which are developed in ►[Chap. 28](#) (Sketching Interfaces) of this handbook. They generally apply to dynamic, on-line recognition contexts. A part from some very recent references such as [6, 23], this field of research only rarely incorporates complex interpretation goals. Most of the references of this research area deal with recognition issues and rarely consider interpretation strategies as those developed here.

## Classification of Graphical Document Analysis Systems

In order to get a structured view of the existing approaches to graphical document analysis, four different axes of observation should be considered, keeping in mind that there are many possible strategies to address the general interpretation problem, and categorization is always a fairly arbitrary task:

1. The application domain: what kind of documents is concerned?
2. What specific visual representation context characterizes the documents and what features are appropriate to the problem? In some cases the set of features is fixed; in other cases, the authors propose training capabilities to the system, in order to give them the ability to learn from examples and adapt to different feature sets.
3. How is knowledge modeled? Knowledge modeling is often tightly related to how the visual representation context and features are chosen (it may, for instance, depend on the structural/statistical description of the objects). Some authors try to differentiate the knowledge representation formalism from the one used for the object recognition process as is the case for those using ontologies [33], semantic networks [2], or constraints networks [32]. Knowledge can also be distinguished in the way authors store it, according to the reasoning process used in the interpretation module.
4. How is the interpretation strategy implemented or guided? The interpretation strategy can be completely hardwired in the case of bottom-up approaches, but it can also be guided by a blackboard in the context of centralized reasoning processes [43]; it can be distributed in multi-agent systems [3, 30]. Furthermore, the reasoning process handling the interpretation strategy can either be implemented as predefined interpretation scenarios (which often rely on bottom-up strategies) or, on the other hand, rely on opportunistic approaches based on progressive assessment of the knowledge which is produced by the different modules of the system.

Table 17.1 gives an overview of how the previous criteria apply to the main systems documented in the literature over the last couple of decades. It has to be noted that besides giving an overview of how interpretation strategies have been implemented,

**Table 17.1** Interpretation systems and application domains

Application domain	Features of handled documents	References	Features of the systems knowledge modeling	Interpretation strategy
Engineering drawings and mechanic charts	Black and white, containing lines, symbols, and texts Existing captions	[17]	Blackboard, modeling of knowledge based on grammars	Cycle of perception, opportunistic strategy; perceptive cycles, Neisser perceptive principles
		[43]	Blackboard-based modeling	Planned bottom-up approach
		[32]	Declarative language-based knowledge modeling, blackboard	Bottom-up approach
		[14]	Object-based approach, hierarchical class decompositions	Hypothesis emission/validation principle, dynamic planification
		[10]	Object-based approach defining the expected properties of objects	Bottom-up approach for interpretation; decomposition of the interpretation in 5 specialized subsystems specialized for 3D object reconstruction
Telephone networks, electric wiring diagrams, gas distribution network plans	Black and white, containing lines, symbols, and texts. Existing captions	[3]	Blackboard, heuristic-based knowledge introduced in the source code	Multi-operator-based approach based on cooperation between low-level operators, emission/validation strategies
		[11]	Adaptive strategies based on results assessment, perceptive cycles, semantic network for knowledge modeling	Centralized control, opportunistic strategy
Electronic diagrams	Black and white, containing lines, symbols, and texts Existing captions	[44]	Blackboard, heuristic knowledge introduced in the source code (no externalization)	Bottom-up and planned strategy

(continued)

**Table 17.1** (continued)

Application domain	Features of handled documents	References	Features of the systems knowledge modeling	Interpretation strategy
Geographical maps	Colored documents, containing textures, lines, symbols, and texts	[40]	Structure-based approach for knowledge modeling	Rule-based strategy and process assessment strategy
Existing captions		[38]	Distributed knowledge on agents and blackboard for centralized information	Distributed approach, agent-based approaches
		[37]	Blackboard approach, heuristic knowledge introduced in the source code	User interaction based interpretation strategy
Cadastral maps	Black and white, containing lines, symbols, and texts.	[2, 41]	Blackboard, semantic networks	Bottom-up approach, centralized strategy
Existing captions-high precision of document		[16, 29]	Blackboard, object model approach	Cyclic interpretation strategy, according to the semantic consistency of the interpreted data
		[33]	Ontology-based approach, centralized modeling system	Dynamic interpretation scenario, based on adaptive strategy Graph-based strategy
Architectural floor plans	Black and white, containing lines, symbols, and texts	[15]	Centralized knowledge, structure-based descriptions, knowledge based on constraints networks	Progressive solution construction, on the basis of the checking of primitives properties in a constraints network
Existing captions-high precision of document		[6]	Heuristic knowledge, blackboard	Bottom-up and planned strategy
Hand-sketched graphics	Black and white, manuscript containing text, lines, and symbols			

it is quite impossible to proceed to a more thorough objective analysis of these approaches and establish quality measures or possible rankings. This is due to the fact that, unlike what has occurred for more classical pattern recognition problems (symbol recognition, signature relevance analysis, etc.) or low-level processing evaluation (binarization, segmentation, etc.), the document analysis community has never organized interpretation evaluation contests. On the one hand, this lack of objective evaluation is due to the difficulty of implementing comparison of semantics on the basis on numerical scoring, and on the other hand, it is due to the fact that many research teams abandoned this exhaustive interpretation objective in favor of spotting information problems.

However, it is possible to provide some qualitative aspects guiding the reader in the implementation of an interpretation system. These qualitative aspects concern the facility of implementation as well as the ability of the system to incrementally integrate new knowledge and to automatically analyze the semantic consistency to the interpreted data in relation with the expectations of the user.

When dealing with an interpretation problem, the first point that must be considered in the implementation of a complete system is the necessity to simply formalize and externalize domain knowledge, in order to define the expectations of the user in terms of interpreted objects. This formalization must be based on user-friendly interfaces allowing the user to define which objects she is expecting and how they are graphically represented. From this point of view, the most relevant approaches are the one based on ontologies that permit to express this kind of knowledge in straightforward ways [33].

Considering the different manners to store the information, based on a centralized (mainly based on blackboard principle) vs. more distributed approaches (often multi-agent and multi-operator based), centralized knowledge-based systems seem to be much easier to implement, compared to distributed approaches, for which it is quite difficult to maintain overall consistency.

Concerning the interpretation strategy, while bottom-up and planified approaches were the most widely developed at the beginning of these research studies, cyclic approaches [17, 29] offer some very interesting advantages related to the possibility of using opportunistic interpretation strategies, often based on automated semantic consistency analysis.

Also, the interpretation process often relies on the use of pattern recognition tools, which can be based either on a statistical description of objects or on a structural description. From this point, even if the statistics-based approaches appear to be much more interesting from the algorithmic point of view, their poorness of description gives a large advantage to structural-based approaches [32].

It is also worthwhile to mention recent approaches, which consider user interactions allowing the system to integrate corrections provided by the user in the interpretation process and sometimes providing the possibility to infer on the knowledge of the system [20, 33].

Some of these systems try to implement generic and flexible interpretation strategies, most of the time on the basis of explicit knowledge modeling. However, they generally remain quite domain specific, due to the high number of heuristics

introduced in the processing chain. In this context, the commonly accepted notion of interpretation of graphical documents is to consider it as the result of an automated analysis process converting a poor format document (paper, pdf) into a format close to human interpretation. This, of course, is only partially satisfactory, since it defines computer interpretation in terms of human interpretation, without fully assessing what the latter actually entails. The rest of this chapter will develop in detail how these various approaches and applications have been constructed and how they consider “interpretation” of graphical data.

## **Relations Between Machine Perception and Image Analysis Systems**

The interpretation problem is a widely spread question, especially in computer vision communities. In many cases, document interpretation strategies can be partially inspired from computer vision communities, in which many image interpretation systems were also developed. Indeed, in the last 50 years, a lot of image interpretation applications have been developed in many fields (medicine, geography, robotics, industrial vision, etc.). However, image-processing specialists design applications by trial errors cycles and there is no identifiable tendency to reuse already-developed solutions. The lack of application modeling and context formalization may be a reason for this behavior:

- Accounts of full analysis systems are rare. Usually, publications focus on specific parts of the analysis pipeline, highlighting the scientific and theoretical foundations of their contributions. Very often, these reports conclude by providing experimental validation on specific data, claiming an improvement over competing approaches on the same, or similar, data. This results in a very result-focused definition of interpretation problems and obfuscates, in some way, both the actual interpretation context, on the one hand, and a formal description on how the analysis process advocates between possible choices.
- The reusability of these applications is therefore very poor because the limits of the solution applicability are not explicit. Moreover, they often suffer from a lack of modularity and the parameters are also often tuned manually without giving explanations on the way they are set. Besides, these parameters and their impact on the final interpretations hold a tight relationship with the interpretation context, as already stated before. If the context cannot be formalized on the one hand and if the parameter domain cannot be mapped to the context, reusability and generality can only occur through trial and error tuning.

There exist some approaches that try to address these issues, however. Knowledge-based systems such as OCAPI, MVP, or BORG were developed to automatically construct image-processing applications and to make explicit the knowledge used to solve such applications.

However, most a priori knowledge of the application context (sensor effects, noise type, lighting conditions, etc.) and the interpretation goal to achieve were still more or less implicitly encoded in the knowledge base. This implicit knowledge

restricts the range of application domains for these systems, and it is one of the reasons of their failure.

More recent approaches bring more explicit modeling but they are all limited to the description of business objects for detection, segmentation, image retrieval, image annotation, or recognition purposes. But they do not completely tackle the problem of the application context description and the effect of this context on the images (environment, lighting, sensor, image format). Moreover, they do not define the means to describe the image content when objects are *a priori* unknown or unusable (e.g., in robotic, image retrieval, or restoration applications). They also assume that the objectives are well known (to detect, to extract, or to recognize an object with a restrictive set of constraints) and therefore they do not address their specification.

---

## Examples of Analysis Approaches and Interpretation Knowledge Representations

It should be clear to the reader, by now, that from a technical point of view, there is no formalized and standard approach nor definition of graphical document analysis and interpretation. There are merely interesting and successful approaches that have proven efficient in specific application contexts. Taking a closer look to those, there are, however, some lessons to be learned from how they integrate various levels of knowledge and what strategies are deployed to make them as flexible as possible to adapt to other contexts. This section will try and provide an overview of these strategies.

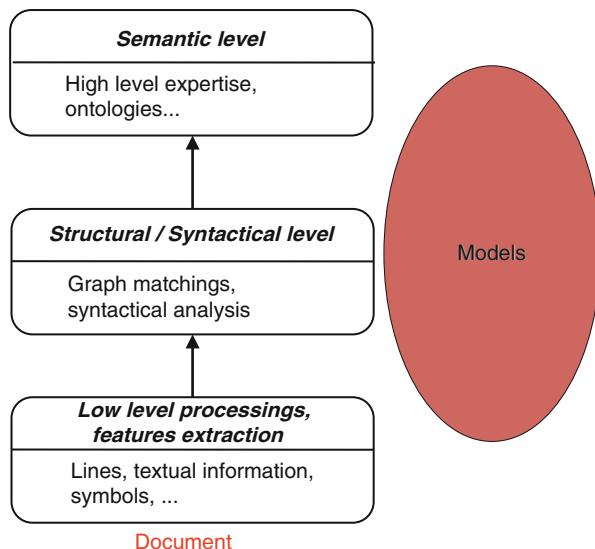
### Classical Strategies: Bottom-Up Approach

Image or document analysis is a difficult task since it requires a large amount of different data-processing techniques, from low-level treatments (e.g., noise filtering, data restoration) to high-level interpretation (e.g., object identification, decision making) through intermediary operators (e.g., segmentation). In order to solve this problem, most of the different strategies available in the literature are very much based upon the hierarchical decomposition of the problem shown in Fig. 17.4.

From the acquired data, treatments are most of the time run sequentially within a bottom-up strategy. Each operator of this decomposition provides a result, constituting the entry of the next operator. Following this approach, the most sensitive points are the choice of the optimal operators, the definition of the adequate sequential ordering of these treatments, the management of the quality (or uncertainty) of their results, and the communication between the different levels. Most of the time, this kind of conventional approach relies on three main levels (Fig. 17.4), each of which manages a particular level of information:

- The first level manages the extraction of low-level primitives: it often includes prepossessing techniques and extraction of primitives (lines, circles, textures,

**Fig. 17.4** Illustration from [27] depicting the main steps classical bottom-up interpretation approach



textual information, etc.). The techniques and tools related to this have been described in ►Chap. 15 (Graphics Recognition Techniques).

- The second level generally manages statistical, structural, or syntactic information and tries to combine low-level primitives into syntactically, structurally, or statistically described objects, on the basis of classification techniques, graph-matching approaches, or syntactical methods. Most of the approaches related to this level have been described in ►Chap. 16 (An Overview of Symbol Recognition).
- The third level generally tries to integrate semantic constraints, in order to solve ambiguities. This level is usually the less formalized and forms the core focus of this chapter.

Most often, graphical document analysis follows a bottom-up strategy. Algorithms are performed in a fixed sequence, usually starting with “low-level” analysis of the gray level or black and white image (sometimes combined with noise filtering and binarization cf. Part B (Page Analysis) in this handbook), in which primitives are extracted by specialized operators. Generally, these primitives correspond to segments, associated or not to polygonization algorithms, to symbols and characters, textures, circles or circular arcs, dashed lines, arrows, etc.

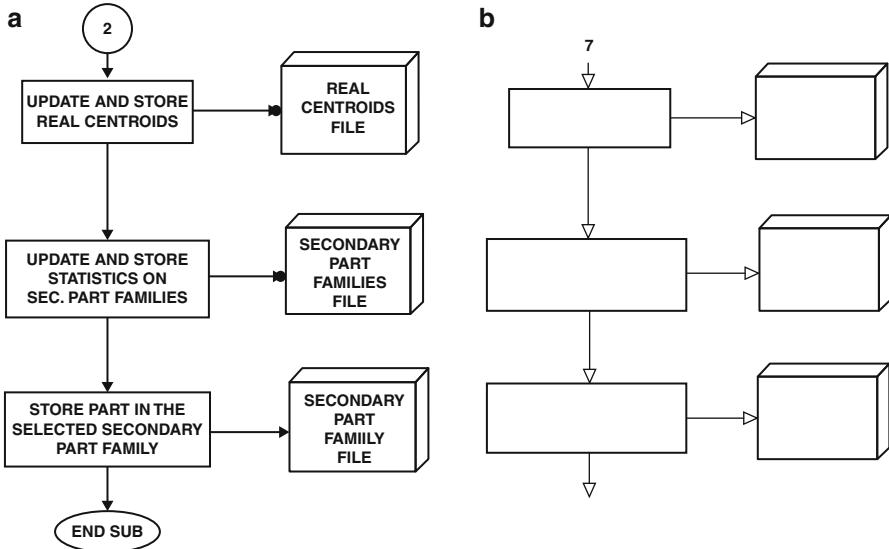
In the next phase, associations between all or a part of these primitives are detected, and higher-level graphical entities are constructed, guided by some a priori knowledge. This knowledge is either directly written into in the source code, or it can be declarative knowledge based on explicit rules for graphical entities. An analysis of graphical entities and their relationships allows one to propose an interpretation, in the case of strictly bottom-up approaches such as [5, 12, 19, 39, 41]. The main difficulty in this kind of process comes from obtaining significant and

robust graphical entities from the low-level operators and reliable association rules between each primitive in order to achieve a correct interpretation. These issues have already been partially discussed in ►Chap. 15 (Graphics Recognition Techniques). In fact, contradictory as it may seem, these systems all extract low-level primitives the same way, using the best state-of-the-art approaches as off-the-shelf tools, without necessarily taking into account the specificity of the visual representation of each graphical object within the context they are confronted with. As a consequence, due to the variability in representation and the manual fine-tuning of many of the intervening parameters as well as the handmade and fixed combination of supporting extraction and detection algorithms, many situations in technical documents are difficult to solve by these approaches. They usually concern the connection and overlapping between different visual entities (e.g., text, lines, and texture), text identification in handwritten annotations, isolated character recognition under multi-orientation constraints (for instance, in city maps or utility maps), low image quality, and variability in the representation of graphical entities. For all such strictly bottom-up systems, the main problem is related to the poor adaptation of the parameters of the extractors and to the inadequacy of operators to the local features of certain objects. As a side note, and with respect to text identification, it has to be stressed that OCR in graphical documents is a different challenge than character recognition addressed in ►Part C (Text Recognition) and more particularly, but not exclusively, ►Chap. 10 (Machine-Printed Character Recognition) in this handbook. Textual references are very often very short sequences for which no “text-only” context is available, as in full text environments (where dictionaries or other linguistic heuristics can guide in solving nondeterminism). Very often the interpretation context of textual annotations is related to the graphical context on the one hand and syntactical reference conventions or encoding on the other hand.

The most emblematic bottom-up approaches in the graphics literature are [19] and an updated version, applied to architectural drawings, adapted to the evolution of low-level treatment and higher-level recognition processes [15]. It is interesting to note that [19] considers “shapes” as the ultimate level of interpretation, regardless of what these shapes may represent. This means there is a complete lack of semantics in this approach. The goal of the approach is to have a geometric description (vector image) that would be as faithful as possible to the initial raster image but that would go beyond strokes and connected lines and incorporate coherent descriptions of shapes (circles, hexagons, parallelograms, etc.). Dosch [15] extends this low-level consideration to not only integrate symbol recognition but also add an interpretation step that is targeted toward their application context: architectural drawings.

Since it lacks a more semantic verification step, the former has interpretation artifacts like those shown in Fig. 17.5. While the method (almost) correctly separates geometric shapes from text, it identifies all 2D polygon shapes independently, failing to establish that they stem from a 3D projection and thus missing the vertex co-occurrence constraints on some boxes.

In [15] the bottom-up approach is extended to incorporate more elaborate shape recognition on the one hand, but also to relate them (and their visual context) to



**Fig. 17.5** Taken from [19] and showing the interpretation result. Note a minor segmentation error identifying the upper “2” as graphics and the vertices not overlapping for some of the 3D boxes

architectural representation knowledge, as to model 3D buildings from 2D scanned images, as shown in Fig. 17.6.

Recent approaches have tried to revisit this paradigm in the light of symbol spotting, without fundamentally changing the three levels described before. The main shift is operated at the low-level extraction where, instead of trying to extract features justified by human interpretation semantics (lines, text, textures), either non-discriminate small areas are extracted [36], at the standard image scale, scale-invariant interest points are extracted [25], or patches corresponding to regions of interest are used. These are generally based on pure signal processing techniques identifying the maximum of entropy of information-like zones [21]. They have been described in ▶Chap. 16 (An Overview of Symbol Recognition), and although they have proven to be quite efficient in lab environments [9], they have never really been integrated in graphical document analysis contexts beyond symbol recognition. One of the current main obstacles to mainstream development of these approaches in broader analysis processes comes from the fact that there currently is no trivial approach to integrate signal-based patches with the higher levels handling syntactic and semantic constraints.

As a summary, one could say that the major drawbacks of the bottom-up approaches are due to the fact that the processes running at each of the cited levels do not sufficiently integrate contextual information, if any. At the lower level, image-processing techniques and features extraction are run globally on the whole image, without integrating local contextual knowledge. This highly conditions the quality of the extracted information. These results are transferred from one level to another

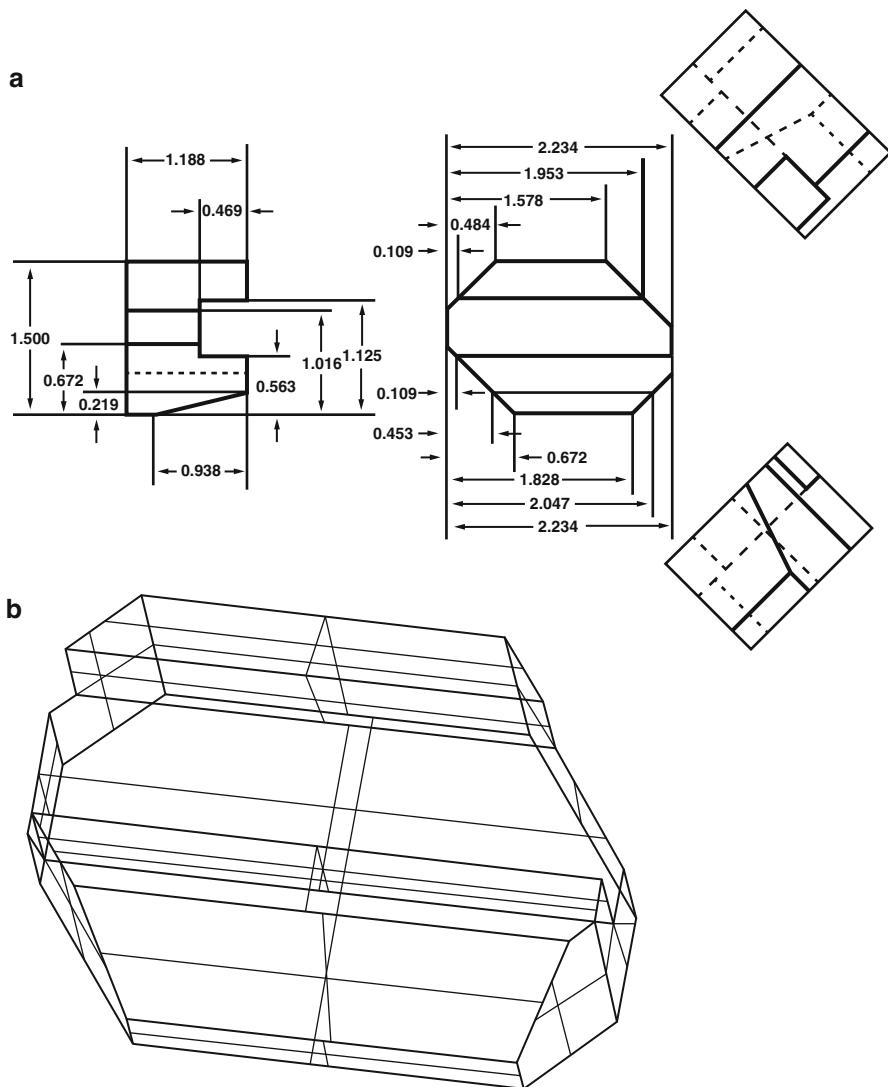


**Fig. 17.6** Taken from [15] representing the results at all major bottom-up stages of the interpretation process: line segmentation, symbol recognition, context identification, 3D reconstruction

without many possibilities of coherence or quality verification. Another key problem of this kind of approach is related to the fact that many sources of knowledge are implicitly embedded in the interpretation process, and this knowledge usually is tightly linked to the data and to the targeted application. This drawback makes it difficult for this kind of approaches to be reused in other contexts. Furthermore, the lack of definition of a memorization strategy in order to apply the most adapted analysis sequence for a specific context, by using contextual information but also by using the history of the device (as would human analysis do), represents a limitation to generalization.

As a consequence the document analysis research community (as well as the broader image-processing community) has quite well identified this problem as related to the adaptability of the interpretation device. To try and address it, knowledge-based alternatives have been developed in the hope of achieving more versatile analysis processes.

An illustration of this transition is the work by Devaux et al. [13] which consists in transforming 2D ANSI representation of 3D objects (containing dimensioning lines, orthographic projections, etc.) into full 3D representations. Although their work can still be considered as very similar to the bottom-up approaches, it clearly



**Fig. 17.7** From [13] obtaining a full 3D representation from a set of 2D ANSI orthographic projections

distinguishes itself from them by representing analysis knowledge as rules. An illustration from their work is shown in Fig. 17.7. One should consider [13] only from the perspective of graphical document analysis and the ways in which it extracts higher-level information from image pixel data. It does not intend to be a state-of-the-art reference to the problem of reconstructing 3D shapes from 2D projections. This problem has been addressed elsewhere and goes far beyond image analysis; it does not fall within the scope of this handbook.

## Knowledge-Based Approaches

In order to solve the difficulties mentioned in the previous section, mainly that low-level segmentation and extraction methods can be used off-the-shelf, but that domain knowledge and analysis “intelligence” is embedded in the underlying algorithms, many references in the literature consider *knowledge-based* approaches. This kind of approaches generally tries to solve the adaptability and genericity problems by formally representing some of the contextual knowledge needed for the interpretation. It thus becomes “externalized” from the analysis process, where it was implicitly embedded, before. This externalization of knowledge together with dynamic links between the process and the knowledge database opens the possibility to implement flexible and adaptable interpretation devices implementing a generic analysis that is capable of adapting itself to contextual information. This section provides an overview of some of the best-known knowledge models used in graphical document analysis.

### About Knowledge Modeling

Some authors propose using models of different knowledge categories that would contribute to the analysis process [1] and that these models be formalized as much as possible as to obtain a truly adaptable and context-independent interpretation system. A classification of the required knowledge in four categories can be found in [30]:

- The most obvious category concerns *descriptive knowledge*. It covers the knowledge over the physical or conceptual domain (semantics) represented in the document on the one hand, as well as the graphical conventions (semiotics) and the rules that govern them (syntax) to represent concepts. It may also include semantics of the document’s conventions like captions, legends, and references. Most often, however, it represents the rules for representing objects within the document and generally relies on structural/syntactic representations, such as graph, trees, grammars, semantic networks, or ontologies, some of which have already been described in ►Chaps. 15 (Graphics Recognition Techniques) and ►16 (An Overview of Symbol Recognition). The representation of this knowledge allows describing the hierarchical organization of elementary primitives, as well as their topological and geometric relationships. This hierarchical description then allows to further define semantic consistency rules that can be used to check whether the information generated from a bottom-up strategy is consistent or not.

Furthermore, it can also be used to define the sequences of tasks and subtasks to be run in order to progressively extract the information from the image and organize it according to the model. Recent developments [33] have started introducing ontologies to formalize this knowledge, since it allows not only to model hierarchies of concepts but also the relations that connect them.

- Another category covers the Image-Processing *operator’s knowledge*. It corresponds to the knowledge that is used by an expert to construct the analysis

process in the context of image-processing or pattern recognition problems. It is related to the behavior of the various image-processing operators that are used to implement the analysis strategy: they allow to describe in which context these operators are most appropriate, how they must be tuned versus a specific context, etc. For instance, they can correspond to the choice of the best image-processing operator for segmenting a texture or to the best couple (features vectors/classification process) that has to be used for recognizing a specific symbol. This knowledge is generally implicit and is rarely formally modeled. It finds itself embedded into the way algorithms are combined together (cf. next knowledge category) to form the overall analysis process, what parameter intervals are used and how they are obtained, which error or decision thresholds are used, etc. However, many papers mention the necessity to integrate this aspect when trying to implement generic systems. One of the steps in the direction of capturing the operator's knowledge, although not sufficient by itself, is to provide clear descriptions of input and output parameters and execution semantics of algorithms [20]. A more formal experiment toward integrating expert operator knowledge can be found in [7] although, strictly speaking, it does not fall within Graphics analysis; it does, in a general sense, apply to the issues described here. It presents the BORG system, aiming to generate image-processing programs and for which the proof of concept was established for cytology in medical imaging. Besides the grammar-based control mechanisms developed in the next sections (ANON, ADIC, etc.) and implementing selection strategies for finding the correct rule set to apply, BORG also allows to integrate quality measures to the image analysis steps expressing conditions like "*if the standard binarization algorithm gives rise to too many small connected components, revise the set of used thresholds in a previous step, or switch to an alternate binarization algorithm.*" The approach is not an image analysis method in a strict sense but an image analysis generator (i.e., given constraints, knowledge, and a set of training images, it will generate an image analysis program satisfying the given conditions and operating on the provided class of images).

- **Strategic knowledge** is a complementary level of implicit knowledge that is used by the image/pattern recognition expert when implementing an interpretation strategy. It deals with the sequential ordering of a set of image-processing operators in order to reach the analysis goal: how to sequence a set of image-processing operators and pattern recognition processes, in a particular context, and in order to achieve a specific level of interpretation. This kind of knowledge is far more difficult to formalize and is of a much higher level than the previous one, since it concerns the way to organize the process and not exactly how to tune each of them. The formalism that can be used for modeling this kind of knowledge can be inspired from Petri networks or serious games.

All these knowledge categories are implicitly involved in the building of an analysis device. It should be obvious to the reader that the genericity of analysis systems necessarily requires some level of formalization and external representation falling into these categories. While there is currently no existing consensus or formal theory

on how to ideally achieve this, there are however tentatives in this direction and the mentioned levels of knowledge are more or less formalized in hybrid systems, which are presented below.

## Hybrid Approaches for Graphics Analysis

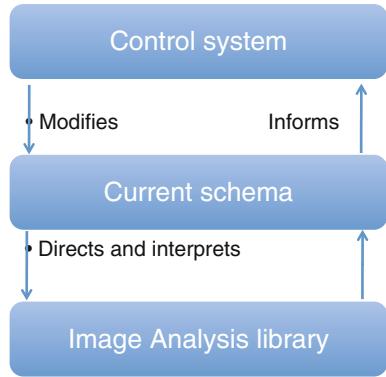
Hybrid approaches use a subset of the knowledge categories mentioned in the previous section for analyzing technical documents by leading the low-level processes as a function of the context. From a historic point of view, two approaches constitute interesting contributions to these approaches and can be taken as representative tokens of a more comprehensive state of the art.

### ANON and Grammar-Based Derivatives

One of them was proposed by Joseph [17] and concerns mechanical engineering drawing analysis. It was called ANON and used the “cycle of perception” proposed by Neisser, the basis of the approach being a continuous loop in which a constantly changing world model direct perceptual exploration determines how its finding are to be interpreted and is modified as a result. In ANON, this role is taken by an instance of one of a number of schema classes. The system is structured in three layers in order to separate spatial and symbolic processing. The first is composed of a large image analysis library associated to both search-tracking functions and management processes. The information extraction is adapted to the context by the second level, the “schema” (prototypical drawing construct), which receives the entities from the lower layer and interprets the result as a function of the current schema. A cycle of hypothesis verification is thus proposed by the schema to the control system (highest layer). On each cycle, the controlling instance invokes appropriate members of ANON’s library of image analysis routines and informs a higher-level control of the results of its actions. This control system analyzes the proposition as a function of the current state of the proposed schema and may modify it if needed. Applied in the context of graphical recognition, the system maintains classes corresponding to solids, dashed and chained lines, solid and dashed curves, cross hatching, physical outlines, junctions, letters, words, witness and leader lines, and certain restricted forms of dimensioning. Each schema instance represents a particular example of some prototypical drawing construct.

ANON’s control module comprises a set of strategy rules written in the form of a grammar. These rules define methods by which high-level drawing entities may be obtained by hierarchical combination of low-level constructs. On each cycle, the control system determines an appropriate modification to the current schema. Modifications may correspond to an updating of an internal variable, the adding of new subparts, or the replacing of the instance with a new one representing a different type of construct. Strategy rules, like string grammars, describe acceptable sequences of events ANON’s control model is represented in Fig. 17.8.

**Fig. 17.8** Control structure for ANON, as represented in [17]

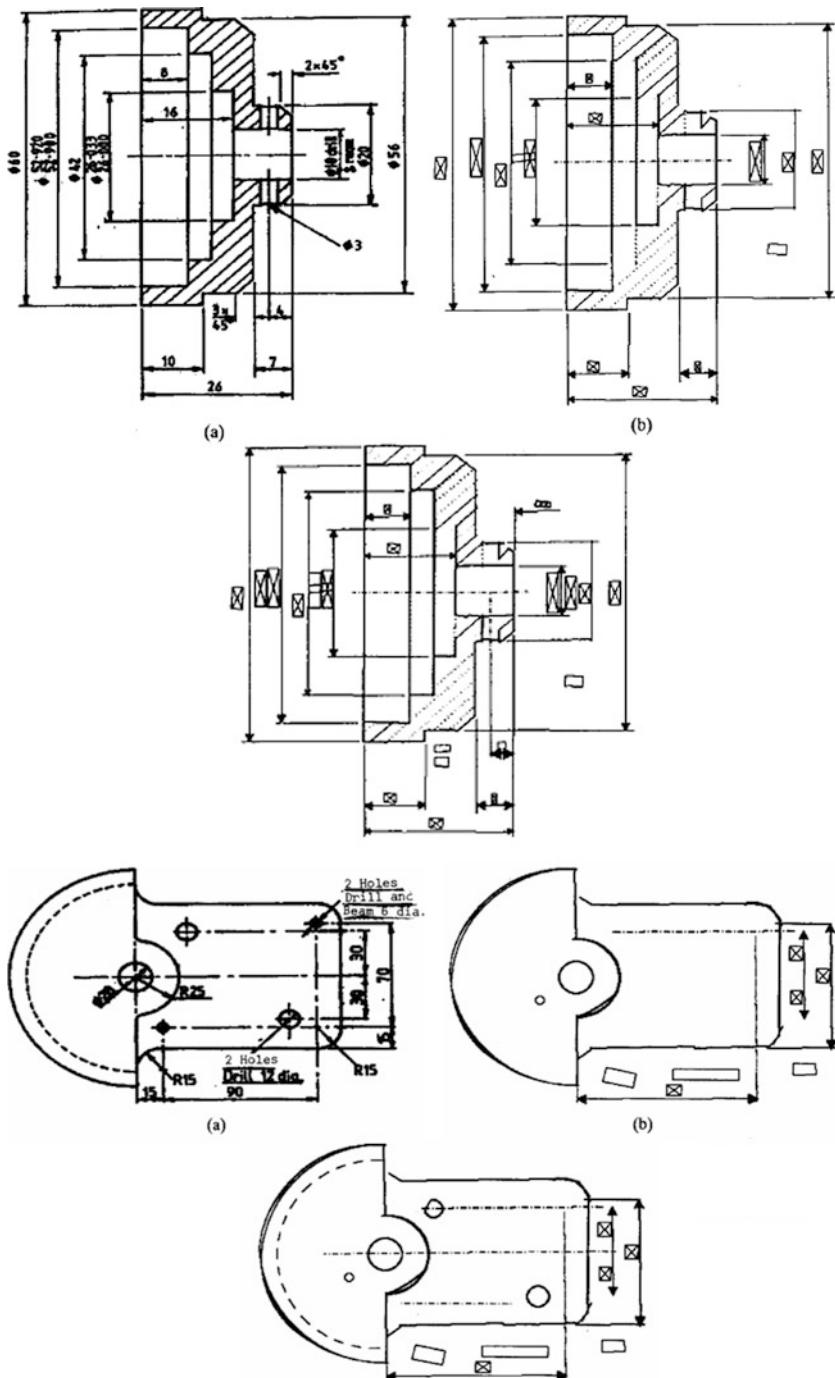


The results obtained in [17] are shown in Fig. 17.9 and clearly illustrate the limitations of the knowledge-based approaches in their beginnings. They have difficulties accounting for noise or for configurations that would be slightly deviating from the conventional configuration.

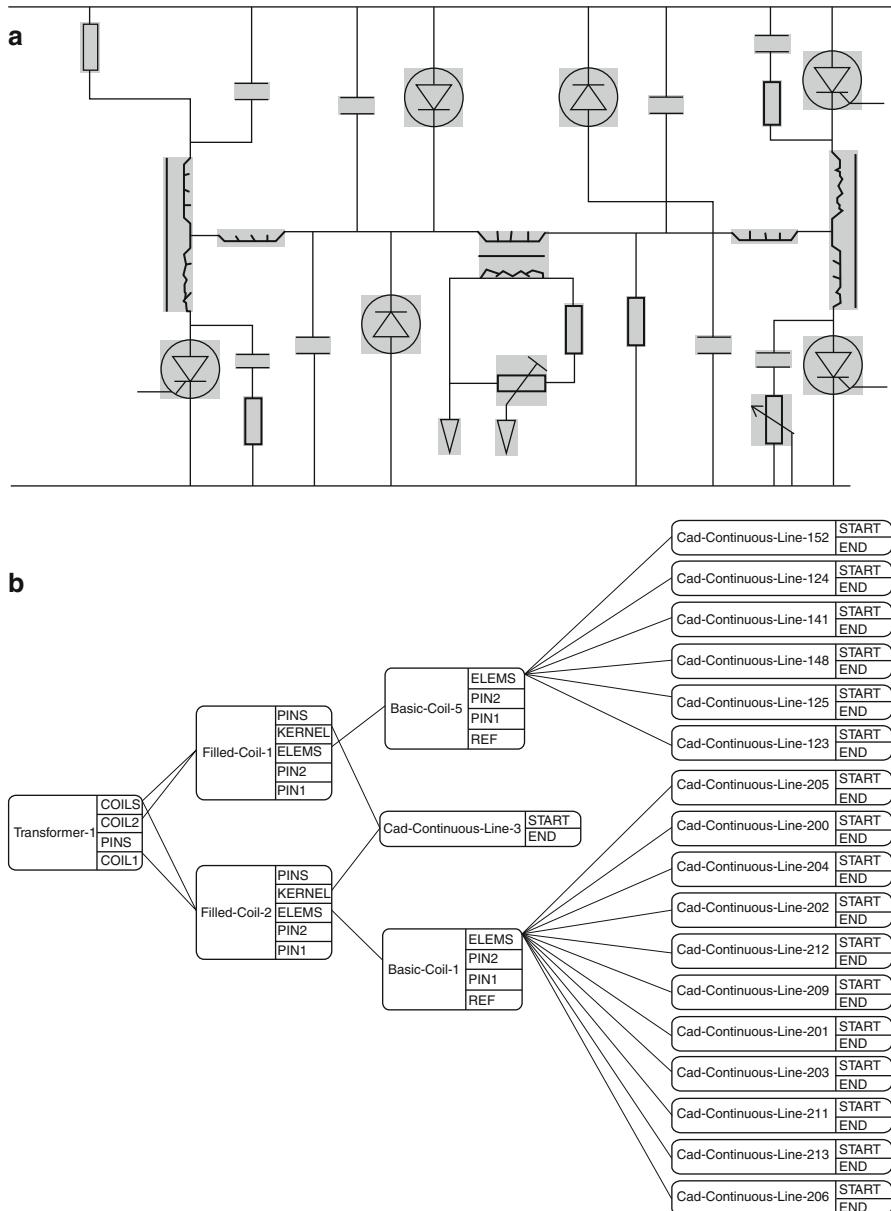
The knowledge-directed image analysis and the construction cycle according to the context are two interesting concepts that are applied to 15 different schema classes.

A similar approach is ADIK [31]. Its approach is very much related to syntactic symbol recognition and addresses the interpretation of technical diagrams by representing visual knowledge in the form of a grammar, expressing the various relationships and hierarchies between primitives and shapes as well as tolerances on allowable perturbations. The approach is conceptually similar to ANON, but is more flexible where its grammatical expression is concerned. The LR-grammar is extended with “placeholders,” “triggers,” and “constraints” which give it more possibilities to express local contextual conditions, making its behavior on triggering interpretations more flexible. Figure 17.10 gives some examples of detection results and the resulting exploration tree that results from the interpretation of the drawing.

In the same category, den Hartog [11] proposed a mixed approach based on a top-down control mechanism associated with bottom-up object recognition. The system decomposes the binary image into primitives (and not vectors) having a good morphological representation of the information and uses template matching to recognize each of them. Then, contextual reasoning is performed based on a loop that includes inconsistency detection and search action generation in a region of interest (ROI). The control system defines an ordered search action list to search for a specific object type in the ROI. The user specifies priorities to define the most important search actions and to assign priority to the relationship between objects. A consistency test is applied to each recognized object in order to verify the hypothesis defined at the system’s top level as a function of knowledge of the object to recognize. The knowledge framework of the methodology relies essentially on spatial relationships between primitives, without integrating and describing hierarchical relationships. In the case of particularly complex documents, this kind



**Fig. 17.9** These examples, as reported in [17], show input images and their corresponding output:  
 (a) the original input image, (b) the raw algorithm output, (c) the manually corrected results



**Fig. 17.10** Taken from [31] showing results on electrical wiring diagram analysis and the resulting interpretation tree that results from the interpretation process, connecting high-level concepts (transformers, for instance) to image features (continuous lines)

of system is penalized because of the drastically increasing number of relationships and the necessity to generate new search actions for the “designed objects.”

More recent work, revisiting the previous approaches, can be found in [22]. They identify several shortcomings, the main ones being related to the fact that only graphical constraints are explicitly modeled, while, in reality, technical drawings are also governed by implicit composition rules. This has led to over-investigate approaches that are essentially “linear” in their approach to combine graphical information and achieve interpretation from a set of pixels and for which non-shape domain information is not explicitly represented or at best, according to the authors, embedded in complex rule sets.

Lu et al. [22] therefore suggest using the explicit geometric shape definitions as entries for which implicit (in the sense that they are implicit for the human interpreter, meaning they need to be made explicit for an automated process) composition rules and representation conventions are used for guiding the analysis process and to check consistency or remove ambiguity. Their architecture is based on a knowledge interpreter, a knowledge parser, and an entity searcher, very much like ANON [17]. Using the assumption that automatic interpretation is composed of a series of condition-driven processes, these conditions are represented as knowledge descriptors addressing either representation issues (recognition) or interpretation issues (control).

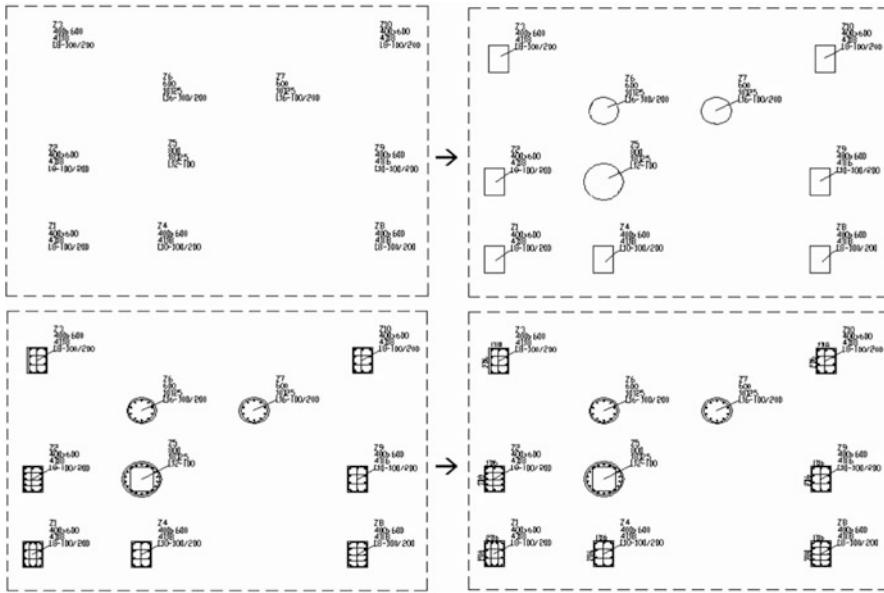
They identify four levels of interpretation targets: project, drawing, engineering entity, and graphical primitive for which knowledge is represented in EBNF (Extended Backus-Naur Form), all of which have external (purely graphical representations) and internal states (based on contextual and composition rules).

As shown in Fig. 17.11, the method is capable of detecting “similar” items, not only based on shape but also based on actual semantics.

The approach remains very sensitive to exhaustive visual modeling and errors introduced by noise on the one hand, as well as incoherency or missing information (local non-respect of conventions).

Among other relevant work, it is important to mention Couiasnon’s DMOS system (Description and Modification of Segmentation) for analyzing structured documents and which can also be applied to graphical documents. The aim of this system is to design a generic recognition system, being able of producing either general or specific systems. The DMOS system is made of a grammatical language (Enhanced Position Formalism—EPF) and an associated parser able to deal with noise. As for the previously presented systems, the main principle of DMOS is to separate domain knowledge from source program, in order to develop the adaptability of the system. Actually, DMOS relies on a compilation phase, which, in its turn, builds an adapted recognition system, on the basis of an EPF description of the expected document structure.

As the authors state in [8], “*This method has been successfully used to produce recognition systems on musical scores, mathematical formulae and even tennis courts in videos. This. Therefore, for a same kind of document like table structures, it is possible to define with EPF, more or less specific descriptions to produce more or less specific recognition systems. For example, we have been able to produce*

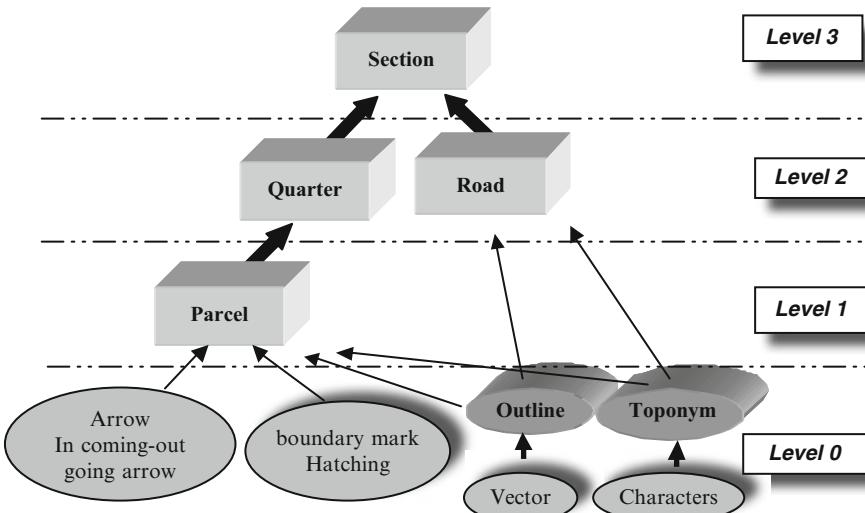


**Fig. 17.11** Taken from [22] looking for reference source columns in technical construction blueprints, starting with an initial example, and progressively expanding its knowledge tree to find all occurrences in the drawing

*a general recognition system of table structures. It can recognize the hierarchical organization of a table made with rulings, whatever the number/size of column/rows and the deep of the hierarchy contents in it, as soon as the document has a not too bad quality (no missing rulings for example). We will present the way the description is done using EPF to be general enough to recognize very different table organizations. With the same DMOS generic method, we have also been able to easily define a specific recognition system of the table structure of quite damaged military forms of the nineteenth century. This specific description was necessary to compensate some missing information concerning the table structure of those military forms, due to a very bad quality or hidden part of the table. This system has been successfully validated on 88,745 images, showing that this DMOS generic method can be used at an industrial level.”*

## Context Modeling and Ontologies

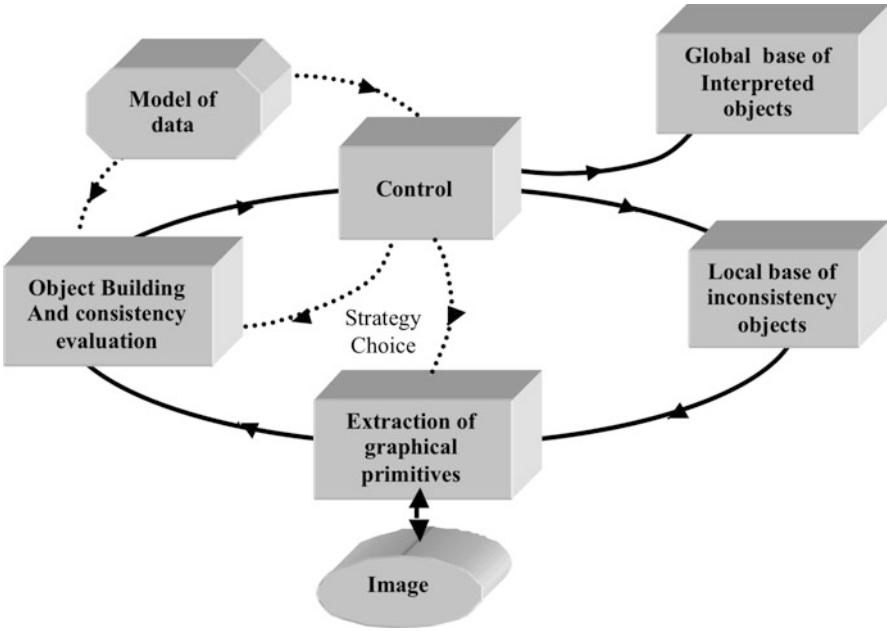
Another hybrid approach is the system described in [28] for map interpretation. In this system, features are grouped together to constitute primitive objects, then these objects are assembled together to compose a larger object in the hierarchy and the process continues until it reaches the most global object which is the map itself (Fig. 17.12).



**Fig. 17.12** Taken from [29] illustrating a model of knowledge representation in the context of French cadastral maps

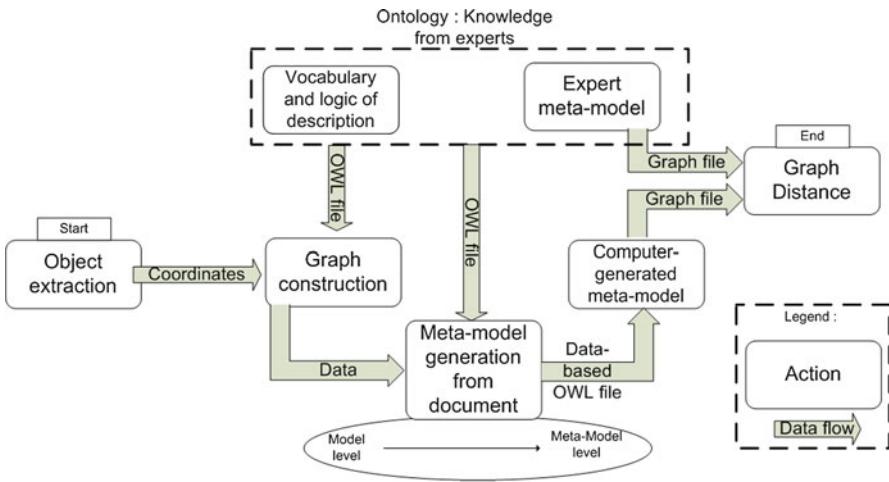
Although this formulation is not fundamentally different than the ones expressed previously (bottom-up and/or parser-based), the focus lies more on the fact that consistency checking is performed at every level in the hierarchy. Recognized objects are analyzed to verify if they are internally and externally consistent with each other. For example, a parcel is composed of segments to set up the outline, it has a number or an arrow, and it can involve a hatched area and symbols. Internal consistency means all the components composing the object are successfully detected; if not, a forward heuristic rule is used to correct this situation by re-extracting features in this region after modifying and relaxing the parameters of the low-level image-processing tools. On the contrary, external consistency takes into account the neighborhood of the treated object. If an object has all the components and responds to the semantic of the considered level, it is defined as an internal consistent; furthermore, if all the objects adjoining it are all internally consistent, this object will equally become more reliable through the construction of the superior hierarchical level (e.g., the parcel by the block). It is then called externally consistent. This approach is summarized in Fig. 17.13.

One of the more recent approaches was proposed in [33]. In this thesis an object extraction method from ancient color maps is proposed. It consists of the localization of the frame, text, quarters, and parcels inside a given cadastral map. First, a model of what visually characterizes a cadastral map is defined by combining knowledge from various domain experts: historians and architects on the one hand, and image analysis professionals on the other hand. Next, dedicated image-processing tools aim at locating the various kinds of objects laid



**Fig. 17.13** From [29] illustrating a cyclic strategy for the interpretation of French cadastral maps

out in the raster image. These especially designed detectors can retrieve different components such as characters, streets, frame, quarters, and parcels. Thereafter, this information feeds a higher level, which elaborates a graph structure where nodes refer to the presence of objects found during the detection step and edges represent the spatial relations between them. Terms, words, and appellations to qualify node and edge labels are so called concepts. All concepts have been previously modeled by the domain experts and are represented in an ontology containing the vocabulary and the description logics of each element required to model a cadastral map. Therefore, the produced graph can be seen as a particular instance of the generic map model. On the other hand, given the relatively “bottom-up” extraction method used to obtain the graph, the latter is not constrained by the ontology and variations which are nonconforming to the knowledge base may have been introduced into the graph structure (due to defects in the detection algorithms, noise in the images, unexpected shape variations, etc.). As a consequence, a higher level of representation is required to determine up to which level the extracted graph conforms to the expert knowledge. The structure of the graph is analyzed with the joint use of a cadastral map ontology to re-engineer a meta-model corresponding to the instance data. In a last phase, the meta-model corresponding to the instance data is compared with the meta-model defined by the experts. This comparison is carried out thanks to a graph-matching algorithm. An overview of the main actions carried out at this stage is shown in Fig. 17.14.



**Fig. 17.14** Data flow process for meta-model inference from a model, taken from [33]

## Discussion and Limitations

### General Discussion

Although there is no formal evidence of the following assumption, the graphical document analysis advances seem to have reached a plateau where end-to-end generic interpretation systems are concerned. The collection of concepts and methods coming from compartmentalized research communities that are required to be integrated with one another to produce full document analysis seem to resist to all efforts trying to remove human ingenuity. Therefore, current tendencies show the evolution toward strategies that are easier to implement, based on user interaction or based on partial interpretation strategies. Among these partial interpretation strategies, many of them rely on spotting-based concepts, which consist in offering navigation services into document database, without systematically interpreting the document content. These research axes appear to be very promising since they represent a good trade-off between efficiency, genericity, and automation. Some of these approaches integrate user interactions for the management of knowledge in order to dynamically adapt to new interpretation contexts, without the need of having them formalized. A short overview of these systems is given in the following part.

### Cookbook and Practical Tips for Graphics Interpretation

The essential remaining question that needs to be addressed in this chapter is “What do I do with my particular interpretation problem?” for the user who wants to implement some of the tools reviewed in this handbook.

It has already been stressed on multiple occasions throughout this chapter that there is no standard answer to the question. There are, however, some decent rules

of thumb that may guide the interested reader to a practical, operational, and efficient compromise. Table 17.2 gives an overview of configurations that are likely to occur, based on how much one knows about the graphical data at hand on the one side and the intended interpretation context on the other side.

---

## Conclusion

### **Recent Evolutions: Learning, Spotting, and Indexing for Navigation into Graphical Document Repositories**

Because of both the theoretical and practical considerations mentioned in the previous section related to the difficulty of developing generic analysis systems that would be able to manage any kind of document within a broad range of representations, recent approaches, principally developed during the last decade, try to approach analysis from another viewpoint. Rather than to aim for a fully automated analysis process and subsequent interpretation, with the difficulties of capturing all contextual knowledge, the research focus has shifted to leaving part of the analysis to a human interpreter and to offering efficient tools for handling large volumes of documents, mainly using “intelligent” indexing strategies.

One of the main turning points introducing this paradigm shift can be attributed to [42] spurring investigations toward the idea of “spotting” or [24] insisting on human interaction. Their overall observation is that rather than trying to fully represent contextual knowledge for solving interpretation problems, three main substitution strategies may prove just as effective:

1. Example-based or supervised learning and classification techniques can be used to replace interpretation context modeling (although there are pitfalls to be avoided if the sample population is inadequately chosen [24]).
2. Spotting and indexing can be used to (partially) replace full contextual interpretation by guiding a human interpreter rapidly to documents or document parts that have a high probability of fitting his or her interpretation.
3. Human interaction should be much more seen as a continuous part of the analysis process, rather than just participating in the knowledge modeling phase (e.g., by dynamically influencing classifier decision boundaries or indexing feature selection through relevance feedback).

As an illustration, one can consider the example of an analysis system using spotting techniques and indexing. The idea of spotting essentially consists in favoring recall over precision by proposing realistic navigation and retrieving services on large document corpora, fine-tuned on the requirements of a specific use case (usually given by the organizations managing the corpora), without systematically running costly full recognition processes (i.e., both high precision and high recall). One of its main advantages is that it avoids trying to interpret the whole document (which, for specific applications, is not really useful, anyway) and allows a possible second-stage process to focus on the interpretation on relevant spotted objects. Therefore, this kind of approach allows to simplify the analysis process, since it can use

**Table 17.2** Synoptic cookbook for solving practical document analysis problems in function of how well the interpretation context is known, relating to the intended public and users (horizontal) and how data variability can be controlled (vertical)

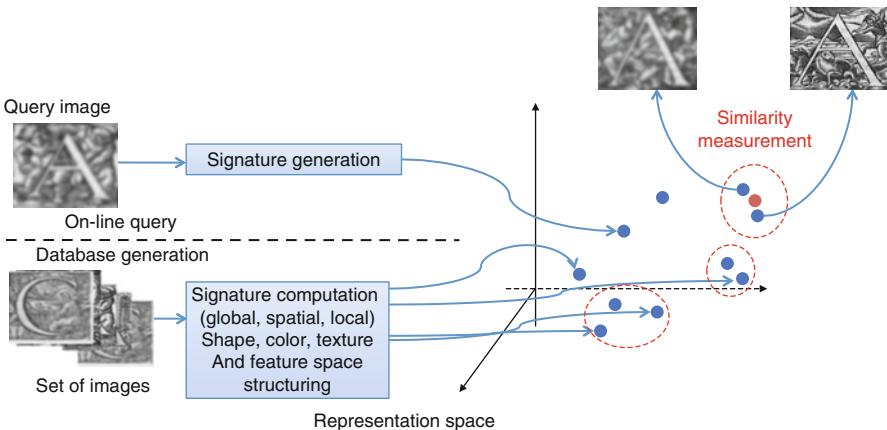
Control over interpretation context (horizontal) vs. control over data variability (vertical)	Low – interpretation context cannot be formally determined: general public and nonspecialist users	Medium – interpretation context can be formalized in some way: public or users are assumed to be knowledgeable to some level	High – interpretation context is highly constrained and can be formalized: trained public and specialized users	Invariants for given variability on data
<b>Low</b> Data may contain relevant patterns but they cannot be determined in advance and data of interest may be hidden in stream of potentially irrelevant documents	No reports exist of proven valid approaches in this very open context. Current consensus is that the most appropriate way of dealing with these classes of problems is to use weakly supervised classification and user relevance feedback to implement a symbol spotting approach and hope for the best [34]	This is one of the harder cases, where there is a fairly well, although insufficiently formalized, interpretation context, but without sufficient visual support in the data (i.e., there is a large <i>semantic gap</i> ). In this case, the most appropriate approach is likely to investigate further whether there are classes of interpretation contexts that can be better specified and reduce the problem to a more constrained one	If the interpretation context can be formalized in explicitly, it is possible to establish a set of constraints the data must fulfill (although they need not necessarily be expressed in visually verifiable ways). This can in its turn guide top-down analysis strategies, on the one hand. On the other hand, the presence of specialized users can also lead to a two-stage coarse-to-fine approach where supervised learning techniques can be trained to reduce the number of irrelevant documents first and then separate the data “manifold” into “sub-manifolds” of lower variability [28]	For this category of graphical data, the best approach is to use statistical learning over a large set of possible relevant descriptors. There is no guarantee that the combination of classifiers and descriptors will eventually capture all possible instances of relevant data. Statistical evaluations of the power of discrimination of descriptors should be used [35]

(continued)

**Table 17.2** (continued)

<b>Medium</b>	Data can be assumed to obey general visual rules, but variations in shape or noise can be important and are difficult to model. The application context is such that outliers containing completely irrelevant data are rare	This situation corresponds to particular cases for which similarity between data can be expressed through statistical descriptors as well as visual key point approaches in the context of information spotting. This kind of approach has been recently used in the context of historic document indexing [26] in the case of weakly structured graphical images. User interaction should be used to characterize	This context is very rare and there are no relevant references available	In this situation the focus of the analysis process will be on establishing how the formal interpretation knowledge projects onto the approximate and noisy (yet globally coherent) graphical data. This is probably the most active area in the current state of the art. Coulaudon [8] and Raveaux [33] are excellent examples	When data is only approximately formalized yet most of it is relevant, the current state-of-the-art consensus is to use non- or semi-supervised classification techniques to determine optimal descriptions of the graphical data, e.g., by using visual bag of words. Depending on how much is known about the interpretation context, the supervision of the classification techniques and/or their a posteriori validation can be more or less elaborate and efficient
<b>High</b>	Data falls into a very well understood and formally modeled scope, noise and deformation levels are within well-established bounds, and outliers are rare	Highly controlled graphical data most often comes from very well-controlled production contexts and obeying to a very well-established visual language. It seems contradictory to combine this with an undetermined interpretation context, but there are situations where this makes	Since the graphical data can be fully modeled and noise and deformation levels are known, the appropriate image description tools can be used to fully characterize the data. According to the available interpretation knowledge, the user can be presented with classes of visual	In this configuration the level of available formalization of both the interpretation and the data does not require a complex level of knowledge representation. A custom-tuned (probably bottom-up) and hardwired approach is very likely to be the most efficient and cost effective [15]	Given the high accuracy of knowledge about how the data behaves, robust and reliable visual descriptors can be easily extracted from the document images. Depending on what kind of interpretation knowledge is available, this visual (low-level) interpretation can then be more or less related to

<p>sense. For instance, electric wiring diagrams or architectural floor plans may be used for a very broad variety of interpretation contexts. The most reasonable approach here is to consider “retro-conversion,” without trying to capture the end user’s interpretation intent, but by trying to express the visual information into a higher-level formal description that can then be used as a support for a user-driven analysis process [18]</p>	<p>information or partial interpretations with respect to the level of available interpretation contexts’ formalization.</p> <p>Semi-supervised learning or user relevance feedback can be applied to converge toward a user-acceptable interpretation. Fully automatic interpretation remains out of reach [22]</p>	<p>higher-level interpretations, either fully automatically or with some user interaction, when possible</p>	<p>The existence of a very well known and formalized interpretation context, as well as specialized users, offers the possibility to fully express acceptance/rejection criteria of interpretations. Depending on what quality of data it is applied to, these acceptance/rejection criteria can more or less be related to visual information</p>
<p><b>Invariants</b> for given class of interpretation contexts</p>	<p>In this context it is impossible to make any assumption about what final interpretation is required and needs to be left to subsequent treatment or analysis; either this is done by the user itself, or enough is known about the visual description language so that a formal description of the document can be extracted for further automated analysis</p>		



**Fig. 17.15** Principle of signature computation and information spotting in the context of drop caps spotting

alternative detection and recognition processes which rely on less complex data representations and pattern recognition strategies. Furthermore, from the knowledge management point of view, this kind of system does not necessarily require exhaustive formalization since most of it can be dynamically caught through man-machine interactions or relevance feedback. The global schema of such a spotting system is presented in Fig. 17.15.

This kind of system is generally composed of two main parts, one off-line and another on-line.

The off-line part aims at analyzing the content of documents by extracting some features allowing to characterize each of them in a unique way. Until early 2000, the features that were used to describe documents generally relied on classical pattern recognition-based descriptors, i.e., based on statistical or structural approaches described in previous sections of this handbook. More recently, bag-of-words approaches were proposed, trying to characterize document contents without necessarily describing them on the basis of human knowledge.

The on-line part aims to propose interactive interfaces allowing the user to retrieve documents on the basis of queries, which can be expressed either on the basis of keywords or on the basis of images.

## Challenges in Graphics Interpretation

The main conclusion drawn from the previous sections is that there is no actual commonly agreed set of best practices or globally adopted methodology for complete analysis and interpretation systems. As a matter of fact, the research community has gradually abandoned the investigation of end-to-end applications in this domain, focusing more on subparts like recognition and indexing, segmentation,

or spotting. This is the main reason why many of the references in the previous sections are relatively old. While the results of these described approaches are quite interesting in their respective application contexts, it becomes less easy to really assess their value from a more general viewpoint: how do they adapt to other contexts, how do they perform with regard to recent developments in lower level treatments, etc.? The result is that most interpretation and analysis methods remain very context specific and that, therefore, analysis problems are handled on an ad hoc basis. Therefore, the main challenge for the graphical interpretation community is to establish a classification of its methods and low-level approaches described in the previous sections and to relate their appropriateness to higher-level interpretation contexts. As made clear through the overview given in this chapter, and based on the results presented in the previous chapters, there is a significant gap between the performance of individual graphical image treatment and recognition approaches and the performance of full analysis methods. In this chapter the focus has been set on knowledge modeling as a means to bridge this gap, and several partially successful approaches have been developed in this domain, over the years. However, effective conclusions still need to be drawn from these experiments and there is no established consensus on good practices or better choices in specific application contexts.

The main challenges for graphics analysis therefore are related to performance analysis on the one hand, and context characterization on the other hand as well as fully integrating the human user in the analysis and interpretation process, helping to focus the knowledge modeling or information characterization through relevance feedback, for instance.

It remains an open question whether this is actually a realistic goal. Since there is no recent published work in what interpretation exactly means or entails from a Machine Perception point of view, the problem of measuring the state of the art remains open. However, when broadening the scope beyond Machine Perception into formal semantics and model checking on the one hand and reaching out even further into linguistics and even metaphysics, it does not seem absurd to try and relate recent advances in those domains concerning semantics and interpretation to what has been described in this chapter. While automated interpretation of perception data is a very ill-posed problem in the current state of the art, trying to formulate it in a more abstract way will probably show a number of limitations related to tractability and decidability and therefore allow the graphical document analysis domain to better grasp the reasons behind the currently observed limits of its approaches and perhaps provide means to try and overcome them.

---

## Cross-References

- ▶ [An Overview of Symbol Recognition](#)
- ▶ [Graphics Recognition Techniques](#)
- ▶ [Imaging Techniques in Document Analysis Processes](#)
- ▶ [Sketching Interfaces](#)

## References

1. Adam S, Ogier J-M, Cariou C, Mullot R, Labiche J, Gardes J (2000) Symbol and character recognition: application to engineering drawings. *Int J Doc Anal Retr* 3(2): 89–101
2. Antoine D, Collin S, Tombre K (1992) Analysis of technical documents: the REDRAW system. In: Baird HS, Bunke H, Yamamoto K (eds) *Structured document image analysis*. Springer, Berlin/Heidelberg, pp 385–402
3. Arias JF, Lai CP, Surya S, Kasturi R, Chhabra A (1995) Interpretation of telephone system manhole drawings. *Pattern Recognit Lett* 16:355–369
4. Baum LS, Boose JH, Kelley RJ (1997) Graphics recognition for a large-scale airplane information system, GREC. Lecture notes in computer science, vol 1389. Springer, Berlin, pp 291–301
5. Boatto L et al (1992) An interpretation system for land register maps. *IEEE Comput Mag* 25(7):25–33
6. Broelemann K, Jiang X (2012) A region-based method for sketch map segmentation. In: *Graphics recognition. New trends and challenges*. Lecture notes in computer science, vol 7423. Springer
7. Clouard R, Elmoataz A, Porquet C, Revenu M (1999) Borg: a knowledge-based system for automatic generation of image processing programs. *IEEE Trans Pattern Anal Mach Intell* 21(2):128–144
8. Coüasnon B (2006) DMOS, a generic document recognition method: application to table structure analysis in a general and in a specific way. *Int J Doc Anal Retr* 8(2–3): 111–122
9. Coustaty M, Uttama S, Ogier J-M (2012) Extraction of light and specific features for historical image indexing and matching. In: 21st international conference on pattern recognition, Tsukuba, 11–15 Nov 2012, pp 1326–1329
10. Das AK, Langrana NA (1997) Recognition and integration of dimension sets in vectorized engineering drawings. *Comput Vis Image Underst* 68(1):90–108
11. Den Hartog JE, ten Kate TK, Gerbrands JJ (1996) Knowledge-based interpretation of utility maps. *Comput Vis Image Underst* 63(1):105–117
12. Deseilligny MP, Mariani R, Labiche J, Mullot R (1998) Topographic maps automatic interpretation: some proposed strategies. In: Tombre K, Chhabra AK (eds) *Graphics recognition algorithms and systems*. Lecture notes in computer science, vol 1389, pp 175–193
13. Devaux PM, Lysak DB, Kasturi R (1999) A complete system for the intelligent interpretation of engineering drawings. *Int J Doc Anal Retr* 2(2–3):120–131
14. Dori D, Wenyin L (1999) Automated CAD conversion with the machine drawing understanding system: concepts, algorithms, and performance. *IEEE Trans Syst Man Cybern Part A* 29(4):411–416
15. Dosch P, Tombre K, Ah-Soon C, Masini G (2000) A complete system for the analysis of architectural drawings. *Int J Doc Anal Retr* 3(2):102–116
16. Janssen RDT (1995) The application of model-based image processing to the interpretation of maps. PhD thesis. Technische Universiteit, Delft
17. Joseph SH, Pridmore P (1992) Knowledge-directed interpretation of line drawing images. *IEEE Trans PAMI* 14(9):928–940
18. Kasturi R, Tombre K (eds) (1996) *Graphics recognition – methods and applications*. Lecture notes in computer science, vol 1072. Springer, Berlin
19. Kasturi R et al (1990) A system for interpretation of line drawing. *IEEE Trans PAMI* 12(10):978–992
20. Lamiroy B, Lopresti D (2011) An open architecture for end-to-end document analysis benchmarking. In: 11th international conference on document analysis and recognition – ICDAR, Beijing. IEEE Computer Society, pp 42–47
21. Li L, Tan CL (2012, to appear) Real scene graphics symbol recognition. In: Post-conference proceedings of the GREC 2011, Seoul. Springer

22. Lu T, Tai C-L, Yang H, Cai S (2009) A novel knowledge-based system for interpreting complex engineering drawings: theory, representation, and implementation. *IEEE Trans Pattern Anal Mach Intell* 31(8):1444–1457
23. Mas J, Lladós J, Sánchez G, Jorge JAP (2010) A syntactic approach based on distortion-tolerant Adjacency Grammars and a spatial-directed parser to interpret sketched diagrams. *Pattern Recognit* 43(12):4148–4164
24. Nagy G (2004) Visual pattern recognition in the years ahead. In: Invited plenary presentation, Proceedings of the international conference on pattern recognition, Cambridge, vol II, pp 311–314,
25. Nguyen T-O, Tabbone S, Boucher A (2009) A symbol spotting approach based on the vector model and a visual vocabulary. In: Proceedings of the 10th international conference on document analysis and recognition, Barcelona, pp 708–712
26. Nguyen TTH, Coustaty M, Ogier J-M (2011) Bags of strokes based approach for classification and indexing of drop caps. In: Proceedings of the 11th international conference on document analysis and recognition, Beijing, pp 349–353
27. Ogier J-M, De l'image au sens, Habilitation à diriger de recherches thesis (2000). University of Rouen, 22 Dec 2000
28. Ogier J-M, Mullot R, Labiche J, Lecoutier Y (1998) Methodology and evaluation of a document interpretation device. In: Proceedings of the SCI'98 and ISAS'98, world multiconference on systemics, cybernetics and informatics and the 4th international conference on information systems analysis and synthesis, Orlando, vol 2, 12–16 July 1998, pp 457–463
29. Ogier J-M, Mullot R, Labiche J, Lecourtier Y (2000) Semantic coherency: the basis of an image interpretation device-application to the cadastral map interpretation. *IEEE Trans Syst Man Cybern B* 30(2):322–338
30. Ogier J-M, Mullot R, Labiche J, Lecourtier Y (2001) Technical map interpretation: a distributed approach. *Int J Pattern Anal Appl* 3:88–103
31. Pasternak B (1994) Processing imprecise and structural distorted line drawings by an adaptable drawing interpretation kernel. In: Proceedings of the IAPR workshop on document analysis systems, Kaiserslautern, pp 349–365
32. Pasternak B, Neumann B (1995) The role of taxonomy in drawing interpretation. In: Proceedings of the 3rd international conference on document analysis and recognition, Montreal, pp 799–802
33. Raveaux R (2010) Graph mining and graph classification: application to cadastral map analysis. PhD thesis, University of la Rochelle, La Rochelle
34. Rendek J, Lamiroy B, Tombre K (2006) A few steps towards on-the-fly symbol recognition with relevance feedback. In: 7th international workshop, document analysis and systems, Nelson. Lecture notes in computer science, vol 3872. Springer, Berlin, pp 604–615
35. Rosenberger C, Ogier J-M, Cariou C, Chehdi K (1998) A statistical solution to evaluate image processing techniques. In: 7th international conference IAPMU, Paris, vol 2, pp 1648–1645
36. Rusiñol M, Borràs A, Lladós J (2010) Relational indexing of vectorial primitives for symbol spotting in line-drawing images. *Pattern Recognit Lett* 31:188–201
37. Samet H, Soffer A (1994) A legend-driven geographic symbol recognition system. In: Proceedings of the 12th international conference on pattern recognition, Jerusalem, vol II, pp 350–355
38. Shimada S, Maruyama K, Matsumoto A, Hiraki K (1995) Agent-based parallel recognition method of contour lines. In: Proceedings of the ICDAR'95, Montreal, pp 154–157
39. Shimotsuji S, Hori O, Asano M, Suzuki K, Hoshino F, Ishii T (1992) A robust recognition system for a drawing superimposed on a map. *IEEE Comput* 25(7):56–59
40. Smeulders AWM, de Boer C (1997) Design and performance in object recognition. In: Second IAPR international workshop on graphics recognition (GREC), Nancy, pp 335–346
41. Suzuki S, Yamada T (1990) MARIS: map recognition input system. *Pattern Recognit* 23(8):919–933
42. Tombre K, Lamiroy B (2003) Graphics recognition – from re-engineering to retrieval. In: ICDAR, Edinburgh

43. Vaxivière P, Tombre K (1995) Knowledge organization and interpretation process in engineering drawing interpretation. In: Spitz AL, Dengel A (eds) Document analysis systems. DFKI, pp 307–317
44. Yu Y, Samal A, Seth SC (1995) A system for recognizing a large class of engineering drawings. In: Proceedings of the 3rd international conference on document analysis and recognition, Montreal, pp 791–794

## Further Reading

Although the knowledge-based approaches described in the previous section seem intellectually much more satisfactory and more enhanced than classical approaches since they try to dynamically link knowledge, image analysis techniques, pattern recognition tools, and document interpretation contexts, very few production-ready systems or significant technological breakthroughs have been reported since 2000. Kasturi and Tombre [18] contains a good state-of-the-art review of the main available techniques and tools for the analysis of technical and cartographic documents and subsequent publications have not really contributed to fundamentally change the state of the art.

Exception can be made for the recent evolutions involving the development of ontologies and web semantic approaches [33]. They offer some interesting alternatives that allow to better formalize knowledge and render them quite versatile for use in analysis systems. However, considering the high level of complexity of these problems, even if scientific communities agree on the importance of the development of generic interpretation systems, and the necessity to dynamically connect knowledge management and analysis scenario, one must admit that there is still no generic system allowing to solve broad classes of interpretation problems. One of the reasons may be that ontologies themselves (or any other formal knowledge representation) are only partially capturing the underlying complexity of the required knowledge by requiring that it be expressed within the boundaries of computational description logics, thus shifting the intrinsic difficulties just a level further without actually addressing them in full.

Anastasios Kesisidis and Dimosthenis Karatzas

## Contents

Introduction.....	592
History and Importance.....	593
Applications.....	595
Trademark Retrieval Systems.....	596
Problem Definition.....	596
Trademark Descriptors.....	598
Information Fusion, Indexing, and Retrieval Strategies.....	606
Systems Overview.....	609
Open Challenges in Trademark Retrieval.....	612
Logo Recognition in Document Images.....	614
Problem Definition.....	614
Detection.....	615
Recognition.....	616
Logo Spotting.....	618
Systems Overview.....	618
Open Challenges in Logo Recognition in Documents.....	618
Logo Detection and Removal in Images and Videos.....	622
Problem Definition.....	622
Detection and Retrieval of Logos in Images and Videos.....	625
TV Logo Detection and Removal.....	630
Open Challenges in Logo Detection and Removal in Images and Videos.....	635
Conclusion.....	638
Description of Reference Datasets.....	639

---

A. Kesisidis (✉)

Department of Surveying Engineering, Technological Educational Institution of Athens, Aigaleo, Athens, Greece

e-mail: [akesidis@teiath.gr](mailto:akesidis@teiath.gr)

D. Karatzas

Computer Vision Center, Universitat Autònoma de Barcelona, Bellaterra, Spain

e-mail: [dimos@cvc.uab.es](mailto:dimos@cvc.uab.es)

Trademark Retrieval Systems.....	639
Logo-Based Document Classification.....	640
Logo Detection and Removal in Images and Videos.....	640
Cross-References.....	641
Notes.....	641
References.....	642
Further Reading.....	646

---

### Abstract

The importance of logos and trademarks in nowadays society is indisputable, variably seen under a positive light as a valuable service for consumers or a negative one as a catalyst of ever-increasing consumerism. This chapter discusses the technical approaches for enabling machines to work with logos, looking into the latest methodologies for logo detection, localization, representation, recognition, retrieval, and spotting in a variety of media. This analysis is presented in the context of three different applications covering the complete depth and breadth of state of the art techniques. These are trademark retrieval systems, logo recognition in document images, and logo detection and removal in images and videos. This chapter, due to the very nature of logos and trademarks, brings together various facets of document image analysis spanning graphical and textual content, while it links document image analysis to other computer vision domains, especially when it comes to the analysis of real-scene videos and images.

---

### Keywords

Logo recognition • Logo removal • Logo spotting • Trademark registration • Trademark retrieval systems

---

## Introduction

Logos and trademarks offer a particularly interesting setting for document analysis applications. By their very nature, they can incorporate any combination of textual and graphical content; subsequently, their treatment typically spans different areas of document analysis. At the same time, the range of applications centered on logo and trademark detection and recognition stems from various, quite different in nature necessities. In this chapter the main species of such applications are defined, and related methodologies and best practices are reviewed.

The word “trademark” is a legal concept that covers any type of indicator that can distinguish the products or services (in which case it is called a “service mark”) of an entity from those of the competitors. Trademarks in the wider sense can in principle be defined over any modality such as sound, movement, a distinctive color, or individual words, as long as they are capable of distinguishing goods or services in a particular market segment.

A logo on the other hand is a graphic representation such as a symbol, badge, emblem, icon, or picture. Formally, when a logo comprises just an arranged typeface (e.g., the Microsoft logo), it is called a logotype, although habitually the term logo is used to denote any combination of graphic representation and arranged typeface. A logo can be used as a trademark if a business opts to register it as such.

The terms “logo,” “trademark,” “service mark,” “certification mark,” “brand,” “mark,” and “label” are often used interchangeably, although there are key differences as far as their legal definition is concerned. In this chapter the terms “logo” and “trademark” are used without making any special distinction between them, to refer to any combination of graphic representation and arranged typeface used to distinguish an entity, product, or service.

## History and Importance

The use of distinctive marks to designate ownership goes a long way back in time with livestock ownership marks and pottery seals use dating back to the fourth millennium BC. After the fall of the Roman Empire, marks were increasingly used to identify product makers with a guild, which eventually led to the recognition of the property value of marks as a means to carry the reputation of a maker. The earliest enactment on marks was probably the Bakers Marking Law, introduced in England in 1266, which required bakers to mark their work with either pinpricks or stamps, followed by a similar requirement on silversmiths in 1363. Nevertheless, no strict legal framework protecting the trademark value was implemented until the early fifteenth and sixteenth centuries. In the USA, Thomas Jefferson advocated trademark laws in 1791, but it took until 1870 to pass the law on registration of trademarks and 1905 for the US Patent and Trademark office to be created. In the UK, the law on registration of trademarks was enacted in 1875. Numerous contenders exist for the earliest, continuously used mark, including Lowenbrau (since 1383) and Stella Artois (1366), while the oldest registered trademark in the world is considered to be the mark of “Bass & Co,” registered in the UK in 1875.

Nowadays, national or regional Intellectual Property offices around the world deal with the registration of trademarks, while the legal framework has been revised to serve the needs of a globalized market. The Madrid system for the international registration of marks was established in 1891 and offers a trademark owner the possibility to have his trademark protected in several countries. The Madrid system is administered by the World Intellectual Property Organization (WIPO), a specialized agency of the United Nations. In the EU, the Office of Harmonization for the Internal Market (OHIM) was established in 1994 to deal with European Union-wide trademark registrations.

According to the World Intellectual Property Organization, trademark applications per year worldwide have been increasing almost linearly over the past 25 years reaching about 3.5 million applications and 1 million new trademark registrations in 2010. The total number of trademarks in force around the world is estimated to

**Table 18.1** Trademark registration statistics for the year 2010 and selected IP offices. Note that new trademarks registered can exceed the number of applications due to back processing applications of previous years

IP office	Trademarks in force	Trademark applications	Trademark registrations
China	4,603,995	1,057,480	1,333,097
Japan	1,751,854	124,726	102,597
United States of America	1,544,184	281,867	167,641
France	1,119,000 <sup>a</sup>	93,187	4,250
Spain	887,122	47,120	41,092
OHIM (European Union)	609,373	98,616	102,227
Australia	446,760	59,459	39,943
Russian Federation	392,202	56,856	40,136
Germany	N/A	74,339	53,300
<i>Worldwide estimates (based on available data from 188 IP offices)</i>	20,259,654	3,481,763	2,954,227

Source: “2011 World Intellectual Property Indicators,” World Intellectual Property Organization Publication No. 941E/2011, ISBN 978-92-805-2152-8, 2011

<sup>a</sup>Data available for 2009 only

exceed 20 million (see Table 18.1). The vast majority of trademark registrations is graphical or combined textual and graphical.

Considering the above statistics, it is easy to appreciate the significant economic and social impact of trademark registration and use. Apart from the vast amount of man-months invested in managing the registration of new trademarks, a number of derivative services and applications exist that complement the trademark ecosystem. For example, enforcing the rights of ownership of registered trademarks is a commercial necessity that has resulted to the birth of trademark watch services, while numerous applications have been developed to leverage the recognizability of trademarks and logos to automate tasks such as incoming document categorization in digital mailroom implementations.

The automation of such services and applications is not trivial, as can be witnessed by the continuous investment in important research and development activities related to trademark recognition. Just to mention a few recent related projects, the EU project eMAGE (eContent programme, 1.1m €, 2004–2006) followed by eMARKS (eTEN programme, 2m €, 2007–2009) looked into a search service for images protected by IPR (trademarks and industrial designs). More recently, the EU project PROFI (FP6-IST, 1m €, 2004–2007) researched into perceptually relevant retrieval of figurative images (clip art, logos, signs). Over 2010–2012 the OHMI Cooperation Fund funded a 1.2m € project called

“Search Image” aiming to build a service to improve the current results of the figurative trademarks and design searches, revisiting among others the possibility to use computer vision-based techniques (until then considered “not mature enough”).

## Applications

The range of applications related to logos and trademarks stems from various, quite different in nature necessities. This section offers an overview of the main types of applications and the specific needs they address.

There are three main actors in the logo and trademark ecosystem: the trademark owners, the professionals that deal with the legal aspect of trademark management and registration, and a variety of end users that make use of logos to automate or facilitate daily tasks and processes.

The main concern of logo owners is to ensure the correct and efficient use of their logos. First, businesses want to make sure that their logos are seen by their target audience, which translates to achieving a certain degree of visibility and recognition. Second, they want to ensure that their corporate image is not misused, which implies monitoring and detecting any potentially wrong or illegal use of their protected marks. Typical applications required by logo owners relate to the detection of a small number of known a priori logos in unconstraint contexts like videos of events, or press images, but also trademark watch services where new trademark registrations are continuously monitored so that applications for similar trademarks can be identified and opposed to.

On the other end of the spectrum are the target audiences, the consumers of logos. These can cover anything from individuals looking for a place to shop to companies that need to make sense of their incoming mail. Logos are made for these end users; they are designed to be easily identifiable and are intended to increase recognizability. Consumers typically need applications that build on this fact and help them leverage the recognizability of the logos to facilitate or automate different tasks. Logo-based document classification is probably one of the most interesting and commercially exploited applications to examine in this context.

A third type of actors with a vested interest in the logo and trademark world are the government organizations and professionals that deal with the legal aspects of trademark registrations. For these professionals the major challenge is conducting searches for similar previously registered trademarks in large databases and identifying potential issues that would prohibit the registration of a new trademark. A crucial aspect of this application is the ill-defined nature of “similarity,” which, legally speaking, is not restrictive to visual similarity but it also covers the semantics and typically extends to aspects like phonetic similarity of words.

This chapter presents the current state of the art methodologies and best practices using three key applications related to the above domains to provide the necessary context. These applications are trademark retrieval systems, logo recognition in

**Table 18.2** Comparison of the different applications discussed in this chapter and summary of their respective challenges

Application	Input	Scale	Key challenges
Trademark retrieval systems	Logo-only images, high quality	Large number of models, typically in the 10–100 thousands	<ul style="list-style-type: none"> <li>• Scalability</li> <li>• Difficulty to define the concept of visual similarity (as opposed to matching)</li> <li>• Lack of public datasets</li> </ul>
Logo recognition in document images	Scanned paper documents, typically black and white	Small number of models, typically less than 50	<ul style="list-style-type: none"> <li>• Application-driven assumptions</li> <li>• Scalability</li> <li>• Handling of noise and document artifacts</li> </ul>
Logo detection and removal in images and videos	Unconstrained images or video sequences, typically colored	Average number of models, typically in the few hundreds	<ul style="list-style-type: none"> <li>• Generalization</li> <li>• Features applicability</li> <li>• Scalability</li> </ul>

document images, and logo detection in real scenes (images or video). The selection of these three applications has been made so that all different aspects of logo and trademark recognition are covered as shown in Table 18.2.

## Trademark Retrieval Systems

This section outlines the evolution of trademark retrieval systems over the past 20 years and details the state of the art in the topic. First, a problem definition is attempted and the key requirements for a trademark retrieval system are identified. Subsequently, frequently used trademark descriptors are detailed, before the focus is shifted on the aspect of retrieval itself and especially on the different ways to combine results obtained over diverse modalities and on tackling the issue of subjectivity through relevance feedback techniques. A summary of a limited number of selected representative systems is finally presented through a comparison table as well as a summary of the open challenges in trademark retrieval.

### Problem Definition

In most countries, trademarks must be formally registered with the national trademark office to gain legal protection. The objective of the registration process is to ensure that the trademark is sufficiently different from previously registered ones

to avoid confusion within the particular market segment(s) it addresses. The process of registering a new trademark is not trivial and can take more than a year before an application is accepted.<sup>1</sup>

Although different countries have slightly different rules and timelines, a key step of the process is invariably the examination of the application by an assigned attorney, an intensive process involving various searches in the registry to identify “similar” previously registered trademarks. Similarity in this context is defined on the basis of common sense. For example, in Indian law, Section 1(h) of the Trade Mark Act 1999 defines a “Deceptively Similar mark” as “...so nearly resembling another mark as to be likely to or cause confusion.” In the UK law, Section 5(2) of the Trade Marks act 1994 states “A trademark shall not be registered if because it is similar to an earlier trademark [...], there exists a likelihood of confusion on the part of the public, which includes the likelihood of association with the earlier trademark.”

This likelihood of confusion must be appreciated globally taking into account all relevant factors and every possible interpretation of a trademark image, “*including a visual, aural and conceptual assessment.*”<sup>2</sup> The legal definition of trademark similarity leaves a lot of space for subjective interpretation, frequently leading to ambiguous decisions.

In order to facilitate the searches, the trademark offices maintain details of all registered trademarks. Text-only trademarks are generally easy to compare and retrieve based on edit distance measures [47]. An extra complication arises from textual (word-only or device-and-word) trademarks which exhibit aural (phonetic) similarity for which special analysis is required [30].

Nevertheless, the major challenge, and main focus of this section, comes from graphical (also called figurative, device-only, or form-only) or combined textual and graphical (also called device-and-word or form-and-word) trademarks, which represent the majority of received trademark applications. For such trademarks, visual similarity is of prime importance. Figurative trademark searches are principally dealt with using predefined classification codes, the most widely adopted system being the Vienna classification [87] developed by the World Intellectual Property Organization. Such schemes address conceptual similarity, but no visual similarity.

An application very close to trademark retrieval is the so-called trademark watch service, where firms with registered trademarks are actively monitoring all new registrations of trademarks to make sure that they oppose to the admission of any trademarks similar to theirs. The key difference is that the large-scale retrieval process is substituted with a relatively small-scale comparison process, where new trademarks are assessed as similar or non-similar to a limited set of trademarks “under watch.” The main focus of this section is on trademark retrieval, but virtually all methodologies discussed here can be applied to trademark watch.

The application of logo retrieval for the trademark office is defined as a retrieval problem, where the objective is to find all similar, previously registered figurative trademarks given a query figurative trademark. The query is a single trademark image, which is usually a high-quality scan, or even a born-digital image that contains only the trademark in question; a few systems also tackle hand-sketched

query images. The samples in the collection of previously registered logos can also be assumed to be high-quality scans or born-digital images. Metadata are available both for the query and the collection samples and correspond to a list of Vienna codes. “Similarity” in the trademark retrieval problem should be considered at as many levels as possible covering all possible trademark interpretations, usually considering shape similarity, color similarity, and semantic similarity (typically based on the Vienna codes). Trademark retrieval systems aim to facilitate the human examiner by limiting the number of trademarks to check; hence, a high precision is important to ensure their usefulness. From the legal point of view though, achieving a high recall rate is paramount as all potentially conflicting trademarks should be retrieved and checked against.

## Trademark Descriptors

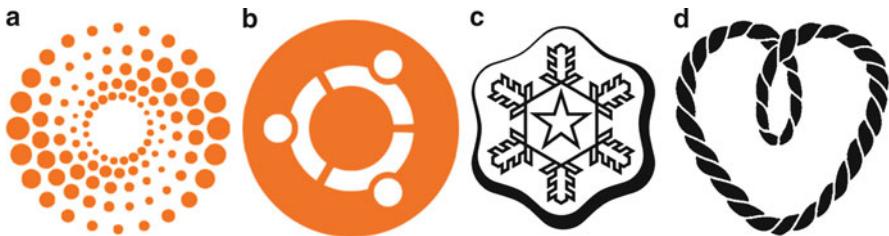
Searches for similar logos include a visual, aural, and conceptual assessment. The technical focus of this section is on visual similarity, although similarity over different domains is indeed discussed to the extent that it is important in the system design, as ultimately systems able to fuse information obtained over different domains are needed.

Searches for similar logos are usually performed over some notion of visual similarity, typically combining features over different modalities, primarily shape and color. Frequently, a core shape description is complemented with features related to other trademark properties, including color features, geometric features, topological information, texture features, and metadata such as semantic category codes. There is no single best-performing combination of features to use, as the importance of each visual modality to assess similarity depends on the particular design and intended use of the query trademark in question. In this section, the key visual descriptors used for assessing visual similarity are discussed as well as the use of semantic category codes as a complementary source of information.

## Perceptual Organization

Depending on the retrieval solution, trademarks are described at various levels of detail. Systems that extract features from individual connected components are common [39], while methodologies that encode the topological relationship between components [2] or systems that include certain preprocessing (e.g., filling [43] or dilation [1]) in order to eliminate fine details of the trademark have also been proposed. These last methodologies stem from the observation that “*the end-user’s recollection of a figurative trademark is of a general and hazy nature*” [43].

At the semantic level what is ultimately sought is an interpretation of the trademark image. This is naturally a difficult proposition for automated processes, more so as there are cases where more than one valid interpretation can be arrived at, while interpretations are usually context related. Nevertheless, it is a fact that human observers interpret the trademark image as a whole. After observing trademark examiners in action, Eakins [26] suggests that the examiners must identify and



**Fig. 18.1** Examples of bi-level figurative geometric trademarks. One distinctive feature of (a) and (b) is the central circular pattern, no matter whether it is made up by *dots* or *arc* segments. In (c) the most distinctive feature is the overall *star* arrangement of the trademark, while in (d) the single shape that stands out is the shape of a *heart*

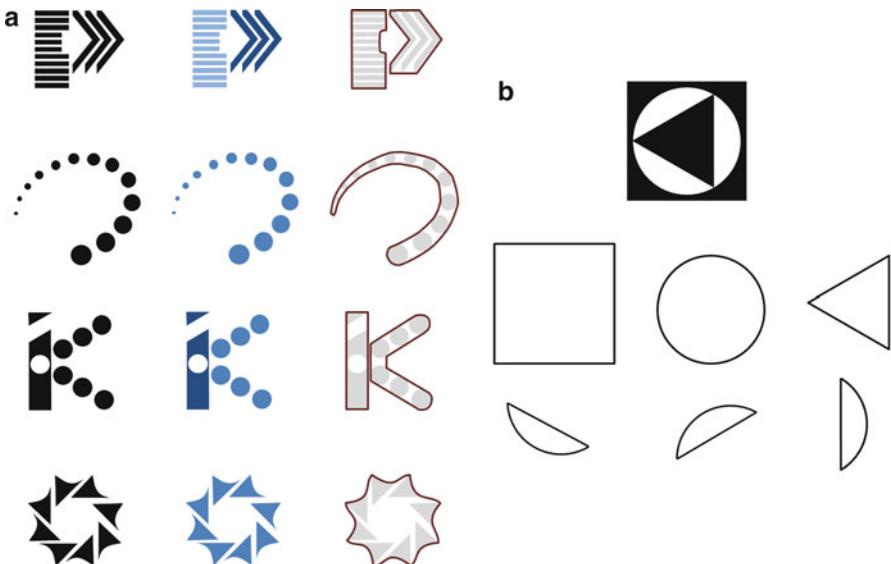
remember the most distinctive feature of the query image, which might be a single large shape or a group of objects making up a recognizable pattern (see Fig. 18.1).

In order to arrive to these higher-level interpretations, an automatic system has to be able to construe certain properties of basic structural elements in the image such as symmetries, continuity, proximity, and colinearity. This is a necessity for the considerable portion of trademarks that comprise abstract geometric designs (see Fig. 18.1) but less so for other pictorial trademarks. This functionality corresponds to an early stage process of human vision, called “visual grouping” or “perceptual organization.” An important function of perceptual organization is to distinguish as accurately as possible between accidental structures in the image and significant structures unlikely to have arisen by accident.

Based on the seminal work of Gestalt theorists [85, 86], a number of attempts have been made to provide computational models for the role of perceptual organization in the functioning of vision. These include early work by Lowe [51], Sarkar and Boyer [70], Amir and Lindenbaum [6], Saund [71], as well as more recent attempts notably by Desolneux [21]. The interested reader is referred to the above referenced work for technical details on the underlying computational models.

Various perceptual organization models have been used in the context of trademark shape analysis and are all viable options for extracting higher-level groups of basic elements for subsequent trademark representation. There is little comparative analysis of the different models in the trademark retrieval problem and no established best practice. See, for example, Alwis and Austin [5] who compared the bin-voting method suggested by Sarkar and Boyer [70] to the pairwise feature extraction process of Lowe [51] with mixed results. The selection of the best perceptual organization computational model depends on a number of system design decisions.

Perceptual organization can arise over different primitives from pixels to lines and components and over a varied set of properties such as shape similarity, color similarity, parallelism, orientation, continuity, and, importantly, proximity between the primitives considered. There is great variability to the choice of



**Fig. 18.2** (a) Perceptual organization modeled on connected components and resulting envelopes (Adapted from [1]). (b) Perceptual organization modeled on lines and curves and resulting closed curves (Adapted from [5])

primitives and associated properties to use. Alajlan [1] opts to perform connected component grouping over four properties: proximity, area, shape, and orientation, using the evidence accumulation framework to decide on the final groupings. Alwis and Austin [5] base the perceptual organization process on straight-line and arc segments and calculate pair-wise perceptual relationships between them such as end-point proximity, orientation similarity, curvature similarity, parallelism, collinearity, and co-curvilinearity. In the ARTISAN system [25–27], trademarks are expressed as a set of closed region boundaries which are subsequently examined to discover families of similar regions based on conditions that include close physical proximity of the boundaries, significant lengths of the boundaries being collinear or parallel, symmetry or shape similarity of the corresponding regions, and boundaries enclosing areas of similar color or texture.

When working at the level of closed boundaries of connected components, the final groups are usually approximated by an envelope shape enfolding the grouped regions (see Fig. 18.2a). When working at the level of lines or curves, the resulting groups are usually closed boundaries that represent alternative interpretations for the trademark (see Fig. 18.2b). In both cases the resulting groups are the basis for subsequently calculating shape descriptors. Consequently, indexing and retrieval is based on a combination of individual components, detected groups, and whole-image descriptors.

Although perceptual organization is accepted as an important stage in trademark image understanding, the performance of current computational models varies a

lot depending on the trademark image. Ren et al. [63] performed a comparison between human-obtained and ranked segmentations of trademark images into their constituent parts and the segmentations produced by the ARTISAN system. They report an average performance of about 58 % of the automatically produced segmentations matching within one grouping mistake one of the top human-produced segmentations. About 28 % of the automatic segmentations were in a perfect agreement with the humans' first ranked choice. A qualitative analysis of the failure types shows that the most important source of confusion is the existence of textured areas, while over-grouping was also the cause of discrepancies in numerous cases.

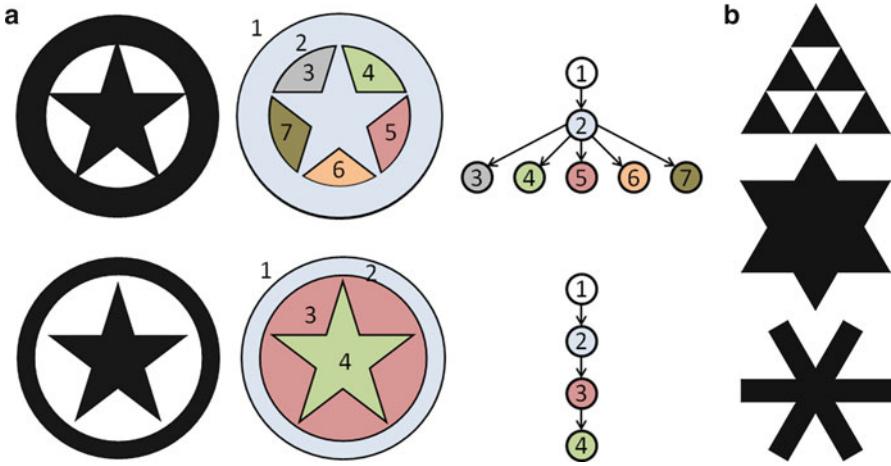
Apart from perceptual organization computational model shortcomings, it is worth noting that perceptual organization approaches implicitly assume that trademark images comprise a number of easily separable parts. This is indeed the case for a sizeable portion of figurative trademarks registered with the exception being photographic trademarks which cannot be treated in this way. Overall, perceptual organization is considered an open challenge, while the domain of trademark retrieval presents a very interesting test bed for novel methodologies.

### Visual Similarity: Shape and Topology

The main modality over which visual similarity is assessed is the trademarks' shape. During the past two decades, virtually every type of shape descriptor has been used for trademark retrieval at various detail levels from individual trademark parts, to "envelope" shapes grouping a number of separable parts, to global shape descriptors over the whole trademark image.

Simple geometric features such as aspect ratio, circularity, compactness, central moments, Euler number, and eccentricity as well as Rosin's features (triangularity, rectangularity, and ellipticity) have been used extensively [2, 13, 27, 28]. These are a common choice when description of individual parts is sought, as they are simple and fast to calculate. More complex contour- and region-based shape descriptors have also been broadly used, including among others Fourier descriptors [27, 84], Zernike moments [45, 84], Angular Radial Transform [27], Hu moment invariants [27], Curvature Scale Space [27, 84], shape context [11, 66, 67], grid-based pixel density [25], and edge direction histograms [43]. The reader should refer to ►Chap. 16 (An Overview of Symbol Recognition) for a detailed account of cited shape descriptors.

Eakins et al. [27] performed a comparative retrieval study between whole-image versus part-based trademark image matching and found strong support for part-based matching being significantly more effective than whole-image matching. A number of interesting observations can be made on this topic. First, the descriptors used for part-based matching are usually simple geometric features, which are much faster to calculate than whole-image descriptors. This counteracts to some extent the increase in computational complexity occurred due to the need to index and search a much larger number of parts compared to whole images. A downside of part-based approaches is that they implicitly target abstract geometric trademarks comprising



**Fig. 18.3** (a) Two very similar trademarks with quite distinct internal topologies. (b) Example of different trademarks with virtually the same edge direction histograms, peaking at  $0^\circ$ ,  $60^\circ$ , and  $120^\circ$  (Adapted from [45])

a number of easily separable parts, rendering them difficult to apply in the cases of photographic trademarks.

Regarding the efficiency of different shape descriptors, direct comparisons between published methods are difficult to perform as authors rarely use the same datasets or the same performance evaluation metrics. Nevertheless, comparisons performed independently by various authors on their respective datasets converge to similar conclusions [27, 84], also confirmed independently of the trademark retrieval applications [90]. Based on whole-image matching performance, Zernike moments seem to perform consistently better for trademark retrieval than Curvature Scale Space (CSS), Fourier, or Hu moment invariants which yield similar performances. It is also a common finding that an early fusion of different shape descriptors results to higher discriminating power and an increase in performance (see, e.g., Fig. 18.3 for the shortcomings of using single descriptors).

A number of authors have proposed ways to bridge the local (usually part based) and global description levels. Alajlan et al. [2] use curvature trees to represent the topology of trademark parts (foreground) and holes (background) and report better accuracy than employing single global-shape descriptors. Nevertheless, they observe that the topology of trademark images can be quite unstable and change due to noise, resolution changes, occlusions, etc., while there are cases where a topological description might not be a good basis for measuring similarity (see, e.g., Fig. 18.3a). Avoiding the requirement for a stable segmentation, Wei et al. [84] propose to describe the internal structure of a trademark based on statistical features calculated locally over the edge map of the image. These are combined with global shape features in a late fusion scheme. The results indicate that such an approach

**a**

591

...  
DE - rot, orange, gelb, grün, blau, violett, schwarz, weiß  
EN - red, black, white, blue, yellow, orange, green, purple  
FR - Rouge, noir, blanc, bleu, jaune, orange, vert, violet  
IT - Rosso, bianco, nero, giallo, arancione, verde, blu, viola

531

25.12.99  
26.2.7

**b**

"... THE COLOR(S) ORANGE, GREY, YELLOW AND GREEN IS/ARE CLAIMED AS A FEATURE OF THE MARK.  
THE MARK CONSISTS OF THE A DESIGN OF A SMILING BABY WITH GREY FACIAL FEATURES IN AN ORANGE, YELLOW AND GREEN BLANKET. TO THE RIGHT OF THE DESIGN IS THE WORDING "TRAVELING BABY COMPANY" IN STYLIZED GREY FONT ABOVE THE WORDING "COMFORT DELIVERED" IN SMALLER STYLIZED ORANGE FONT. THE FOREGOING ELEMENTS APPEAR WITHIN AN ORANGE OBLONG SHAPED BORDER. THE COLOR WHITE APPEARING IN THE MARK REPRESENTS BACKGROUND OR TRANSPARENT AREAS AND IS NOT CLAIMED AS A FEATURE OF THE MARK..."

**Fig. 18.4** (a) A trademark description as it appears in the European OHIM “Community Trademarks Bulletin,” code 591 of the description corresponds to the list of *colors* in the image, while code 531 corresponds to the Vienna classification. (b) A trademark description in the USA “Official Gazette for Trademarks,” the *color* content and its use in the image is explicitly described in the text

can yield significantly better retrieval rates than a number of global shape descriptor combinations compared against.

Techniques for shape and object recognition based on local descriptors are gaining popularity. An example of such a descriptor used for trademark matching is the shape context, proposed by Belongie et al. [11]. In [11] matching between trademarks is performed through bipartite graph matching and the Hungarian method, yielding very good results but being difficult to scale up for large databases of trademarks. Rusiñol et al. [66] made use of the shapeme histogram descriptor [56], combined with locality-sensitive hashing to achieve approximate k-NN-based retrieval in sublinear time, a methodology later extended to incorporate color information [67].

Shape is a key modality over which visual similarity is assessed. Although there is no best shape descriptor for the trademark retrieval application, there is consensus that multiple-level (part-based combined with global) descriptors work better than any single ones, while combining various shape descriptors is generally necessary, especially if diverse types of figurative trademarks are targeted (e.g., geometric and photographic).

### Visual Similarity: Color Content

In the case of figurative trademarks, registering them in color can have certain legal implications. In general, logos are submitted in black and white in which case the registered trademark covers the rendering of the logo in any color combination. But if the applicant claims color as a feature of the mark, then the particular color scheme is explicitly included in the registered trademark description<sup>3</sup> (see Fig. 18.4).

For all trademarks that are registered in color, the specific color scheme used is intended to be a recognizable feature and strongly associated to the brand, hence important to take into account during the search (think, e.g., of the Coca-Cola red/white logo, the BMW blue/white logo, or the colored Burberry pattern). According to the OHIM manual of trademark practice<sup>4</sup> “*the use of the same color or color pattern may increase a visual similarity of the figurative or word elements themselves. The exact effect of colors or color patterns has to be assessed individually in each case, since this depends very much on the impact of the color on the overall impression of the signs involved.*” Therefore, as far as trademark registration searches are concerned, both color similarity and shape similarity should be taken into account as “confusingly similar” trademarks can arise in either domain.

Contrary to logo detection in videos and real scenes, relatively few trademark retrieval systems incorporate color in the trademark descriptors. Within these systems, color is variably used either as local feature [67], complementary to part-based shape descriptors [39], or as a global feature at the image level [68]. Mere coincidence of the color if the figurative or word elements are not otherwise similar is not legally considered enough to lead to a relevant similarity. It can therefore be argued that color information should be always assessed in the context of its spatial arrangement and global color features are less relevant in the trademark retrieval application.

Employing color as a feature aims to improve the retrieval of visually similar (as opposed to semantically close) trademarks; hence, it is a reasonable assumption that perceptually relevant color systems and metrics should be used to assess color differences (see, e.g., [68, 88]). This is not yet common practice, with most authors opting instead to more computationally efficient color representations.

It is also worth noting that figurative trademarks comprise highly stylized graphics, made up of few, identifiable colors. Therefore, methodologies such as color naming or other fixed color categorization schemes offer a valid alternative to color information encoding given the nature of trademark images. In this line, Rusiñol et al. [67] combine a statistical color naming model and shape context features in a late fusion scheme to perform trademark retrieval. Although color names information is included in the trademark description (see Fig. 18.4), no existing methods make use of the registered color names, bridging the metadata and image domain to enhance image-based trademark retrieval.

Overall, color information has not been extensively incorporated in trademark retrieval systems up to date, although there is a consensus that incorporating color information boosts retrieval results. Given the growing number of trademarks registered in color, this is one aspect of trademark retrieval that is expected to receive increasing attention in the years to come.

### **Conceptual Similarity: Semantic Categories**

Another key modality in which a likelihood of confusion might arise is conceptual similarity. According to current trademark practice “*Signs are conceptually identical or similar when the two signs are perceived as having the same or a similar semantic content.*”

The main instrument available to the registration offices to detect such conceptual similarity is the Vienna classification system [87], which offers a comprehensive list of tags under different categories, adopted by trademark registrants around the world. An excerpt of the categories tree is shown below.

Category 1 CELESTIAL BODIES, NATURAL PHENOMENA, GEOGRAPHICAL MAPS

Category 2 HUMAN BEINGS

Category 3 ANIMALS

Category 4 SUPERNATURAL, FABULOUS, FANTASTIC OR UNIDENTIFIABLE BEINGS

Category 5 PLANTS

### 5.1 TREES, BUSHES

5.1.1 Trees or bushes of triangular shape, conical shape (pointed at top), or “candle-flame” shape (firs, cypresses, etc.)

5.1.2 Trees or bushes of oblong shape (poplars)

5.1.3 Trees or bushes of some other shape

5.1.4 Trees or bushes without leaves

...

Category 6 LANDSCAPES

Manual classification of images is time consuming and potentially error prone [26]. There is ongoing research in methodologies for the efficient use of such hierarchical classification schemes to improve retrieval. See, for example, [17] for a trademark retrieval system based solely on the semantic description of an image, where the analytical hierarchy process is used to manually assess the pair-wise relative importance of different categories within a trademark image at the time of categorization. Although such systems technically fall outside the context of this chapter, they should nevertheless be considered as baseline for evaluating CBIR approaches.

Cautiousness is advised when relying on classification codes for retrieval. A drawback identified by various authors is that classification codes are not helpful for abstract images (a sizeable portion of registered trademarks comprise geometric designs that have little or no representational meaning), while they are inadequate to describe such attributes as color, shape, texture, and layout [26, 43]. Figure 18.5 shows a number of trademarks categorized under the same code (05.01 – “Trees, Bushes”). It can be easily appreciated that conceptually similar trademarks might show no visual similarity whatsoever. Conceptual similarity might come into play for two signs as a whole or in comparing elements of composite signs. Moreover, conceptual similarity between the word and the figurative aspects of different trademarks (e.g., the word mark “tiger” against a figurative mark depicting a tiger) is also legally accepted as a basis to establish a likelihood of confusion condition and is therefore searched for by the officers. The main conclusion to draw is that semantic category tags and visual descriptors are highly complementary.

A limited number of trademark retrieval systems combine visual similarity search with semantic information to improve or refine results. In certain cases a



**Fig. 18.5** Trademarks categorized under code 05.01 – ‘Trees, Bushes’ of the Vienna classification (Trademarks obtained through the eSearchPlus online service of OHIM (May 2012), accessible online at: <http://esearch.oami.europa.eu/copla/advanced#/trademarks>)

prior keyword-based search is assumed to have taken place to reduce the size of the search space before content-based image retrieval methodologies are applied [43]. Combining shape, color, and semantic information (afforded through Vienna codes) is attempted in [68], through the use of predefined weights to specify the relative importance of each modality. In [88] a fuzzy system is used to calculate the distances between sets of terms, which are subsequently combined with image-based descriptors.

The existence of an international classification standard should be taken into account both as a possible means for training object (concept) detectors and in its capacity as a complementary retrieval domain enabling multimodal retrieval systems.

## Information Fusion, Indexing, and Retrieval Strategies

Contemplating the functioning of a complete trademark retrieval system, it is important to remember that three domains are legally defined where a likelihood of confusion might arise: visual, aural, and conceptual. Moreover, within the visual domain, which is the main focus of this chapter, numerous modalities can be taken into account, shape and color being the most prominent ones discussed here. An extra complication when comparing pairs of trademarks is that different aspects of their description, not known *a priori*, bear more importance than others, making it difficult to properly weight the contribution of each description modality.

Developing a working trademark retrieval system implies selecting the right strategies to efficiently fuse information from different domains. The best strategy to use is inherent to the structure and focus of each system. An overview of the main schemes along with a discussion of advantages and disadvantages is given here.

### Early Fusion and Unimodal Retrieval

Approaches that rely on early fusion first extract unimodal features which are then combined into a single representation based on which classification and retrieval is

performed. The main advantage of such schemes is their computational efficiency at the time of retrieval, especially for relevance feedback techniques as a single feature vector representation is involved. Nevertheless, the meaningful normalization of features extracted over different domains and modalities (e.g., shape, color, texture, semantic codes, aural representations) is generally difficult to achieve.

Early fusion schemes frequently boil down to unimodal retrieval systems, most commonly shape similarity-based systems that combine a variety of distinct shape features.

### **Late Fusion and Multimodal Retrieval**

Late fusion is a relatively easier way to deal with information originating over different domains. Late fusion focuses on the individual strength of separate modalities, and in the trademark retrieval application, it is usually employed in two distinct manners. Before retrieval, unimodal similarity (distance) scores are fused into a single multimodal similarity (distance) measure which is then used to rank results. Otherwise, retrieval can be performed separately using different descriptors, subsequently combining the obtained ranked lists of results.

Usually, similarity scores obtained over different modalities are fused through a weighted average scheme. Other choices include different operators (e.g., sum, multiplication, max [39]) and fuzzy systems. An early example of this practice is presented in [88]. The system enables comparisons between word marks, between device marks, and between marks of different types through a sequence of heuristics based on weighted distance calculations. For word marks, edit distance, phonetics, and interpretation scores are calculated independently and subsequently combined based on user-defined weights. Similarly for device marks, meaning (based on Vienna classification codes), structure, and shape similarities are combined. Finally, to enable comparison between marks of different types type-mismatch penalties are introduced.

A typical need of trademark retrieval applications is the combination of retrieval results (ranked lists of similar trademarks) obtained over different modalities. Using ranks for this fusion has advantages over combining incommensurates which differ in range, mean, and variance since they come from different representations and mechanisms [10]. Reciprocal rank fusion, Borda count, logistic regression, and the highest rank methods are typical choices for combining ranked lists obtained from multiple retrievals [2, 5, 39].

It is important to note that when working with semantic categories, there is no implicit ranking between trademarks that share the same number of classification codes. Instead, where a retrieval system would produce ranked lists for modalities such as shape and color, it produces separate sets of equally ranked trademarks (sharing the same number of categories with the query trademark) in the case of semantic information. For such cases, it is important to use methodologies that take care of the ties in the semantic results, such as the Condorcet method (see, e.g., [68]). A downside of this kind of methods is that they are usually computationally expensive as they require a lot of pair-wise comparisons; hence, approximate solutions or prior filtering of the results should be sought.

### Multistage Retrieval

A different variety of systems employ a multi-scale retrieval strategy where a cascade of filters are applied, aiming to reduce the search space before a final ranking is calculated among the filtered trademarks. This approach is common with perceptual grouping-based methods and methods that combine multiple-level (part-based and whole-image) descriptors. It also provides an alternative way to define the relative importance of different modalities, reflected on the order of the applied filters.

Systems based on multiple-level descriptors usually employ a two-stage retrieval scheme, where global features act as a kind of hash variable to retrieve a subset of trademarks which are then compared on a part-by-part basis. As an example, in the ARTISAN system [25], a hierarchical record is created for each image with shape information stored at each level (whole image, family, boundary). Retrieval is accomplished by first comparing whole-image shape features, deemed more generic, and then traversing ordered lists of image components comparing each query component with the closest-matching stored one.

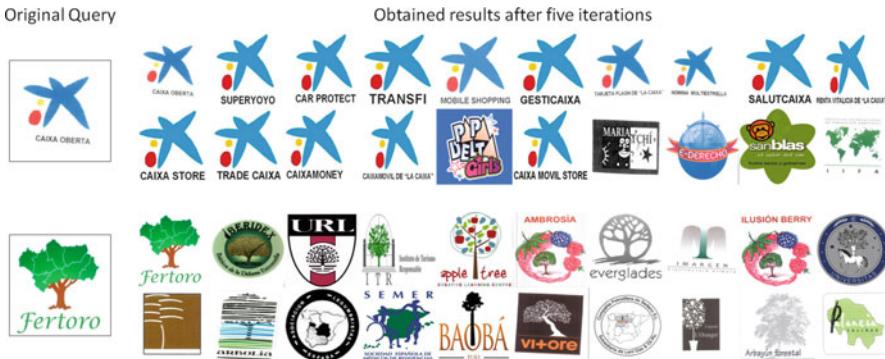
On a similar line, Hung et al. [42] calculate two shape features per trademark image, a rough contour-based descriptor and a more detailed region-based one. The database of trademark images is categorized automatically into 11 classes based on the contour feature. During retrieval, the contour descriptor of the query image is used to dynamically select the possible classes that the image belongs to, forming candidate sets with different priorities, then the region-based descriptor (Angular Radial Transform) is employed to decide on the final ranking. Jain and Vailaya [43] use a similar process, performing a rough retrieval based on edge direction histograms and invariant moments, followed by a refinement stage based on deformable template matching.

An example spanning different modalities is the methodology of Hsieh and Fan [39]. They follow a two-step similarity assessment, where first shape and topology similarity probabilities are combined, and then the result is combined with the color similarity probability. Other typical applications of multistage retrieval techniques are when semantic categorization is available and is used as a first filtering.

### Relevance Feedback

The importance of different modalities in the visual similarity search is to a great degree a function of the particular trademark under question. As this is not easy to define prior to the search, certain trademark retrieval systems allow the search officer different ways to decide the relative importance of the different modalities. This is achieved either by adjusting the relative weights of different features manually or dynamically by allowing for a relevance feedback process.

Relevance feedback is a user-controlled iterative process for query reformulation. The key concept is to iteratively improve the retrieval result by emphasizing previously retrieved relevant items and de-emphasizing irrelevant ones. The two main variants work either by reformulating the query in every step (e.g., the Rocchio algorithm [64]) or by revising the similarity metric (e.g., Giacinto and Roli [34]).



**Fig. 18.6** In the *top* query, the user opts to select relevant images that are visually similar to the original query. In the *bottom* query, the user opts for trademarks that are conceptually similar (they contain some representation of a *tree*). The system adapts accordingly (Reproduced from [68])

Numerous examples of relevance feedback-based systems for trademark retrieval have been reported in the literature, including Chou and Cheng [17] who propose a semantic-only system using Rocchio algorithm and Ciocca and Schettini [18] who propose an algorithm that updates both the weights used to combine the different shape features as well as reformulate the query accordingly in every step.

Applying relevance feedback over three modalities independently, Rusiñol et al. [68] propose a framework that combines shape, color, and semantic information (Vienna codes). The system considers the three modalities independently, producing ranked lists for shape and color and separate sets of trademarks sharing the same categories in the case of the semantic information. These are combined using the Condorcet method in order to take care of the ties in the semantic results. A relevance feedback mechanism then allows the user to shape the query in real time. In reality, the three queries (independent modalities) are independently updated and the ranked results are fused as above in each step. Two typical retrieval scenarios are shown in Fig. 18.6.

Relevance feedback is an important addition to trademark retrieval systems, accepting the limitations of automatic descriptors in covering the whole space of possibilities and the importance of expert's feedback to the search process. Relevance feedback allows the expert user to direct the search towards the desired outcome focusing on visual or conceptual similarity, as required by the trademark under question, on the basis of his iterative feedback.

## Systems Overview

Table 18.3 attempts to provide a summary of the key aspects of selected methods spanning the past 15 years of research in trademark retrieval systems. This is not supposed to convey a complete list of trademark retrieval systems but instead

**Table 18.3** Comparative summary of different trademark retrieval systems

Name	Year	Description	Shape descriptor	Color	Semantic categories	Perceptual organization	Dataset	Relevance feedback
Wu et al. [88]	1996	Part-based	Fourier coefficients, moment invariants, projection histograms	N/A	Vienna codes, processed into a Fuzzy thesaurus	User-assisted grouping	3,000 color trademarks, scanned in-house	N/A
Kim and Kim [45]	1998	Image level	Zernike moments	N/A	N/A	N/A	3,000 bi-level trademarks, scanned in-house	N/A
Jain and Vailaya [43]	1998	Image level	Histogram of edge directions, moment invariants, deformable template matching	N/A	N/A	N/A	1,100 bi-level images, scanned in-house	N/A
Alwis and Austin [5]	1998	Multilevel	Graph-based representation, parts described by primitive features	N/A	N/A	Pair-wise feature extraction (Lowe and Sankar/Boyer)	1,000 bi-level trademarks, subset of [26]	N/A
Hsieh and Fan [39]	2001	Part-based	Fourier coefficients, topological relationships between parts	RGB or HSI	N/A	N/A	155 color trademarks, scanned in-house	N/A

Eakins et al. [25–27]	2003	Multilevel	ART, moment invariants, Fourier, CSS, Rosin descriptors	N/A	N/A	Boundary families based on proximity, symmetry, and colinearity conditions	10,745 bi-level trademarks from UK trade mark registry	N/A
Alajlan et al. [2]	2006	Multilevel	Curvature trees, Triangle Area Representation (TAR)	N/A	N/A	N/A	1,500 bi-level logo images (400 from MPEG-7 CE-2 database, 1,100 from [43])	N/A
Alajlan [1]	2007	Multilevel	Eccentricity, solidity, orientation for the parts, TAR for final envelopes	N/A	N/A	Hierarchical clustering and evidence accumulation	110 bi-level trademarks	N/A
Wei et al. [84]	2009	Image level	Zernike moments, edge curvature, and distance to centroid	N/A	N/A	N/A	1,003 bi-level trademarks, 14 classes	N/A
Rusiniol et al. [67]	2010	Image level	Shape context	Color names histogram	N/A	N/A	323 color trademarks, in 24 classes	N/A
Rusiniol et al. [68]	2011	Image level	Shapeme histograms	CIE LCh color histogram	Vienna codes	N/A	~30,000 color trademarks from the Spanish IP Office, 1,350 Vienna categories	Rocchio

to offer a representative selection of methods that cover in various combinations all the topics discussed in this section. It is extremely difficult to provide any direct comparisons between the different systems exposed as both the datasets and evaluation metrics used vary widely.

## Open Challenges in Trademark Retrieval

For a trademark retrieval system to be practical and commercially successful, it has to be able to work in real time, achieve a recall rate close to 100 % (no potentially similar trademarks missed) with a high precision (minimum number of irrelevant trademarks shown to the officer), and allow for variable definitions of similarity that span different domains. A number of open challenges stand on the way to achieve this, a summary of which is attempted next (see also [73] for an interesting discussion).

### Scalability and Variability Issues

The number of trademarks registered worldwide is estimated to exceed 20 million. The majority of registered trademarks are actually word-only ones, substantially bringing down the number of device, and word-and-device ones, which are the main focus of this chapter. From the point of view of the collection size, this number of samples is manageable. Nevertheless, when considering that multiple retrieval processes might be necessary to tackle different modalities or when considering methodologies based on large numbers of local descriptors extracted from each image with their associated matching processes, it becomes obvious that a number of the techniques currently proposed would not scale gracefully with the size of trademark collections.

The variability of trademarks is another source of challenges. Trademarks are formally classified as word-only, word-and-device, and device-only ones (leaving out special trademarks like 3D, smell, and color per se) depending on their content. But within these categories there is substantial variability including geometric versus photographic trademarks, monochrome versus color trademarks, and stylized text versus plain word-only trademarks. It is commonplace that individual methods put forward only address a particular subtype of logos.

### Availability of Resources (Datasets and Ground Truth)

It might come as a surprise that although an intricate international structure exists to register and control trademarks, it is relatively difficult to encounter publicly available collections of trademarks. Although trademarks are available to browse online one by one, they are not available to download and use as a collection. As a result, the use of private trademark datasets is widespread.

A consequence of the lack of any standardized dataset is the lack (and difficulty to obtain) of associated ground truth data to train and evaluate trademark retrieval systems on. A direct way to evaluate such methods would be to obtain and compare against human expert responses given the same test queries and dataset. A few

authors attempted this with limited data [13, 43], but the sheer amount of time required to obtain such responses makes this unpractical. Jain and Vailaya [43] report that it took each subject between 1 and 2 h to select the top ten similar results for 5 query images, within a dataset of 1,100 geometric-only trademarks. In the absence of any (easily obtainable) ground truth, performance evaluation is often based on a posteriori assessment of the returned queries, which gives a subjective qualitative measure but no quantitative evaluation.

### **Visual Similarity and the Importance of User Feedback**

Visual similarity is an ill-defined concept, as it can arise over different attributes for any given pair of trademarks. There is no way to define a priori the relative importance of the different modalities for assessing visual similarity. Two common practices exist to partially alleviate this problem. On one hand, a trademark retrieval system should take into account all possible interpretations of a trademark image. This entails performing multiple visual similarity checks over different modalities or over alternative descriptions (e.g., different groupings of trademark parts) and either combining their results or directly presenting alternative results to the user. On the other hand, relevance feedback methodologies let the user implicitly decide over what modalities search should be performed, iteratively adjusting the relative significance of different attributes.

A couple of observations can be made here. First, it seems that the increased participation of the user is necessary. This opens up new problems both in terms of user interfaces (see [80]) and on how to best learn from the user feedback, what calls for interdisciplinary research. Schietse et al. [73] suggest various ways to achieve the improved involvement of the user, including the ability to specify how the query should be formulated (e.g., based on a complete image or on specified parts, define the search parameter weights such as shape and color), relevance feedback, and improved display paradigms.

Second, the trademark offices' databases contain a lot of metadata information that is not efficiently used by current methods. For example, the fact that a trademark is submitted in color is important as it signifies that color has an increased significance in this particular case. Future systems should examine how this information can be best used to fine-tune the searches. Use of metadata information can be made both for learning and for filtering and improving results presentation. The existence of millions of logos classified under the Vienna classification is a valuable resource that could drive graphics recognition and training of complex object detectors.

### **The Difference Between Similarity and Matching**

The aim of trademark retrieval systems is to identify visually similar trademarks as opposed to perform exact image matching. The difference is subtle and should indeed boil down to relaxing sufficiently the similarity thresholds used. Nevertheless, many of the trademark descriptors employed were never designed to be used like this; instead they were designed to be robust in an image-matching scenario and have a sharp response as similarity decreases in order to reduce false positives. Another way to say this would be that depending on the trademark descriptor, small

changes in the distance threshold do not necessarily correspond to a human notion of smoothly decreasing similarity.

### Future Outlook

Overall, trademark retrieval, as well as trademark watch, is an open area for research with clear commercial value. Most existing research on CBIR is focused on natural images. Techniques for trademarks, but also for other artificially produced images such as icons, logos, coats of arms, and clip-art images, have received less attention, even though there is evidence that these images require a different set of techniques for effective retrieval [73].

The topic of trademark retrieval offers a lot of opportunities for future research. Apart from document analysis-related research, looking into better and more perceptually relevant shape, texture, topology, and color representations for this kind of images, there is also a wide scope for multidisciplinary research. A non-exhaustive list of research areas with potential impact includes the enhanced involvement of the user, scalable indexing and retrieval methodologies, the efficient use of information from nonimage domains, and research on better perceptual organization models.

---

## Logo Recognition in Document Images

This section discusses methodologies for the detection and recognition of logos in document images. It also highlights the usage of logos for document classification and retrieval purposes. First, a problem description is offered. The section continues with a discussion on the detection and localization of logos in documents, followed by the particular use of logos for document classification. Finally, a summary of the open challenges in logo detection in document images is given.

### Problem Definition

Logo detection and recognition in document images is of great interest for automatic document processing, since logos are indicative of the document source [91]. As such, logo detection and recognition facilitates several aspects of the document processing pipeline as it offers a viable solution to document classification and indexing (see also ►Chap. 7 (Page Similarity and Classification) for alternative methodologies to page classification). To illustrate the economic importance of such tasks, it suffices to at digital mailroom implementations. It is estimated that companies receive on average three million documents per year, more than half of which originate in paper. Based on recent market research in the UK, the cost to classify and archive a document is circa £1, while the cost of searching for a document ranges between £5 and £300.<sup>5</sup> In this section the different logo-related processes involved in such tasks are discussed, namely, logo detection, logo recognition, and logo spotting.

Logo detection refers to the process of asserting whether a document contains areas that could possibly be logos and localizing such areas. Generally, no specific *a priori* defined logo models are assumed, instead global logo characteristics are employed. Logo recognition then refers to the subsequent process of classifying a logo candidate area to one of a list of predefined logo models.

Logo spotting on the other hand refers to the process of detecting whether a predefined query logo appears in the target document. Logo spotting typically involves an off-line processing stage where incoming documents are analyzed and indexed based on local features obtained over identified keypoints or key regions. Following this, given a query logo, spotting becomes a retrieval exercise in the indexed keypoints (or key region) domain. Given the recognizability-by-design of logos, searching for documents cast as a logo-spotting exercise that proves to be a highly effective way to retrieve relevant documents.

Real-world applications feature continuous document flows that need to be processed in real time. The time allocated to document categorization is limited; hence, a key aspect of all above tasks is their ability to perform in a close to real-time manner. This issue is typically manifested as a scalability requirement, where proposed methodologies should be easily scalable both in terms of the number of documents processed per unit time and in terms of the number of logo models considered.

## Detection

Several methods in the literature related to logos in document images deal with the logo recognition problem assuming that the detection of logos has been performed by a separate module [15, 22, 35, 57]. On the other hand, there are methods that attempt to detect and localize instances (if any) of a logo in a document page. One way to distinguish these methods is according to the analysis level used for the description of the document's content. There are two levels of description. The first one is based directly on the pixel color information without any preprocessing, while the other involves some kind of preprocessing or layout analysis in order to extract higher-level structures from connected components or regions.

### Pixel-Based Detection

Approaches in this category do not involve any preprocessing step, and the focus is on finding pixel-based local structures that are likely to be logo candidates. For example, in [61] a training-free unconstrained logo detection method is described based on the principle that the spatial density of the foreground pixels within a given windowed image that contains a logo is greater than those of non-logo regions. Each pixel is considered as a potential cluster center of a windowed area, and its spatial density is computed in order to decide if it is a logo or not.

In a similar way, Wang and Chen [81] proposed a method based on the assumption that almost all documents with logos keep a relatively larger distance for the logo from other parts in the documents. In their algorithm, each foreground

pixel is taken as a seed feature rectangle which is growing in order to embrace the logo-candidate area. A tree classifier is then employed to quickly discard the candidates whose likelihood to be logos is very small.

The advantage of the pixel-based approach relies on its robustness in case where the content of the document image is subject to small changes of low variability. Moreover, pixel level techniques may be efficiently implemented by fast image processing operations that are benefiting by the local distribution of pixels (windowing approaches, parallel calculations, etc.)

### **Region-Based Detection**

Instead of using the pixel information directly, there are several methods where the binarized image is segmented either into labeled areas (e.g., text, image, graphics) or into a number of connected components which are then described by properly defined features. For instance, Zhu and Doermann [91] proposed a multi-scale boosting approach that involves Fisher classifiers in order to detect logo candidates in a document image. An initial two-class classification is performed at a coarse image scale on each connected component. An identified logo candidate region is successively classified at finer image scales by a cascade of simple classifiers, which allows false alarms to be quickly rejected and the detected logo to be more precisely localized. The context distance, the spatial density, the aspect ratio, and the area are some of the features used to describe the connected components.

In an early approach, Seiden et al. [74] considered the problem of classifying segments of a binary document image according to whether or not they are likely to contain a logo. Segmentation is performed using a top-down, hierarchical X-Y tree scheme. For each segment a set of 16 features are derived that are based on statistics about the connected components of black pixels within the segment. A subset of segments is used as training set from which classification rules are derived based on the C4.5 algorithm. The rules are then used to classify the remaining segments based on whether or not they are likely to contain a logo or not.

Overall, if layout analysis or segmentation into connected components can be efficiently applied on the document under consideration, then this approach is advantageous since the extracted higher-level structures have a semantic importance that is meaningful and can be further explored. However, region-based detection methods are intrinsically prone to any errors inheriting from the layout analysis or the connected component extraction step.

### **Recognition**

Given a detected logo candidate found in a document image, logo recognition attempts to classify the logo as belonging to one of a finite number of logo classes or conclude that it does not belong to any known class. The efficiency of the recognition system relies on the discriminative power of the features that describe the logo candidate image as well as on the robustness of these features on distortions and noise. Virtually all the descriptors discussed previously can be applied to

recognition of logos in documents. However, in contrast to the trademark retrieval process where the goal is asserting visual similarity (trademarks spanning various classes are expected to be retrieved), here the focus is on recognition, namely, on deciding on a specific logo class given the detected candidate area.

### Fusion of Structural and Topological Features

A characteristic that distinguishes a logo from other visual objects is that it consists of several parts whose structure and topological relations is important for the description of the logo. Feature-based approaches are characterized by high sensitivity to the features selected for representation and are unable to represent any specific information about the structural relationships among components. Therefore, a desirable property is the ability of a method to take into consideration the local structure information as well as to handle structural changes. Several authors tackled this problem by proposing extensions of classical Artificial Neural Networks (ANN). For example, Diligenti et al. [22] proposed to represent the input patterns by means of directed ordered acyclic graphs where the nodes may contain either symbolic or real-valued features. A properly extended version of the classical multilayer perceptron called backpropagation through structure is also involved so as to take into account both the numerical and the structural nature of the given input graph.

A modified learning algorithm for an ANN called edge-backpropagation is also used in [35], which considers a new weighting error function. The method tries to overcome the problem of spots, stripes, and ink blobs that are present in real-world documents. The weights used in the computation depend on the gradient of the image so as to give less importance to uniform color regions, like the spots. As a result, the edge-backpropagation ANN parameters are updated by taking into account only the part of the logo which is clearly readable, and the network learning capabilities are not wasted in the reproduction of the noise.

### Multimodal Approaches

A number of authors highlighted that a multilevel approach is beneficial for the recognition of logos; however, it is difficult to a priori train such a system. Therefore, depending on the applications, there are logos better described in global level while others are benefiting by a partial-based description. A hybrid method that uses both approaches in an adaptive way is given by Neumann et al. [57]. In this work, three different approaches for classifying logos are examined. A local method computes statistical and perceptual shape features for each connected component. A global method uses a wavelet decomposition of the horizontal and vertical projections of the image. Finally, a hybrid approach involves adaptively defined weights that specify the relative importance of each method according to an estimate of their relative performance.

In Doermann et al. [23] a hierachal approach is used to prune the searching space in order to match a candidate logo using different features at the different stages of the approach. A combination of text, shape, algebraic, and differential invariants is involved. The algebraic invariants handle cases in which the whole shape of the logo

is given and it is easy to describe. The differential invariants cover complex arbitrary logo shape and handle situations in which only part of the logo is recovered.

A multi-scale strategy is also adopted in [91]. The initial classification is performed on each connected component using the Fisher classifier at a coarse-scale level, regarding the logo region as a gray-scale blob. Each logo candidate region is further classified at successively finer scales by a cascade of simple classifiers.

## Logo Spotting

Another way for detecting and recognizing logos in documents is based on interest point detection and extraction of features. In recent years there has been growing interest in detection and recognition models that use local image features. These features are calculated at particular interest points and are typically characterized by scale and rotation invariance as well as robustness to noise and to changes in illumination. The difference compared to the aforementioned methods is that in approaches based on local image features, the logo model is given and the methods attempt to localize instances of the model (if any) in a document page. As an example, local features combined by a set of spatial coherence rules are used in [65]. A logo model is spotted inside the document image, and in addition the category of the queried document is also determined. The document categorization and the logo detection are performed in a training-free fashion by using a bag-of-words model of visual words. These visual words are described by two types of local features, namely, the SIFT features [50] and the shape context descriptor [11]. Figure 18.7 depicts an example of the feature-matching process between the local features of the query logo and the document image.

Instead of using classical keypoints detectors, connected components can also be used in order to extract key structures. Li et al. [48] proposed such a method where a logo is defined as a group of features with restrictive geometric relationships. For every connected component, a descriptor is calculated based on geometric and statistical properties that are related to its convex hull and are invariant to image scaling and rotation.

## Systems Overview

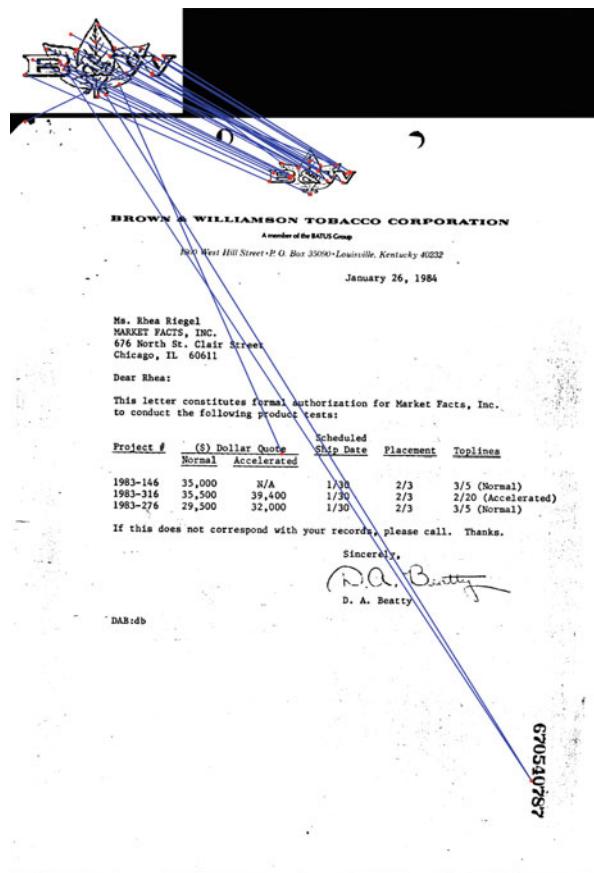
In Table 18.4 a comparative summary is provided that examines selected methods for logo recognition in document images.

## Open Challenges in Logo Recognition in Documents

### Assumptions and Heuristics

It is not uncommon to adopt certain assumptions in order to facilitate the detection and recognition processes and improve the overall performance. For example,

**Fig. 18.7** Feature matching between a logo model (shown in the upper left corner) and the complete document image



prior knowledge is commonly used regarding the logo size and aspect ratio for prescreening. It is also often assumed that the logos are located at certain parts of the document, e.g., on the top one-third of the document, or in a relatively larger distance from other parts of the document. The number of logos in a document image is another issue that is usually addressed by considering only one logo per image. Alternatively, if more than one logo is detected, then only the one with its projection furthest away from the decision threshold is considered. Such implementation heuristics are commonly applied for speeding up the computational execution time. There are also methods where part of the computational cost is transferred to a preprocessing step. For example, before the matching process is initiated, the logo instances undergo several transformations in order to achieve translation, rotation, and scale invariance.

Overall, the large intra-class variations of logos and the diverse quality and degradations in captured document images increase the difficulty of the logo

**Table 18.4** Comparative summary of methods for logo recognition in document images

Name	Year	Logo segmentation required	Features + descriptors	Detection + recognition	Dataset
Doermann et al. [23]	1996	No	Text + contour features, similarity invariants for unrecognized contours	Database pruning by recognized characters and shapes	University of Maryland logo database
Seiden et al. [74]	1997	No	16 features of connected components statistics	Training with logos subset + C4.5 classification	University of Maryland logo database
Diligenti et al. [22]	2001	Yes	Symbolic or real-valued features represented as directed ordered acyclic graphs	Backpropagation through structure	University of Maryland logo database
Neumann et al. [57]	2002	Yes	Local negative symbols + global wavelet decomposition	Ranking by adaptively weighted contributions of local and global methods	University of Maryland logo database
Chen et al. [15]	2003	Yes	Normalized line segments of contours	Similarity of line segments + modified Hausdorff distance	University of Maryland logo database
Gori et al. [35]	2003	Yes	Gradient of image pixel values	Modified edge-backpropagation	University of Maryland logo database
Pham [61]	2003	No	Spatial density of foreground pixels	Maximizing the mountain function	University of Maryland logo database
Zhu and Doermann [91]	2007	No	Context distance, spatial density, aspect ratio, and area	Multi-scale simple classifiers	Tobacco-800 dataset
Rusinol and Lladós [65]	2009	No	SIFT, shape context descriptor	Bag-of-words approach	18 logos + 1,000 real document images
Wang and Chen [81]	2009	No	Candidate logo areas defined by boundary extension	Tree classifier	Tobacco-800 database
Li et al. [48]	2010	No	Geometric and statistical properties of connected components + line profiles	Matching to database features using the anchor line	Tobacco-800 database

detection problem. Therefore, most researchers make certain assumptions in order to have their system work, and the main challenge is to eliminate these assumptions as much as possible.

### **Distortion and Noise**

Logo detection and recognition becomes a demanding issue due to the diverse scanning qualities, distortions, and noise that usually appear in document images [35]. In [15] Chen et al. provide a list of such distortions that includes scaling, rotation, random broken lines, random added strips, occlusions, Gaussian noise, skew, component editing (adding, removing, or skewing the logo component), as well as photo deformations such as pinch, punch, sphere, twirl, and ripple. Besides the aforementioned distortion types that refer to documents in general, there are special degradations that obstruct the document in unpredictable positions, changing the visual appearance of the pictures significantly. The most common are [22] (a) stripe noise, which is created by randomly positioned stripes of varying thickness on the logo; (b) blob noise, which is produced by means of circular isolated blobs with random center and radius; and (c) spot noise, which is obtained by superimposing many circular blobs so as to produce complex shapes that look like ink blots.

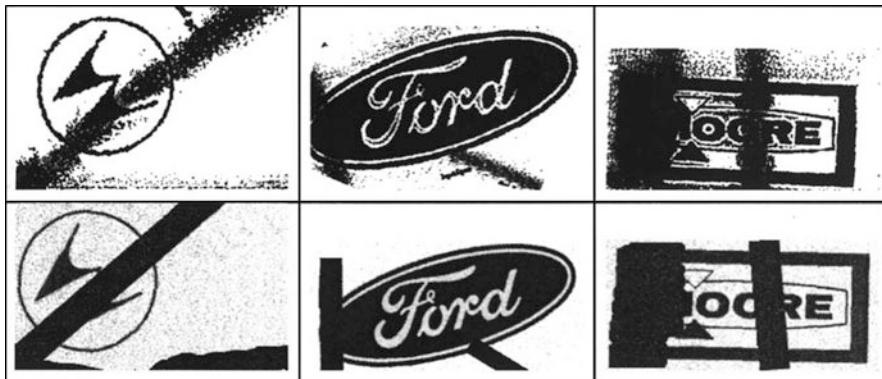
These are quite usual in real-world documents due to the common document processing chain. For example, faxes can show stripes due to transmission errors or dirt in the equipment; ink blobs or lines can be inserted accidentally or willingly to add notes or special signs to the document. The spots can change drastically the structure of a symbol (e.g., by closing a hole or connecting two or more distinct subparts), thus making the recognition with structural or syntactical methods very hard. Figure 18.8 depicts some noisy logo instances on real documents as well as their modeled counterparts.

In general, the content of documents generally includes a mixture of machine-printed text, diagrams, tables, and other elements. An efficient logo detection method must consistently output complete logos while attempting to minimize the false alarm rate. Additionally, the accurate localization needed for logo recognition poses another major challenge.

### **Scalability**

As Tombre and Lamiroy pointed out in [77], the system's scalability is, in general, one of the main concerns in the symbol and graphics recognition domain. Indeed, in recognition schemes that rely on a learning stage and a classification strategy for the recognition of input logos, it is common to observe a decrease in the recognition rate as the number of considered model classes grows.

On the other hand, there are scalability issues regarding the number of documents processed per time unit. The ability to robustly detect logos in large volumes of documents is pivotal for logo recognition. Therefore, testing the detection and recognition methods on large document collections in order to demonstrate achievable generalization performance is an ongoing challenge [23].



**Fig. 18.8** Examples of noise produced by fax machines and Xerox copiers (*top*) and the corresponding images (*bottom*) obtained with noise models (Reproduced from [22])

## Logo Detection and Removal in Images and Videos

This section reviews logo detection in natural images and video streams as well as the special topic of logo detection and removal in TV broadcasts. The background and processes involved in the applications discussed in this section are not fundamentally different from the applications discussed in the previous sections on “Trademark Retrieval Systems” and “Logo Recognition in Document Images.” In particular, logo descriptors, logo recognition, and the principles of logo detection have already been introduced in previous sections. The main difference here is the distinct context of the application, namely, real-life images and videos, which in turn affects the efficiency and effectiveness of previously discussed approaches and guides the selection of the methodology. As a result, the focus of this section is on the particularities that the application context impinges on the methodological choices, and as such it is structured in an application-oriented manner.

First, a technical definition of the problem is given as well as an overview of typical applications and the needs they address. Subsequently, the detection of logos in images and video sequences is discussed and several methods are presented. Special attention is paid in methods regarding sports events which are the most common category of videos in which logos are searched for. This part finishes with a comparison table that summarizes the methodologies presented. The focus is then shifted on TV logo detection and removal where a comparison table is also provided. Finally, a summary of the open challenges in logo detection in images and videos is offered.

### Problem Definition

Contrary to high-quality images of trademarks or scanned document images discussed before, this section focuses on the analysis of video sequences and

real-scene images that might or might not contain logos of interest. As will be appreciated next, although the main blocks of a logo detection system (logo descriptors, classification approaches, etc.) are similar to previously discussed ones, the methodologies developed for different applications such as trademark retrieval and document classification are generally not directly applicable in the domain of real-scene videos and images [8].

Before delving into the technical details, a description of the applications discussed in this section is offered. Similarly to text detection in videos and images (see ▶Chap. 25 (Text Localization and Recognition in Images and Video)), two variants of the problem are identified, depending on whether the logo of interest is part of the scene represented in the image or video sequence or it is a posteriori superimposed on it, a practice typically used with TV sequences.

### Detection of Logos in Images and Videos

The appearance of logos in real scenes (videos or images) can be affected by various factors such as variations in illumination, occlusion, and projection distortions due to their placement in the scene. In addition to factors related to the scene layout, the media itself can introduce certain challenges and limitations, such as low resolution, blurring (especially in videos), and compression artifacts. On the other hand, as a matter of design, logos usually exhibit characteristic shapes and color schemes and are generally positioned so that they are easy to observe even in cluttered background [32].

The objective of a logo detection application is to detect the presence and location of a known logo in an unconstrained scene image or video. The logo to be detected is known to the application *a priori*, provided either as an image query or as a model learned off-line over various known instances of the logo. The number of logo models to be spotted is usually limited and rarely exceeds 20 or 30 models [8]. On the other hand, it is a usual requirement for the processing to take place in close to real time.

Typical applications of logo detection in videos and images are related to trademark use monitoring and advertisement impact assessment services. The objective of the former ones is to establish cases of logo misuse, while the latter focuses on quantifying the visibility achieved during specific events by measuring the time a company's logo was visible in the event media [8,37] (see, e.g., Fig. 18.9). A different branch of applications is built around mobile-based logo detection, where a typical goal is to provide the user with supplementary information about the corresponding company [44].

### TV Logo Detection and Removal

Contrary to scene logos, logos superimposed on the video or images present a different set of challenges. The most emblematic application of superimposed logos is as a form of visible watermark, as they provide a direct assertion of content ownership. Superimposed logos are extensively used in the television broadcasting industry [19, 24, 59]. Almost all television broadcasts feature the television channel's logo in a prominent place, usually one of the video frame corners.



**Fig. 18.9** Logos in sports videos (Reproduced from [37])

Superimposed logos can be opaque, semitransparent, or (partially) animated [82]. An opaquely superimposed logo overlaps a portion of the image content (see Fig. 18.10a), while a semitransparently superimposed logo blends with the host media (see Fig. 18.10b) allowing the original content to remain partially visible.



**Fig. 18.10** TV logo types (a) opaque logo and (b) transparent logo (Reproduced from [89])

A typical challenge in terms of logo detection stems from semitransparent and animated logos which are not normative in terms of length and type of motion.

A frequent practice creating a different set of challenges is the superposition of multiple logos on the same content. This is typical of rebroadcast media, where the logo of the original station can be overlapped by the logo of the broadcasting one. TV logo recognition can also be beneficial for archival purposes in order to navigate in large video archives as well as for commercial detection since TV logos partly or fully disappear during commercials.

The main reason for detecting superimposed logos, apart from establishing the ownership of the media, is to automatically restore the substrate media. In such applications, once the logo is detected, it is removed and replaced by an estimation of the original media calculated from the surrounding visual content. This process is frequently called image inpainting, a term that is borrowed from the arts used to describe a restoration process for damaged paintings [54, 89].

## Detection and Retrieval of Logos in Images and Videos

There are two main categories of algorithms which are widely used for logo detection in images and videos: methods based on color features and methods based on features at local keypoints. The methods based on color features are usually sliding window approaches that involve a classification step that compares the query logo image to subareas of the target image. A variety of color spaces are used in the literature, for example, RGB [14], HSV [60], and CIE-LAB [52], and most of the approaches use histogram representations; therefore, they can be efficiently implemented using integral histogram approaches. On the other hand, local keypoints-based methods use point correspondences for matching object templates involving a voting scheme. They regard an object as a set of local keypoints and fit an affine transformation to simulate their geometric consistency. Both are valid approaches and both have been proven to provide reasonable results under certain conditions.

### **Sliding Window-Based Methods**

Logo detection and localization in color-based approaches is usually performed using a sliding windows framework [14, 38, 52, 60]. In order to reduce computation time, the target image is often subsampled before further processing [38, 52, 60]. For detecting multiple sizes of the input query, multiple scale factors of the input query are also considered. Choosing equally spaced scale factors is one option, but proper size matching is more important at smaller sizes than at larger ones [52]. Regarding similarity measure, histogram intersection is commonly adopted for comparison purposes.

Color is one of the most discriminatory and commonly used features that are frequently employed for logo detection tasks in unconstrained images. For example, Chang and Krumm [14] proposed an object detection algorithm using color co-occurrence histograms (CCH) as an object representation. They quantized multiple object models to a small number of colors using a  $k$ -means clustering algorithm in the RGB color space and then quantized the test images using the same color clusters.

However, color clustering methods are sensitive to changes in illumination and object size which can vary widely between images. The robustness of such approaches can be significantly increased by considering spatial relationships between the colors while allowing for some degree of distortion (see also logo representations combining topological and color features as discussed in the context of trademark retrieval). In this line, Luo and Crandall [52] proposed a method that captures the separation of pairs of color pixels at different spatial distances when these pixels lie in edge neighborhoods, overcoming the problem with the disproportionate energy contributions demonstrated by a single color on the CCH. A pixel is considered an edge pixel if it has different color from any of its eight connected neighbors. This implies that only pixels that are on, or very close to, an edge are included and these tend to be pixels containing the most important spatial-color information. The notion of edge pixels has been further explored by Phan and Androultsos [60] who created an edge map of valid edge points by using vector order statistics that depend on edge gradients. In another approach [38] wavelet decomposition coefficients are introduced and a co-occurrence histogram of both color and wavelet directional detail information is adopted.

An important decision in approaches based on color information is the color space in which color differences are interpreted. The quantization scheme or distance function must be carefully designed to ensure that perceptually similar colors are spatially close and mapped to the same quantized color value. The RGB color space is not perceptually uniform prompting researchers to use more perceptually uniform color spaces and color appearance models. For example, in [52] the CIE LAB color space in conjunction with the ISCC–NBS system is adopted, while in [60] the colors are classified under similar hue orientations in the HSV color space.

Overall, employing color for logo detection in real scenes is a natural choice since color information is a dominant feature in this type of medium. Having said

that, it is important to note that color is not necessarily a good feature for all logos. For certain logos (e.g., the Starbucks or the Coca-Cola ones) the color scheme is a registered feature of the trademark and reproduced faithfully in every instance of the logo. But there is a huge number of logos for which the color scheme is not that important and logos might be reproduced in arbitrary color combinations (e.g., the Nike or the Apple logo). For these logos, purely color-based descriptors might not be the best option.

Color histogram-based methods can be efficiently implemented based on techniques such as integral images or dynamic programming. On the other hand, finding logos in different scales requires the application of the method in several window sizes which results to a very large number of patches that are too expensive to search exhaustively, even in small-sized target images. Furthermore, sliding windows provide poor results when the objects in the target image, are occluded or are subject to heavy deformations.

### Keypoint Correspondence-Based Methods

Keypoint-based local feature methods have also become a common choice regarding logo detection in natural images through keypoints correspondences [44, 46]. They have been successfully used to describe logos and obtain flexible matching techniques that are robust to partial occlusions as well as linear and nonlinear geometric transformations. In these methods a voting scheme is usually involved in order to detect and localize potential logo instances in the target image. Typically, an affine transformation is then calculated that maps the logo onto the target image and patches which accumulate above a certain threshold of votes are retained.

Besides the voting approach, several authors address the problem of properly grouping the matched local descriptors in the target image in order to detect potential logo instances. For instance, Kleban et al. [46] enrolled data mining association rules that capture frequent spatial configurations of quantized local SIFT descriptors. It is an extension of an idea introduced by Quack et al. [62] where association rules are employed to select features for object detection. The resulting different types of rules are appropriately weighted, and logo localization is performed by grouping selected features using mutual rule matches with representative training examples.

Spectral saliency analysis is another tool to localize logo instances by detecting the local high-frequency regions of an image. It is based on the hypothesis that a natural image consists of two parts, the redundant part and the novel part. However, it may result to unwanted parts with similar local frequency characteristics. To overcome this, Gao et al. [32] propose a partial spatial context descriptor to formulate the spatial distribution for the set of matching SURF keypoints [9]. The descriptor is based on a bag-of-words formalism and describes both the transformation consistency and contextual information of the point set. In a different line, Meng et al. [55] detect logo instances by finding subimages where the largest point-wise mutual information towards the query is measured. A branch-and-bound

search ranks the subimages by the mutual information, and a relevance feedback scheme further improves the results by verifying relevant and irrelevant subimages.

A common way to calculate the geometric consistency of a potential logo instance is by estimating an affine transformation model between the query and the point set. For this purpose the RANSAC algorithm [31] is often employed in order to model the transformation, resulting to a number of inliers, that is, matches that fit the model. However, using a fixed threshold for the number of inliers as a spatial verification criterion has many drawbacks since it depends on many factors like the average number of keypoints in the query and the image, the redundancy of keypoints, and the size of the query. To overcome this, Joly and Buisson [44] proposed an “*a contrario*” adaptive thresholding approach, adapted to geometric consistency scores. A global geometric consistency is achieved by applying a threshold in normalized scores where the scoring of matches is directly related to the statistical dependence between the spatial positions of the query and the matched keypoints.

In general, methods based on keypoints are usually faster than the ones based on color. They are also adequate in cases where robustness to occlusions and clutter is required. A limitation of these methods is that in a typical application, thousands of nonrelevant keypoints are produced in the target image, where nonrelevant refers to keypoints not belonging to any logo region. This leads to increased storage requirements, while despite the large number of candidate keypoints, the annotated logos may still contain only 0, 1, or 2 points, which is insufficient for recognition.

Local descriptors are based on intensity values ignoring the color information which is a prominent logo feature in many cases. To increase illumination invariance and discriminative power, several color descriptors have been proposed in the literature. In a recent study, Van de Sande et al. [79] provide a thorough discussion on the invariance properties and the distinctiveness of color descriptors. In a logo detection perspective, they can be seen as an attempt to bridge the gap between color-and intensity-based methodologies.

### **Logo Detection in Videos**

Similar to still images, color information is a key feature frequently used in the literature regarding logo detection in video sequences [4, 20, 37]. Invariance under color distortions is an important task for these methods due to the constantly changing color information in the context of video streams. For instance, in [37] the effects of illumination intensity (e.g., due to clouds or other environmental conditions) are reduced by using only the chrominance components of the luminance–chrominance space. Alternatively, the photometric invariant color space [33] has been also opted for since it provides robustness to changes of surface orientation, illumination direction, and intensity [4].

The local color distribution can also be used as an indicator of areas of interest in video frames. For example, Hollander and Hanjalic [20] detect potential logo areas by searching for homogeneously colored regions that are surrounded by large color differences. The image area is represented by intensity profiles along lines, and logo recognition is performed by matching these lines with the line profiles of

the query logo models. Low-cost color processing is also involved in [37] to detect candidate logo regions that are then recognized using scale-normalized Gaussian receptive fields computed over a limited region of interest. In this manner, a model of the logo's visual appearance can be created based on a small number of sample query images. The identification of the logo instances can be based either on the distance between histograms or on probabilistic measures.

Logo detection can be further assisted by closely related techniques that focus on the detection of broader image regions that may enclose the logo under consideration. For example, billboards in sports broadcasting streams are located on the side of the field and are usually the place where brand logos are placed. Therefore, knowing the exact position of billboards significantly narrows down the spatial search space for logo detection. On the other hand, developing a billboard analysis and detection system faces several challenges on its own right [4]. Watve and Sural [83] proposed a method where frames within video shots are segmented to locate possible regions of interests in a frame where billboards are potentially present. A combination of local and global features is also employed for detecting individual billboards by matching them with a set of given templates.

Local keypoints associated with local descriptors are the basis for several methods regarding logo detection on videos. The logo queries are matched against the content extracted from every frame of the video to compute a "match score" indicating the likelihood that a particular logo occurs at any given point in the video [8]. It is evident that the requirement for near real-time response poses a computational burden to the detection of logos in video sequences in a frame-by-frame matching basis. Depending on the application, a common approach is to separate the detection process into an off-line preprocessing step and an online querying step. The benefit of this approach is that matches are effectively precomputed so that at run-time frames and shots containing any particular object can be retrieved with no delay. An example of such a system is given by Sivic and Zisserman [75] where an object retrieval approach is presented which incorporates text retrieval techniques in order to search and localize all the occurrences of an object in a video. The query is provided by a user-specified subpart of an image, and a bag-of-words model is applied in the visual domain by generating a codebook of affine covariant features, represented as SIFT descriptors. In terms of computational efficiency, their method demonstrates two orders of magnitude speedup without significant loss in performance when compared to the standard frame-to-frame matching. More interestingly, the system could efficiently search for instances of an "unknown" logo that was not part of the video stream, that is, its descriptors have not been pre-calculated during the off-line step.

Ensuring the temporal continuity of logo instances or candidate regions within a video shot facilitates the detection process. In this line, in [75] the regions detected in each frame of the video are tracked using a simple constant velocity dynamical model and correlation. Any region which does not survive for more than three frames is rejected. On the other hand, a dedicated tracking module is involved in [37] that uses Kalman filtering. It is based on the estimated position and size of the current logo in order to predict a region of interest in the next frame.

Concluding, it should be noticed that not all of the techniques presented so far are fully automated but instead some of them rely somehow on human intervention. For example, in [8] the time intervals of the video likely to contain the logos are used to drive a user interface through which a human annotator is involved in order to validate the automatic annotation results. Similarly, in [37] a supervisor coordinates the different modules, keeps track of targets that have been identified, and halts the tracking of a target region when identification fails. Table 18.5 provides a comparative summary that summarizes the aforementioned methods for logo detection in images and videos.

## TV Logo Detection and Removal

This section focuses on TV logo detection in broadcast videos which poses a different problem than the one addressed in the last section. Several methods are presented that tackle the problem of finding superimposed logos that may be opaque, transparent, or even animated usually surrounded by constantly changing background content. The task of removing the TV logo is also discussed focusing on the smooth replacement of the removed logo by properly reconstructed background content.

### Logo Detection

TV Logo detection is meant to locate and extract a logo within a sequence of images. Broadcasters' logos are typically over imposed inside one of the four corners of the video frame [3, 59]. Taking this fact into consideration, a significant acceleration of the overall processing time can be achieved, which is essential for real-time implementation. Thus, it can be assumed that the probability of the logos appearing in the four corners of the video frame is higher than the probability of its appearing in the center [89]. Alternatively, prior information can be involved by building a table that contains information regarding in which of the four corners of the video frame the logo under consideration may appear [69]. A general approach that is applied to detect both opaque and semitransparent logos is to find areas in the image that provide some kind of spatiotemporal persistency in terms of properly selected features. For example, in [59] time-averaged edge fields are employed based on classical edge detection methods while preserving the persistency of the extracted edges over a number of frames. In a similar way, Albiol et al. [3] seek for areas with stable contours and they use time-averaged gradients in order to detect them.

A different variety of methods employ pixel-wise approaches which are based on the observation that image areas where logo is overlaid show luminance variance values in a narrower interval than the rest of the image areas, depending on the logo transparency [54, 69, 89]. For example, in opaque logos it is assumed that the video content changes over time except for the logo area and any differences between subsequent frames at the logo position occur due to noise. In general, the objective of the temporal segmentation is to find minimal luminance variance regions in every

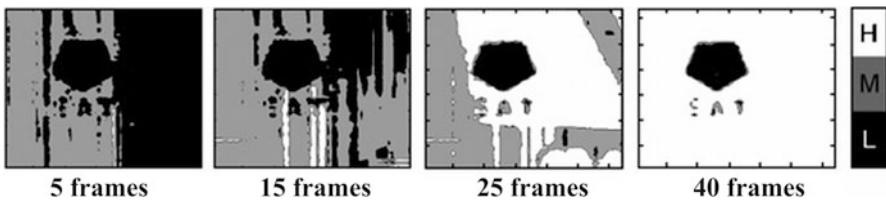
**Table 18.5** Comparative summary of image and video logo detection and retrieval methods

Name	Year	Features + descriptors	Detection + retrieval	Medium	Dataset
Hollander and Handjalic [20]	2003	Affine invariant binary color profiles	String matching	Video	5 logos in 5 sequences
Sivic and Zisserman [75]	2003	Vector-quantized SIFT descriptors	Bag-of-features matching + tf-idf weighted frequency vectors + spatial consistency voting	Video	6 objects in 2 full-length films
Hall et al. [37]	2004	Multidimensional color histograms	Scale-normalized receptive fields + Kalman-filtering tracking	Video	3 logos in 2 formula one videos
Luo and Crandall [52]	2006	Color edge co-occurrence histogram	Spatial-color joint probability functions + normalized cross correlation	Image	Several images from the Web
Bagdanov et al. [8]	2007	SIFT descriptors	Bag-of-features matching + normalized match score	Video	6 logos in 3 sports videos
Hesson and Androultsos [38]	2008	Wavelet decomposition coefficients + color	Sliding non-overlapping windows of varying size in multiple scales + histogram intersection	Image	2 query logos + 3,000 test images
Kleban et al. [46]	2008	Keypoints by Harris affine detector + SIFT descriptors	Spatial pyramid mining of association rules + DBSCAN density-based clustering	Image	7 query logos in 974 images from the Web
Phan and Androultsos [10]	2009	Color edge cooccurrence histogram + vector order statistics + color quantization method based on HSV color space	Sliding non-overlapping windows in multiple scales + histogram intersection	Image	400 logo images + 5,000 unrelated images.
Gao et al. [32]	2009	Spectral + spatial saliency analysis, SURF features	Maximum saliency density, similarity based on partial spatial context	Image	10 query logos + 10,016 test images from the Web

(continued)

**Table 18.5** (continued)

Name	Year	Features + descriptors	Detection + retrieval	Medium	Dataset
Joly and Buisson [44]	2009	SIFT descriptors	Multi-Probe Locality Sensitive Hashing + RANSAC on affine transformation model + “a contrario” query expansion	Image	Oxford Building dataset + BelgaLogos
Meng et al. [55]	2010	Keypoints by Harris affine detector + SIFT descriptors	Branch and bound search + maximum mutual information + relevance feedback	Image	BelgaLogos



**Fig. 18.11** Evolution of the high ( $H$ ), medium ( $M$ ), and low ( $L$ ) luminance variance areas of the LVI as the number of processed frames grows (Reproduced from [19])

frame. In this line, Cozar et al. [19] describe a method that involves the luminance variance image (LVI). The LVI is an image with the same size as the video frames in which a pixel is represented by the difference between the maximum and minimum luminance values in the corresponding pixel location along the sequence of frames. The minimal luminance variance regions can be extracted by applying a properly defined threshold. Figure 18.11 depicts an example where high, medium, and low luminance variance areas of the LVI are shown as the number of processed frames grows. Clearly, such approaches are affected by the set of frames used for variance calculation and the value of the threshold.

Instead of using the logo as one entity, Yan et al. [89] introduced the notion of logolets, which are small image regions corresponding to logo parts. A Bayesian classifier framework is then used in combination with an ANN trained to detect the logolets in video frames. Duffner and Garcia [24] explored further the use of ANN by proposing a system for transparent logo detection where the raw input image is treated as a whole without any assumptions about the features to extract or the areas to be analyzed. For this purpose, a multi-scale convolutional ANN architecture is utilized that derives problem-specific feature extractors from a large training set of logo and non-logo patterns.

In case of animated logos, the approach to be followed depends significantly on the percentage of the logo area that is changing through time. For instance,

partially animated logos can be regarded as opaque ones, because they can be detected through their immovable parts. However, in the general case, a more delicate treatment is required since the temporal persistency has to be preserved not just for a single frame instance but for a set of consecutive frames that form the complete logo animation circle. In this line, Wang et al. [82] use gradient information as low-level visual features and propose a template-matching approach that seeks persistent gradients over multiple video frames. In [29] a method tailored specifically for animated logos is proposed that utilizes a multi-frame contour representation which is matched by the contours of a test video using a voting-based decision scheme.

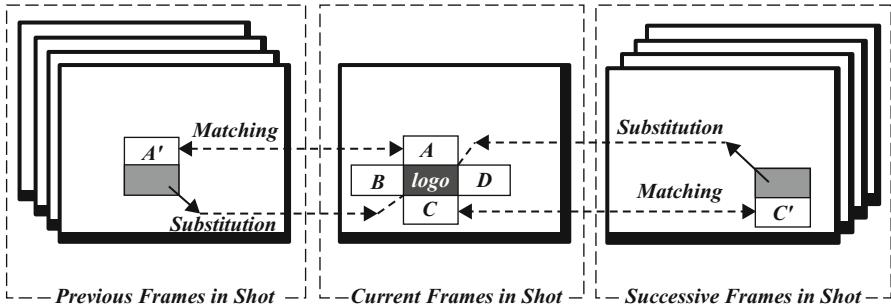
Typically, the result of the detection methods is a binary mask that indicates the region where the TV logo is placed. The images that denote the difference between subsequent frames are binarized by thresholding in order to obtain initial logo masks, and the final logo mask is obtained, for example, by contour relaxation based on Markov Random Fields [54]. It is also common to further refine the logo mask by morphological processing in order to reduce spurious pixels and fill holes [59]. Additional constraints may also be applied regarding the shape or the size of the mask in order to eliminate improbable results [3].

Overall, TV logo detection is a process considering both spatial and temporal domains. Introducing a spatial persistency requirement benefits the detection process. This requirement is derived from the observation that no change occurs over time for the TV logo position and that the logo appears in one of the four corners of the video frame (or, more rarely, interchangeable in different corners depending on the video stream content, e.g., commercials). On the other hand, the persistency of the logo over time is closely related to the number of subsequent frames under consideration. A low number of frames is not sufficient to eliminate noise from accidentally stable background content that could increase false alarms. On the contrary, too many frames tend to blur a logo's inherent features that could increase missed segments [82].

## Logo Removal

The efficient detection of the logo mask is an important prerequisite for automatic TV logo removal in order to improve the viewing experience of rebroadcast programs or to replace the existing logo with another one [54, 82]. An intuitive approach is to find the best replacement patch for the removed region within the video shot. Indeed, if the motion of the logo region is sufficient enough, then the region underneath the logo may be exposed in a nearby frame. In this case, matching has to be based on a properly selected measure that demonstrates low sensitivity to small perturbations of the image while being tolerant in small position errors. Yan et al. [89] follow this approach in order to remove small logos by adopting the Hausdorff distance. However, if the logo region is too large, the overlapping will result in observable edges for the video region. In these cases a video inpainting approach is usually opted for.

Video inpainting can be considered as the approximation of the covered region by using the surrounding information and their differences. However, two main issues arise with this technique, namely, the size of the logo area and how to



**Fig. 18.12** Matching-based algorithm for region overlapping (Reproduced from [89])

maintain coherence between adjacent video frames that is distorted by unwanted edges appearing due to the overlapping processing. In [89] an extension of the 2D gradients in the image inpainting technique is proposed which uses 3D gradients exploiting the temporal correlations in video. Figure 18.12 shows an example where in order to obtain the substitution region of the logo, the regions in the previous and next frame are considered, with the most similar pair among all the candidate pairs being selected. The corresponding region is then used to replace the logo region.

Video inpainting is also used by Wang et al. [82] where a multivalued image regularization with Partial Differential Equations (PDE) is proposed. PDE-based regularization may be seen as the local smoothing of an image along defined directions depending themselves on the local configuration of the pixel intensities [78]. The target is to smooth the image while preserving its edges (discontinuities in image intensities), i.e., perform a local smoothing mostly along directions of the edges, avoiding smoothing orthogonally to these edges. In [82] PDE is employed for inpainting the logo regions while preserving the local geometry of the multivalued image discontinuities. The inpainting of the logo region with the neighborhood area requires a temporal set of frames in order to compute a structure tensor in the spatiotemporal domain.

TV logo removal can also be treated as an extrapolation problem. In this case the image area surrounding the logo is extrapolated using a frequency-selective method and used to replace the logo. Linear combinations of weighted basis functions can be used as parametric models in order to approximate the support area of a logo. In this context, spectral estimation methods are involved in order to describe the logo area by a few dominant features while the logo is given by extrapolation. For example, Meisinger et al. [54] use two-dimensional Discrete Fourier Transform basis functions since they are especially suited in order to conceal monotone areas, edges, and noise-like regions. In case of large logos, the logo is partitioned into blocks and the different blocks are processed subsequently. For inpainting the logo, first the corner blocks and then the inner blocks are inpainted, in order to exploit as much surrounding data as possible.

In general, video inpainting is a powerful tool that delivers visually pleasant results. However, the current methods still encounter difficulties, for example, when dynamic-texture backgrounds are under consideration or in case where occluded objects are involved in the video scene. In such cases the spatiotemporal inconsistency leads to visually annoying artifacts and further refinements are required. For example, in [89] graph cut textures are used to find a similar block and blend the visible edges that arise due to overlapping.

A summary of the key aspects of the above methods regarding TV logo detection and removal methods is provided in Table 18.6.

## Open Challenges in Logo Detection and Removal in Images and Videos

### Generalization

Searching for logos in databases of unconstrained color images is a time-consuming and labor-intensive task since the applicability of any method is subject to many uncontrollable factors such as lighting, deformation, and occlusion. Video streams are further characterized by perspective deformation due to pan, tilt, and zooming operations of the camera, motion blur, as well as often rapid changes in the illumination conditions. Several methods address these issues by requiring a large number of sample queries acquired under various conditions, while others are based on models that aim to predict a large number of potentially problematic cases. However, there is still a lack of approaches that will expose robustness and generalization for logo detection in a variety of conditions.

Similarly, in case of videos the detection efficiency is heavily related to the variance that characterizes the context in video sequences as well as on the representativeness of the query logo model. It has been reported that synthetic query logos perform worse in terms of precision and recall than logo instances cropped directly from the video [8]. This is due to the fact that many other logos consisting of mostly text and graphics are confused for the synthetic logo models. Using more than one query logo model is a solution but again the sample logos should cover reliably the variations during the video. In [37] it is argued that 6–8 instances are sufficient, but in any case the color model for each logo must be acquired from images under actual illumination conditions to reduce the effects of illumination changes.

### Features Applicability

While color-based approaches have demonstrated reliable results in logo detection tasks, it is evident that raw color information might be quite instable due to illumination and shadows. There are methodologies for illumination estimation and shadow removal, but of course including them in the pipeline would imply a considerable time cost. On the other hand, even if some kind of color normalization is assumed, then there are still many alternative human-oriented descriptions for color features that yield better results (see, e.g., the color naming discussion in

**Table 18.6** Comparative summary of TV logo detection and removal methods

Name	Year	Detection	Localization + tracking	Logo removal	Logo type	Dataset
Albiol et al. [3]	2004	Stable contours	Temporal gradient analysis + morphological operations	N/A	Opaque, transparent	6 TV channels
Yan et al. [89]	2005	Frame differentiation + logolets	Bayesian classifier framework combination with an ANN	Overlapping matching + video inpainting by 3D gradients + graph cut textures	Opaque	Video clips collected from various Web sites
Meisinger et al. [54]	2005	Frame differentiation	Binarization through thresholding + contour relaxation based on Markov Random Fields	Extrapolation of surrounding image by frequency-selective method	Opaque, semitransparent	TV streams
Wang et al. [82]	2006	Multispectral gradient information	Generalized interframe gradient + adaptive thresholding	PDE-based regularization	Opaque, transparent, animated	4 h from TRECVID'05 + several TV channels
Duffer and Garcia [24]	2006	Train ANN with positive and negative examples	Subsampling in pyramid of images + localization by proximity in image and scale space	N/A	Transparent	800/257 images from TV with/without logo

Santos and Kim [69]	2006	Edge detection by magnitude of gradient + time averaging	Cross correlation + hypothesis testing	N/A	Opaque, semitransparent, partially animated	24 h from 10 channels
Cozar et al. [19]	2007	Minimal luminance variance region	Spatiotemporal segmentation + nonlinear diffusion filters	N/A	Opaque, semitransparent	6 h of 45 broadcasted video sequences from different TV channels
Esen et al. [29]	2008	Multiframe contour representation	Compare to sample video by Generalized Hough transform + majority voting	N/A	Animated	27 h of 3 channels
Ozay and Sankur [59]	2009	Canny edge detection + time-averaged edges + SVM for best feature selection	Binarization hysteresis threshold + morphological post processing	N/A	Opaque, transparent	240 video samples from 12 TV channels

the section on “Trademark Retrieval Systems”). This would make sense as logos comprise a limited number of named colors.

In the case of local keypoints, spatial consistency is another important aspect of the matching process since it helps filtering out false matches. It can be measured either by simply requiring that neighboring matches in the query lie in a surrounding area in the video frame or more strictly by requiring the same spatial layout in the neighboring matches of both the query image and the video frame [75]. Another problem is posed by logo models that give rise to a relatively low number of feature points, causing the matching process to become unreliable.

Finally, despite that several robust features have been extensively used (e.g., color, shape, local keypoints), contextual information is rarely explored although it may provide a robust estimate of the probability regarding the logo’s presence, position, and scale [58]. Indeed, global scene representations based on aggregated statistics of low-level features may act as sources of contextual information that can improve the detection efficiency while providing tolerance to image degradation and illumination changes.

### **Scalability**

Logo detection in large image databases is a time-consuming process, with the unconstrained nature of real-scene images (as opposed, e.g., to the document images discussed before) creating a demand for substantially more complex analysis methodologies.

The computational load of processing a typical video stream, comprising tens of thousands of frames, is heavy, even if approaches such as frame sampling are applied. As a result, real-time requirements are hardly met by state of the art methods. An even more important observation though is that the majority of existing logo detection methodologies would be difficult to scale gracefully with the number of frames processed or the number of logo models queried. This is particularly true for methodologies relying on indexing of large numbers of local features and analyzing correspondences between them. Having said that, there is a lot of promising research in the field and a lot of investment on big-data processing, thus there are expectations that new breakthroughs will be achieved in the near future.

---

### **Conclusion**

This chapter discussed various applications related to the detection, localization, and recognition of logos and trademarks in different media. The chapter attempted to cover the considerable breadth and depth of the topic at hand by using three identified applications as the guide for the discussion.

Starting with the application of trademark retrieval systems, typical trademark representations and descriptors were introduced, while aspects related to perceptual organization of graphical entities were discussed as well as typical practices in indexing and retrieval. Subsequently, in the context of logo recognition in document

---

images, the use of logos for classification was examined and the application of local keypoints-based descriptors and logo spotting was introduced. Finally, the application of logo detection and removal in images and videos was studied, focusing on the challenges that real-scene media present as well as the typical application of logo removal and restoration of the original substrate in TV broadcasts. In all cases, summary tables of published work were provided, linking to the different aspects of the problem discussed in the text.

As the range of applications examined is extensive and diverse, it is difficult, if not meaningless, to compile a short list of generic open challenges, inviting instead the interested reader to review the respective “open challenges” sections for each application discussed in this chapter. Nevertheless, it is worth noting that one open problem repeatedly encountered in different applications is the issue of scalability of the proposed methodologies, so much in terms of the number of trademark models treated as in terms of the processing time required.

In summary, logo and trademark recognition is a multifaceted area of research, touching upon different domains and covering a variety of distinct real-world applications. As such, it is an interesting topic for multidisciplinary research and an attractive evaluation setting for a wide range of disciplines and topics including document analysis, indexing and retrieval, perceptual organization, and multimodal data fusion to mention just a few.

---

## Description of Reference Datasets

A good source for datasets and tools are the Web sites of IAPR TC10 (“Graphics Recognition”) and TC11 (“Reading Systems”)<sup>6</sup> that host or link to the latest available information. The information given below was correct at the time of writing, but the readers are encouraged to look at the above sites for most up-to-date information. See also ►Chaps. 29 (Datasets and Annotations for Document Analysis and Recognition) and ►30 (Tools and Metrics for Document Analysis Systems Evaluation) of this book for more generic tools and datasets.

## Trademark Retrieval Systems

Public datasets of trademarks with associated ground truth information are difficult to encounter, although individual trademarks are searchable and downloadable one by one through the Web sites of trademark registration offices. The interested readers can check, for example, the CTM-ONLINE service of the OHIM,<sup>7</sup> the “Localizador de Marcas” of the Spanish Patent and Trademark Office,<sup>8</sup> or the “Trademark Electronic Search System” of the US Patent and Trademark Office.<sup>9</sup>

In terms of collections of trademarks, the most used trademarks dataset is the “MPEG-7 Still Images Content Set,” Item S8, which comprises about 3,000

B&W geometric trademark images captured by a scanner, provided by the Korean Industrial Property Office more than a decade ago [12]. The dataset is not publicly available, and according to its licensing agreement, its use is strictly for noncommercial purposes.

An alternative collection of trademarks and logos provided by the IBM Almaden Research Center is also available at the time of writing<sup>10</sup> [40, 41], although the licensing particulars do not seem to be clear.

## Logo-Based Document Classification

The evaluation in most of the methods regarding document classification based on logos is performed using the Tobacco-800 dataset [76] and the University of Maryland logo database [49]. Tobacco-800 is a public subset of the IIT CDIP Test Collection based on 42 million pages of documents (in 7 million multipage TIFF images) obtained from UCSF and released by tobacco companies under the Master Settlement Agreement. The documents were collected and scanned using a wide variety of equipment over time providing therefore a comprehensive dataset characterized by large variability. The University of Maryland logo database is another commonly used dataset that contains 106 distinct logo images in TIFF format. The logos in the dataset have a variety of sizes ranging from  $134 \times 116$  pixels up to  $629 \times 671$  pixels. Some examples of both datasets are shown in Fig. 18.13.

## Logo Detection and Removal in Images and Videos

BelgaLogos [44] is a natural image collection specifically created for logo detection and retrieval evaluation. The dataset is freely available for research purpose only. It considers 26 different logos and consists of 10,000 images covering diverse categories of objects and events. A given image can contain one or several logos or no logo at all. On the other hand, the World Wide Web is another popular source for unconstrained images; therefore, several authors tend to compile their own private datasets which typically consist of several hundred images wherein a small number of logos is searched for.

In the case of videos, logos are usually part of billboards, banners, and other physical advertising media. Most commonly, such advertisements appear in sports broadcasting like soccer and football fields, formula one race circuits, and tennis courts. Therefore, the majority of methods in the literature use sports videos as evaluation datasets. Similarly, for TV logo detection and removal evaluation, a diversity of broadcast video and TV channels are used. However, there is still a lack of any standardized datasets, while it is clear that such data associated with ground truth data would be a valuable aid for evaluation purposes.



**Fig. 18.13** Example logos from (a) Tobacco-800 dataset and (b) University of Maryland logo database

---

## Cross-References

- ▶ [An Overview of Symbol Recognition](#)
  - ▶ [Datasets and Annotations for Document Analysis and Recognition](#)
  - ▶ [Graphics Recognition Techniques](#)
  - ▶ [Image Based Retrieval and Keyword Spotting in Documents](#)
  - ▶ [Page Similarity and Classification](#)
  - ▶ [Text Localization and Recognition in Images and Video](#)
  - ▶ [Tools and Metrics for Document Analysis Systems Evaluation](#)
- 

## Notes

<sup>1</sup> United States Patent and Trademark Office, “Performance and Accountability Report Fiscal Year 2011.”

<sup>2</sup> Source “OHMI Manual of Trade Mark Practice” Part 2, Chapter 2C, OHMI, accessible online: <http://oami.europa.eu/ows/rw/pages/CTM/legalReferences/guidelines/OHIMManual.en.do>

<sup>3</sup> Note that there are also “color marks” per se, referring to the situation that a particular color without any specified contours is registered as a trademark. This type of trademarks is out of the context of this chapter.

<sup>4</sup> OHMI, “Manual of Trade Mark Practice,” online resource, accessed on May 2012, Source: <http://oami.europa.eu/ows/rw/pages/CTM/legalReferences/guidelines/OHIMManual.en.do>

<sup>5</sup> Source: “Implementing a Digital Mailroom,” White Paper, Datafinity Ltd, June 2009, UK.

<sup>6</sup> <http://www.iapr-tc10.org/> and <http://www.iapr-tc11.org/> respectively.

<sup>7</sup> <http://oami.europa.eu/ows/rw/pages/QPLUS/databases/searchCTM.en.do> (last accessed on Sept. 2012).

<sup>8</sup> <http://sitadex.oepm.es/Localizador/homeLocalizador.jsp> (last accessed on Sept. 2012).

<sup>9</sup> <http://www.uspto.gov/trademarks/index.jsp> (last accessed on Sept. 2012).

<sup>10</sup> <http://www.eurecom.fr/~huet/work.html> (last accessed on Sept. 2012).

---

## References

1. Alajlan N (2007) Retrieval of hand-sketched envelopes in logo images. *Lect Notes Comput Sci* 4633/2007:436–446
2. Alajlan N, Kamel MS, Freeman G (2006) Multi-object image retrieval based on shape and topology. *Signal Process Image Commun* 21(10):904–918
3. Albiol A, Fulla MJ, Albiol A, Torres L (2004) Detection of TV commercials. In: Proceedings of the international conference on acoustics, speech and signal processing, Montreal, pp 541–544
4. Aldershoff F, Gevers T (2004) Visual tracking and localisation of billboards in streamed soccer matches. *SPIE Electron Imaging* 5307:408–416
5. Alwis S, Austin J (1999) Trademark image retrieval using multiple features. In: Proceedings of the challenge of image retrieval (CIR-99), BCS electronic workshops in computing, Newcastle-upon-Tyne
6. Amir A, Lindenbaum M (1998) A generic grouping algorithm and its quantitative analysis. *IEEE Trans Pattern Anal Mach Intell* 20(2):168–185
7. Baeza-Yates R, Ribeiro-Neta B (2011) Modern information retrieval: the concepts and technology behind search, 2nd edn. Addison-Wesley, New York
8. Bagdanov AD, Ballan L, Bertini M, Bimbo AD (2007) Trademark matching and retrieval in sports video databases, in James Ze Wang; Nozha Boujemaa; Alberto Del Bimbo & Jia Li, ed., *Multimedia Information Retrieval*, ACM, New York, pp 79–86
9. Bay H, Tuytelaars T, Van Gool L (2006) Surf: speeded up robust features. In: *ECCV*, Graz
10. Belkin NJ, Kantor P, Fox EA, Shaw JA (1995) Combining evidence of multiple query representations for information retrieval. *Inf Process Manag* 31(3):431–448
11. Belongie S, Malik J, Puzicha J (2002) Shape matching and object recognition using shape contexts. *IEEE Trans Pattern Anal Mach Intell* 24(24):509–522
12. Bober M, Preteux F, Kim Y-M (2001) MPEG-7 visual shape descriptors. Technical report, Ref: VIL01-D112, Mitsubishi Electric
13. Chan DY-M, King I (1999) Genetic algorithm for weights assignment in dissimilarity for trademark retrieval. In: Huijsmans DP, Smeulders AWM (eds) *VISUAL’99*, Amsterdam. LNCS 1614, pp 557–565

14. Chang P, Krumm J (1999) Object recognition with color cooccurrence histograms. In: Proceedings of the IEEE conference on computer vision and pattern recognition, Fort Collins, pp 498–504
15. Chen J, Leung MK, Gao Y (2003) Noisy logo recognition using line segment Hausdorff distance. *Pattern Recognit* 36(4):943–955
16. Chen J, Wang L, Chen D (2011) Logo recognition: theory and practice. CRC, Boca Raton
17. Chou T-C, Cheng S-C (2006) Design and implementation of a semantic image classification and retrieval of organizational memory information systems using analytical hierarchy process. *Omega* 34:125–134. Elsevier
18. Ciocca G, Schettini R (2001) Content-based similarity retrieval of trademarks using relevance feedback. *Pattern Recognit* 34:1639–1655
19. Cózar JR, Guil N, González-Linares JM, Zapata EL, Izquierdo E (2007) Logotype detection to support semantic-based video annotation. *Signal Process Image Commun* 22(7–8):669–679
20. den Hollander RJM, Hanjalic A (2003) Logo recognition in video stills by string matching. In: Proceedings of IEEE international conference on image processing (ICIP), Barcelona, pp 517–520
21. Desolneux A, Moisan L, Morel J-M (2008) From Gestalt theory to image analysis: a probabilistic approach. Springer, New York
22. Diligenti M, Gori M, Maggini M, Martinelli E (2001) Adaptive graphical pattern recognition for the classification of company logos. *Pattern Recognit* 34:2049–2061
23. Doermann D, Rivlin E, Weiss I (1996) Applying algebraic and differential invariants for logo recognition. *Mach Vis Appl* 9(2):73–86
24. Duffner S, Garcia C (2006) A neural scheme for robust detection of transparent logos in TV programs. Lecture notes in computer science—II, vol 4132. Springer, Berlin, pp 14–23
25. Eakins JP, Shields K, Boardman JM (1996) Artisan—a shape retrieval system based on boundary family indexing. In: Sethi IK, Jain RC (eds) Storage and retrieval for image and video databases IV (Proc SPIE 2670). SPIE, Bellingham, pp 17–28
26. Eakins JP, Boardman JM, Graham ME (1998) Similarity retrieval of trademark images. *IEEE Multimed* 5(2):53–63
27. Eakins JP, Riley KJ, Edwards JD (2003) Shape feature matching for trademark image retrieval. In: Bakker EM et al (eds) CIVR 2003, Urbana-Champaign. LNCS 2728, pp 28–38
28. El Badawy O, Kamel M (2002) Shape-based image retrieval applied to trademark images. *Int J Image Graph* 2(3):375–393. World Scientific
29. Esen E, Soysal M, Ates TK, Saracoglu A, Aydin Alatan A (2008) A fast method for animated TV logo detection. In: CBMI, London, pp 236–241, June 2008
30. Fall CJ, Giraud-Carrier C (2005) Searching trademark databases for verbal similarities. *World Pat Inf* 27:135–143
31. Fischler M, Bolles R (1981) Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. *CACM* 24(6):381–395
32. Gao K, Lin S, Zhang Y, Tang S, Zhang D (2009) Logo detection based on spatial-spectral saliency and partial spatial context. In: Proceedings of the ICME, New York, pp 322–329
33. Gevers T, Stokman H (2004) Robust histogram construction from color invariants for object recognition. *Trans Pattern Anal Mach Intell* 24:113–118
34. Giacinto G, Roli F (2005) Instance-based relevance feedback for image retrieval. In: Saul LK, Weiss Y, Bottou L (eds) Advances in neural information processing systems, vol 17. MIT Press, Cambridge, MA, pp 489–496
35. Gori M, Maggini M, Marinai S, Sheng J, Soda G (2003) Edge-Backpropagation for noisy logo recognition. *Pattern Recognit* 36(1):103–110
36. Grossman DA, Frieder O (2004) Information retrieval: algorithms and heuristics, 2nd edn. Springer, Dordrecht
37. Hall D, Pelisson F, Riff O, Crowley JL (2004) Brand identification using Gaussian derivative histograms. *Mach Vis Appl* 16(1):41–46
38. Hesson A, Androustos D (2008) Logo and trademark detection in images using color wavelet co-occurrence histograms. In: Proceedings of the IEEE international conference on acoustics, speech and signal processing, Las Vegas, pp 1233–1236

39. Hsieh I-S, Fan K-C (2001) Multiple classifiers for color flag and trademark image retrieval. *IEEE Trans Image Process* 10(6):950
40. Huet B, Hancock ER (1999) Line pattern retrieval using relational histograms. *IEEE Trans Pattern Anal Mach Intell* 21(12):1363–1370
41. Huet B, Hancock ER (2002) Relational object recognition from large structural libraries. *Pattern Recognit* 35(9):1895–1915
42. Hung M-H, Hsieh C-H, Kuo C-M (2006) Similarity retrieval of shape images based on database classification. *J Vis Commun Image Represent* 17:970–985
43. Jain AK, Vailaya A (1998) Shape-based retrieval: a case study with trademark image databases. *Pattern Recognit* 31(9):1369–1390
44. Joly A, Buisson O (2009) Logo retrieval with a contrario visual query expansion. In: Proceedings of the 17th ACM international conference on multimedia (MM '09), Beijing, pp 581–584
45. Kim YS, Kim WY (1998) Content-based trademark retrieval system using a visually salient feature. *Image Vis Comput* 16:931–939
46. Kleban J, Xie X, Ma W-Y (2008) Spatial pyramid mining for logo detection in natural scenes. In: Proceedings of the IEEE conference on multimedia expo, Hannover, pp 1077–1080
47. Levenshtein V (1966) Binary codes capable of correcting deletions, insertions, and reversals. *Cybern Control Theory* 10(8):707–710
48. Li Z, Schulte-Austum M, Neschen M (2010) Fast logo detection and recognition in document images. In: Proceedings of the 20th international conference on pattern recognition, Istanbul, pp 2716–2719
49. Logo Dataset (2012) Laboratory for Language and Media Processing (LAMP), University of Maryland. <http://lamp.cfar.umd.edu>
50. Lowe D (2004) Distinctive image features from scale-invariant keypoints. *IJCV* 60(2): 91–110
51. Lowe DG (1985) Perceptual organization and visual recognition. Kluwer Academic, Boston
52. Luo J, Crandall D (2006) Color object detection using spatial-color joint probability functions. *IEEE Trans Image Process* 15(6):1443–1453
53. Manning CD, Raghavan P, Schütze H (2008) Introduction to information retrieval. Cambridge University Press, New York
54. Meisinger K, Troeger T, Zeller M, Kaup A (2005) Automatic TV logo removal using statistical based logo detection and frequency selective inpainting. In: Proc. European signal processing conference. Antalya, Turkey, 4–8
55. Meng J, Yuan J, Jiang Y, Narasimhan N, Vasudevan V, Wu Y (2010) Interactive visual object search through mutual information maximization. In: Proceedings of the ACM international conference on multimedia, Firenze, pp 1147–1150
56. Mori G, Belongie S, Malik J (2005) Efficient shape matching using shape contexts. *IEEE Trans Pattern Anal Mach Intell* 27(11):1832–1837
57. Neumann J, Samet H, Soffer A (2002) Integration of local and global shape analysis for logo classification. *Pattern Recognit Lett* 23(12):1449–1457
58. Oliva A, Torralba A (2007) The role of context in object recognition. *Trends Cogn Sci* 11:520–527
59. Ozay N, Sankur B (2009) Automatic TV logo detection and classification in broadcast videos. In: EUSIPCO 2009, Glasgow, pp 839–843
60. Phan R, Androulacos D (2009) Content-Based retrieval of logo and trademarks in unconstrained color image databases using color edge gradient co-occurrence histograms. *Comput Vis Image Underst* 114(1):66–84
61. Pham T (2003) Unconstrained logo detection in document images. *Pattern Recognit* 36(12):3023–3025
62. Quack T, Ferrari V, Liebe B, Gool LV (2007) Efficient mining of frequent and distinctive feature configurations. In: ICCV, Rio de Janeiro
63. Ren M, Eakins JP, Briggs P (2000) Human perception of trademark images: implications for retrieval system design. *J Electron Imaging* 9(4):564–575

64. Rocchio JJ Jr (1971) Relevance feedback in information retrieval. The smart system-experiments in automatic document processing. Prentice-Hall, Englewood Cliff, pp 313–323
65. Rusinol M, Lladós J (2009) Logo spotting by a bag-of words approach for document categorization. In: ICDAR'09, Barcelona, pp 111–115
66. Rusiñol M, Lladós J (2010) Efficient logo retrieval through hashing shape context descriptors. In: Proceedings of the 9th international workshop on document analysis systems, Boston, pp 215–222
67. Rusiñol M, Nourbakhsh F, Karatzas D, Valveny E, Lladós J (2010) Perceptual image retrieval by adding color information to the shape context descriptor. In: Proceedings of the 20th international conference on pattern recognition, Istanbul. IEEE, pp 1594–1597
68. Rusiñol M, Aldavert D, Karatzas D, Toledo R, Lladós J (2011) Interactive trademark image retrieval by fusing semantic and visual content. In: Advances in information retrieval: 33rd European conference on IR research, Dublin. Lecture notes in computer science, vol 6611, pp 314–325
69. Santos AR, Kim HY (2006) Real-Time opaque and semi-transparent TV logos detection. In: Proceedings of the 5th international information and telecommunication technologies symposium (I2TS), Cuiabá
70. Sarkar S, Boyer KL (1994) Computing perceptual organization in computer vision. Series in machine perception and artificial intelligence, vol 12. World Scientific, Singapore/River Edge
71. Saund E (2003) Finding perceptually closed paths in sketches and drawings. *IEEE Trans Pattern Anal Mach Intell* 25(4):475–491
72. Saund E (2011) PPD: platform for perceptual document analysis. PARC TR-2011-1, Nov 2011
73. Schietse J, Eakins JP, Veltkamp RC (2007) Practice and challenges in trademark image retrieval. In: Proceedings of the 6th ACM international conference on image and video retrieval, Amsterdam, pp 518–524
74. Seiden S, Dillencourt M, Irani S, Borrey R, Murphy T (1997) Logo detection in document images. In: Proceedings of the international conference on imaging science, systems, and technology, Las Vegas, Nevada, USA, pp 446–449
75. Sivic J, Zisserman A (2003) Video Google: a text retrieval approach to object matching in videos. In: ICCV, Nice, pp 1470–1477
76. The legacy tobacco document library (LTDL) at UCSF (2006). <http://legacy.library.ucsf.edu>
77. Tombé K, Lamiroy B (2003) Graphics recognition – from re-engineering to retrieval. In: Proceedings of the seventh international conference on document analysis and recognition, ICDAR03, Edinburgh, pp 148–155
78. Tschumperle D (2006) Fast anisotropic smoothing of multi-valued images using curvature-preserving PDE's. *Int J Comput Vis* 68(1):65–82
79. van de Sande KEA, Gevers T, Snoek CGM (2010) Evaluating color descriptors for object and scene recognition. *IEEE Trans Pattern Anal Mach Intell* 32(9):1582–1596
80. Venters CC, Hartley RJ, Cooper MD, Hewitt WT (2001) Query by visual example: assessing the usability of content-based image retrieval system user interfaces. In: Shum H-Y, Liao M, Chang S-F (eds) PCM 2001, Beijing. LNCS 2195, pp 514–521
81. Wang H, Chen Y (2009) Logo detection in document images based on boundary extension of feature rectangles. In: ICDAR'09, Barcelona, pp 1335–1339
82. Wang J, Duan L, Li Z, Liu J, Lu H, Jin JS (2006) A robust method for TV logo tracking in video streams. In: Proceedings of the IEEE international conference on multimedia and expo (ICME), Toronto, pp 1041–1044
83. Watve A, Sural S (2008) Soccer video processing for the detection of advertisement billboards. *Pattern Recognit Lett* (29):994–1006
84. Wei C-H, Li Y, Chau W-Y, Li C-T (2009) Trademark image retrieval using synthetic features for describing global shape and interior structure. *Pattern Recognit* 42:386–394
85. Wertheimer M (1938) Untersuchungen zur Lehre der Gestalt, II. Psychologische Forschung, vol 4, pp 301–350, 1923. Translation published as Laws of organization in perceptual forms. In: Ellis W (ed) A source book of Gestalt psychology, Routledge and Kegan Paul, London, pp 71–88

86. Witkin AP, Tenenbaum JM (1982) On the role of structure in human and machine vision. In: Beck J, Hope B, Rosenfeld A (eds) Human and machine vision. Academic Press, New York, pp 481–543
87. World Intellectual Property Organization (2007) International classification of the figurative elements of Marks (Vienna classification), 6th edn. WIPO publication No. 502E/6, Geneva, Academic Press, New York
88. Wu JK, Lam CP, Mehtre BM, Gao YJ, Desai Narasimhalu A (1996) Content based retrieval for trademark registration. *Multimed Tools Appl* 3(3):245–267
89. Yan WQ, Wang J, Kankanhalli MS (2005) Automatic video logo detection and removal. *Multimed Syst* 10(5):379–391
90. Zhang D, Lu G (2004) Review of shape representation and description techniques. *Pattern Recognit* 37:1–19
91. Zhu G, Doermann D (2007) Automatic document logo detection. In: Proceedings of the international conference on document analysis and recognition, Curitiba, pp 864–868

## Further Reading

A number of key references for each of the applications discussed in this chapter are summarized in [Tables 18.3–18.6](#). The cited work is selected to cover all the various aspects of the discussed applications; hence, the proposed lists comprise a good starting point for readers interested in a broad review of the state of the art.

Readers with a specific interest in perceptual organization are encouraged to consult [51] and [21] while the “Platform for Perceptual Document Analysis” would provide further practical insight [72]. For a comprehensive overview of information retrieval systems, the interested researcher can consult [7] and [53], while for a more practical analysis, [36] offers a good starting point. Finally, a recent work dedicated specifically to logo recognition [16] provides an introduction to fundamental concepts and methods in pattern and shape recognition while surveying advances in the area.

Bertrand Coüasnon and Aurélie Lemaitre

## Contents

Introduction.....	648
History and Importance.....	648
What Is a Table?.....	648
What Is a Form?.....	650
Table/Form Representation.....	651
Objectives and Main Difficulties.....	651
Table and Form Processing.....	652
Table and Forms in Image-Based Documents.....	653
Table and Forms in Digital-Born Documents.....	667
Consolidated Systems and Software.....	672
Research Systems.....	672
Commercial Software.....	673
Conclusion.....	674
Cross-References.....	674
References.....	674
Further Reading.....	677

---

### Abstract

Tables and forms are a very common way to organize information in structured documents. Their recognition is fundamental for the recognition of the documents. Indeed, the physical organization of a table or a form gives a lot of information concerning the logical meaning of the content.

This chapter presents the different tasks that are related to the recognition of tables and forms and the associated well-known methods and remaining

---

B. Coüasnon (✉)  
IRISA/INSA de Rennes, Rennes Cedex, France  
e-mail: [couasnon@irisa.fr](mailto:couasnon@irisa.fr)

A. Lemaitre  
IRISA/Université Rennes 2, Rennes Cedex, France  
e-mail: [couasnon@irisa.fr](mailto:couasnon@irisa.fr)

challenges. Three main tasks are pointed out: the detection of tables in heterogeneous documents; the classification of tables and forms, according to predefined models; and the recognition of table and form contents. The complexity of these three tasks is related to the kind of studied document: image-based document or digital-born documents. At last, this chapter will introduce some existing systems for table and form analysis.

---

**Keywords**

Detection • Digital-born documents • Form • Identification • Image-based documents • Structure detection • Table

---

## Introduction

### History and Importance

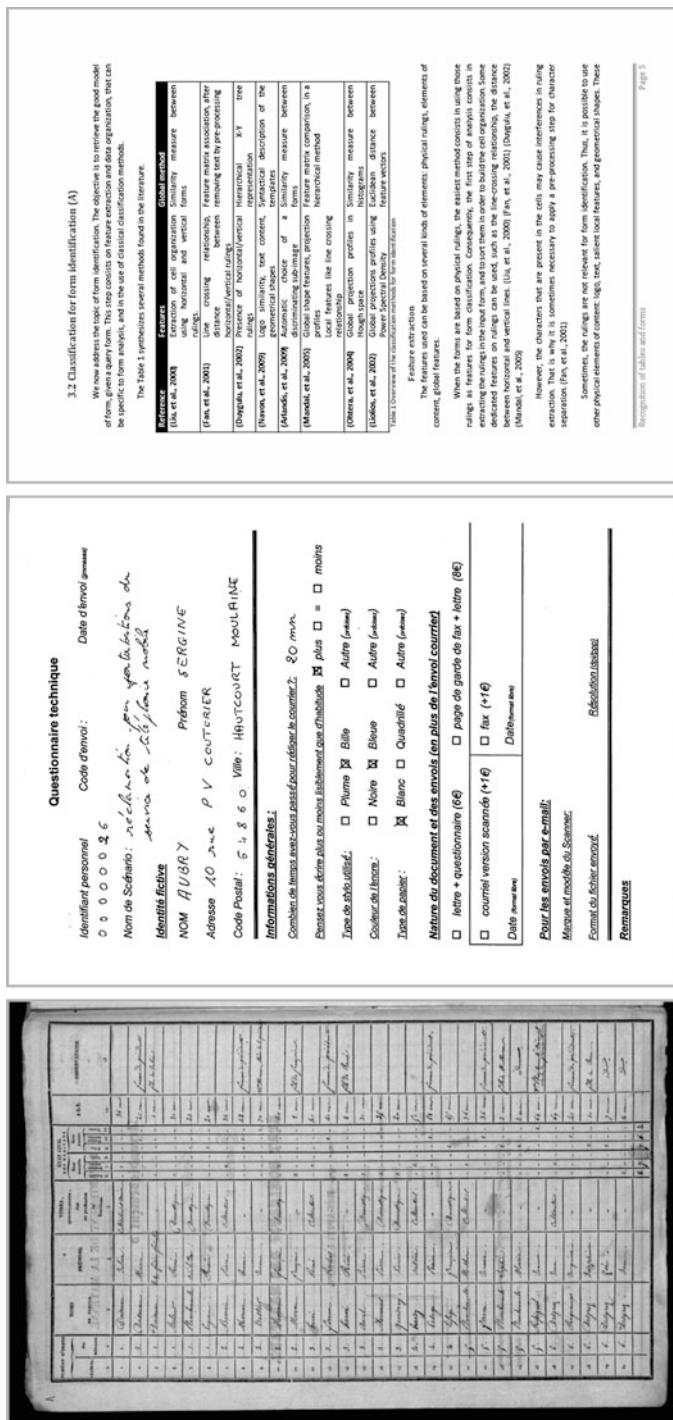
Tables and form are essential in the presentation of the information and the organization of contents in documents. Thus, they have been studied since the beginning of structured document analysis. For a long time, the research has been focused on the extraction of low-level geometric information from scanned images or paper tables. Most recent researches are focused on the analysis of tables and forms in electronic support, in order to obtain a higher level of table understanding.

The recognition of tables and forms is very important in document analysis. Thus, the physical organization of tables and forms provides some information on the logical organization of the content. Consequently, the result of a step of recognition of tables and forms can really improve the recognition of the handwriting.

Moreover, tables and forms are present in a large variety of documents (Fig. 19.1): administrative documents, invoices, question forms, old documents, etc. Consequently, a system that aims at recognizing open documents must deal with the recognition of tables and forms.

### What Is a Table?

The definition of a table has been discussed in many papers. Indeed it is quite easy to have an idea of what a table is, but a precise and formal definition is more difficult. Tables are the prevalent means of representing and communicating structured data. They may contain words, numbers, formulae, and even graphics [1]. They can contain printed information or handwritten information (mainly in archives documents). Tables are a 2D cell assembly where data type is determined by either horizontal or vertical index [2]. Costa e Silva et al. [3] propose a definition of a table as a “graphical grid-like representation of a matrix  $M_{i,j}$  where: (1) each element  $i, j$  of the matrix is atomic; (2) there are linear visual clues, i.e the elements of each row  $i$  (column  $j$ ) of the matrix tend to be horizontally (vertically) aligned;



**Fig. 19.1** Various kinds of tables/forms: hierarchical table in archive document, mixed printed/handwritten form, and digital-born document containing a table

The diagram illustrates the terminology of parts of tables. It shows a table structure with various components labeled:

- Stub Separator**: A vertical line separating the stub from the table body.
- Column Header**: The header for the columns, located above the first row of data.
- Nested Column Header**: A header for a subset of columns within a column header.
- Boxhead**: The header for the entire table, enclosed in a box.
- Boxhead Separator**: A horizontal line separating the boxhead from the body.
- Body**: The main content area of the table.
- Row Header**: The header for the rows, located to the left of the first row of data.
- Nested Row Header**: A header for a subset of rows within a row header.
- Cell**: An individual cell in the table.
- Block**: A group of cells, typically a row or column.

Term	Assignments			Examinations		Final Grade
	Ass1	Ass2	Ass3	Midterm	Final	
1991	85	80	75	60	75	75
Winter	80	65	75	60	70	70
Spring	80	85	75	55	80	75
Fall	75	70	65	60	80	70
1992	85	80	70	70	75	75
Winter	80	80	70	70	75	75
Spring	75	70	65	60	80	70
Fall	75	70	65	60	80	70

**Fig. 19.2** Terminology of parts of tables initially presented in Wang [4] and modified in [5]

(3) linear visual clues describe logical connections [...]; (4) eventual line art does not add meaning otherwise not present in the relative positioning of the cells in the table.” This definition covers a large part of tables but has some limitations to include tables that can have recursive structure, with another table structure in a cell.

The terminology of the different parts of a table is presented in Fig. 19.2. A hierarchy of columns (resp. rows) can be defined in the boxhead (resp. stub) with column (resp. row) headers and nested column (resp. row) headers. This hierarchy can be quite complex and not limited to two levels like in Fig. 19.2. Cells, columns, and rows can be delimited by physical rulings or by perceptive rulings built by alignments of cell contents.

In this chapter, tables are classified according to their complexity: simple 2D-tables and complex 2D-tables. Simple 2D-tables are limited to a matrix of cells with simple rows and columns with boxhead and stub limited to one level of hierarchy, like the table of Fig. 19.1 right. Complex 2D-tables can have some hierarchy of columns and/or rows, and may be recursive, like the table of Fig. 19.1 left. This separation, according to the table complexity, is important to see if a table recognition method is limited or not to simple tables.

## What Is a Form?

A form is a document with a predefined template mainly used for collecting information, with a one-to-one mapping between indices and data and no implication of regularity [2]. The data collected can be handwritten or machine printed, which produces segmentation difficulties with the preprint template. Even if forms are more often used for collecting data, it is important to notice that they can also be used to communicate this data.

When a form is made of rulings, it corresponds to a 1D-table, most of time horizontal, like the example of Fig. 19.3. Forms may contain 2D-tables. Ruled

<b>Hôpital Saint Jean Baptiste</b> Fiche de liaison	<b>Hôpital Saint Jean Baptiste</b> Fiche de liaison	<b>Hôpital Saint Jean Baptiste</b> Fiche de liaison
Nom : <u>COLREL</u> Prénom : <u>Sylvie</u> Date de naissance : <u>18/01/1953</u>	Nom : <u>Lerouanne</u> Prénom : <u>Martine</u> Date de naissance : <u>29/02/1953</u>	Nom : <u>SOURI</u> Prénom : <u>Auriane</u> Date de naissance : <u>12/12/1992</u>
Groupe sanguin : <u>A+</u> Date du dernier bilan sanguin : <u>28/11/2012</u> Date du prochain rendez-vous : <u>11/12/2013</u>	Groupe sanguin : <u>O-</u> Date du dernier bilan sanguin : <u>15/03/2012</u> Date du prochain rendez-vous : <u>11/08/2012</u>	Groupe sanguin : <u>AB-</u> Date du dernier bilan sanguin : <u>...../..../....</u> Date du prochain rendez-vous : <u>12/02/2013</u>

**Fig. 19.3** Example of forms

versions of forms are usually a mix of 1D-tables and 2D-tables, with 1D-tables in a cell of a 2D-tables or a juxtaposition of 1D- and 2D-tables. When a form for collecting information is made of a 2D-table only, it can be recognized as a table.

## Table/Form Representation

Table and form representation is important for table-structured data extracted by document analysis. It depends of the precision of the recognition. Ordered lists of cells and rows (or columns) can represent the table matrix. For more complex tables or forms, graphs and trees can be used to integrate relationships between form elements [1].

For defining a representation, it is interesting to study the formal table model proposed by Wang [4] for generating table images from a table description. This model is the most complete table model in the literature [5]. It separates the logical part and the physical part. The logical part contains row and column hierarchy, while in the physical part, separator type, size, and content display (fonts, alignments, etc.) are specified for a cell or a set of cells.

## Objectives and Main Difficulties

The problem of table and form processing can be split into three themes: the detection of tables in documents that contain heterogeneous information, the classification of forms using predefined models, and the recognition of the tabular structure and its content. Table and form processing can be done in both image-based documents and digital-born documents.

### Detection

The problem of table detection/localization is fundamental in the analysis of heterogeneous document images. This task consists in finding *if* there is a table/form in a document and *where* the table/form in the document is. These two tasks are often joined. This topic concerns mixed text and table document images such as scientific papers or business and law books. But the problem is also addressed for more complex documents such as internal reports, financial statements, technical and commercial notes, magazines, scientific papers, tickets, and certificates.

The detection of table is as well sometimes necessary in the context of mixed printed/handwritten documents.

In the field of digital document processing (text, HTML, PDF), the table localization is a challenging task. Thus, in text documents, the tables are not always marked by physical rulings but by tabulations and spaces, which complicates their detection. Table detection can also be difficult in PDF documents. At last, the detection of tables in HTML documents requires a specific processing as the <TABLE> HTML tag is often used for both real meaningful tables and for the construction of the web page layout.

### **Classification**

Another task for table/form processing, which differs from the complete recognition of the document, is form classification. What is called form, in this context, are preprinted documents that are hand-filled. Thus, for the processing of the forms, the objective is to identify the type the candidate form belongs to and then to analyze the content of handwritten fields.

One may identify two levels of complexity for the classification. When the candidate form is exactly the same as the blank model, the difficulties are to deal with skew, shift, and the presence of handwriting. In more complex cases, the forms can vary inside of a given template: size of boxes, text, print quality, scales, etc. In these cases, the methods have to deal with the variation of templates and possibly with a reject option.

### **Recognition**

The most complex task in table processing consists in the recognition of the physical and the logical content. Recognition implies to understand the structural organization of tables to extract data and if necessary to reorganize this data.

The recognition of tables often requires specific systems that are adapted for the analysis of the tabular structures. Once the structure is detected, the recognition of the content can often be realized using the classical writing recognizers. However in complex documents, the recognition of the content is necessary to perform the structure recognition. Consequently the process of table recognition needs to combine both structure and content recognition.

---

## **Table and Form Processing**

The main content is organized as follows. Two kinds of documents will be studied: the image-based documents and the digital-born documents. For each kind of document, there will be three tasks: the detection, the classification, and the recognition.

## Table and Forms in Image-Based Documents

This section focuses on the analysis of images of documents. They are handwritten, printed, or mixed documents that have been digitized. There is no available electronic metadata for the recognition of these documents. For this kind of documents, three tasks are studied:

- The detection of tables/forms in heterogeneous documents
- The classification of forms into models
- The recognition of tables

### Detection

In document images, the simplest way for table detection and localization is to analyze structural features. It is possible to define rules or heuristics that are applied to detect the presence of tables in complex documents.

The easiest configuration for table/form localization is when physical rulings are present in the documents. Thus, the strategy for table localization consists in first extracting the rulings and then looking for a specific arrangement in lines, rows, and cells.

For the detection of rulings, the classical techniques of line segment detection can be used (see ►Chap. 15 (Graphics Recognition Techniques)). Then, the extracted line-segments can be organized to match with a tabular structure. A tabular structure can be characterized by the presence of at least two parallel lines, having the same size.

Depending of the used technique, the ruling extraction may require a phase of preprocessing, like skew correction. Some methods are especially convenient for table analysis (►Chap. 4 (Imaging Techniques in Document Analysis Processes) gives details on imaging techniques for document analysis). A recursive X-Y tree analysis of the page requires thresholds like the criteria to decide whether two parallel lines have the same length, or the size of a white strip to separate columns. These thresholds can be learned using the principle of a solving and optimization problem [6].

Some recursive structural approaches can also be used in regular documents. Thus, it is possible to realize a recursive decomposition of the document images into rectangles until they form regular cells. This approach is used by Ramel et al. [7] in the analysis of printed PDF documents.

In order to avoid specific learning, the morphological approaches enable to estimate the presence of vertical rulings, horizontal rulings, and the intersection of rulings [8] (Fig. 19.4). The morphological tools are particularly convenient for the analysis of hand-filled tables/forms. Indeed, the presence of handwriting complicates the detection of the rulings. The use of watershed transform, combined with statistical analysis, enables to deal with the noise due to the presence of text inside the cells [9] (Fig. 19.3).

sults of all the thinning algorithms.

TABLE II  
STATISTICS ON THINNING ALGORITHMS  
(FOR 1,000 CHARACTERS IN DATABASES 1 AND 2)

Algorithm	Database 1				Database 2			
	Time (sec)	Extra Points	End Points	Printed	Time (sec)	Extra Points	End Points	Printed
Chen & Hsu [3]	39.0	373	2109	2295	184	431	2284	2440
Chen & Tsai [5]	54.3	19072	2189	2323	179	36369	2541	2510
Chin et al. [6]	15.0	2342	2145	2238	65	3578	2404	2454
Ong & Hall [8]	9.5	1004	2186	2315	41	968	2351	2468
Hilditch [12]	12.9	0	2221	2337	60	0	2384	2486
Hok et al. [13]	12.2	7387	2119	2294	49	13828	2281	2442
Suzuki & Abe [23]	3.9	0	2250	2330	14	0	3447	2511
Wang & Zhang [25]	9.3	1111	2085	2287	41	1118	2229	2438
Wu & Tsai [26]	12.4	433	2190	2305	53	489	2560	2466
Zhang & Sun [28]	10.0	8538	2084	2286	42	15359	2232	2437

### C. Number of End Points

sults of all the thinning algorithms.

TABLE II  
STATISTICS ON THINNING ALGORITHMS  
(FOR 1,000 CHARACTERS IN DATABASES 1 AND 2)

Algorithm	Database 1				Database 2			
	Time (sec)	Extra Points	End Points	Printed	Time (sec)	Extra Points	End Points	Printed
Chen & Hsu [3]	39.0	373	2109	2295	184	431	2284	2440
Chen & Tsai [5]	54.3	19072	2189	2323	179	36369	2541	2510
Chin et al. [6]	15.0	2342	2145	2238	65	3578	2404	2454
Ong & Hall [8]	9.5	1004	2186	2315	41	968	2351	2468
Hilditch [12]	12.9	0	2221	2337	60	0	2384	2486
Hok et al. [13]	12.2	7387	2119	2294	49	13828	2281	2442
Suzuki & Abe [23]	3.9	0	2250	2330	14	0	3447	2511
Wang & Zhang [25]	9.3	1111	2085	2287	41	1118	2229	2438
Wu & Tsai [26]	12.4	433	2190	2305	53	489	2560	2466
Zhang & Sun [28]	10.0	8538	2084	2286	42	15359	2232	2437

### C. Number of End Points

sults of all the thinning algorithms.

TABLE II  
STATISTICS ON THINNING ALGORITHMS  
(FOR 1,000 CHARACTERS IN DATABASES 1 AND 2)

Algorithm	Database 1				Database 2			
	Time (sec)	Extra Points	End Points	Printed	Time (sec)	Extra Points	End Points	Printed
Chen & Hsu [3]	39.0	373	2109	2295	184	431	2284	2440
Chen & Tsai [5]	54.3	19072	2189	2323	179	36369	2541	2510
Chin et al. [6]	15.0	2342	2145	2238	65	3578	2404	2454
Ong & Hall [8]	9.5	1004	2186	2315	41	968	2351	2468
Hilditch [12]	12.9	0	2221	2337	60	0	2384	2486
Hok et al. [13]	12.2	7387	2119	2294	49	13828	2281	2442
Suzuki & Abe [23]	3.9	0	2250	2330	14	0	3447	2511
Wang & Zhang [25]	9.3	1111	2085	2287	41	1118	2229	2438
Wu & Tsai [26]	12.4	433	2190	2305	53	489	2560	2466
Zhang & Sun [28]	10.0	8538	2084	2286	42	15359	2232	2437

### C. Number of End Points

sults of all the thinning algorithms.

TABLE II  
STATISTICS ON THINNING ALGORITHMS  
(FOR 1,000 CHARACTERS IN DATABASES 1 AND 2)

Algorithm	Database 1				Database 2			
	Time (sec)	Extra Points	End Points	Printed	Time (sec)	Extra Points	End Points	Printed
Chen & Hsu [3]	39.0	373	2109	2295	184	431	2284	2440
Chen & Tsai [5]	54.3	19072	2189	2323	179	36369	2541	2510
Chin et al. [6]	15.0	2342	2145	2238	65	3578	2404	2454
Ong & Hall [8]	9.5	1004	2186	2315	41	968	2351	2468
Hilditch [12]	12.9	0	2221	2337	60	0	2384	2486
Hok et al. [13]	12.2	7387	2119	2294	49	13828	2281	2442
Suzuki & Abe [23]	3.9	0	2250	2330	14	0	3447	2511
Wang & Zhang [25]	9.3	1111	2085	2287	41	1118	2229	2438
Wu & Tsai [26]	12.4	433	2190	2305	53	489	2560	2466
Zhang & Sun [28]	10.0	8538	2084	2286	42	15359	2232	2437

### C. Number of End Points

sults of all the thinning algorithms.

**Fig. 19.4** Table detection based on physical rulings (initial image, detected line intersections, image without rulings, final table reconstruction) [8]

In many cases, there are no physical rulings for the delimitation of each cell. Some rulings are sometimes present to delimit only certain columns or rows. Consequently, table detection must be based on other physical elements such as regularity of the positions, alignments, and presence of white spaces.

The simplest approach consists in the study of blocks that are horizontally or vertically aligned, with a certain threshold of similarity. The projection method can be used to build columns and rows. However, this method requires parameters and clean documents. This approach is followed by the T-Recs table recognition system [10].

Some other important information for table detection is the fact that the spacing between columns is regular. Thus, it is possible to group connected components into word and words into lines and then to segment the lines into columns depending on the interword space. This method does not need specific parameters [11]. However, it is dedicated for documents with Manhattan layout and may not work for complex documents with heterogeneous arrangements.

**Table 19.1** Example of forms

Reference	Features	Global method
[12]	Extraction of cell organization using horizontal and vertical rulings	Similarity measure between forms
[13]	Line-crossing relationship, distance between horizontal/vertical rulings	Feature matrix association, after removing text by preprocessing
[14]	Presence of horizontal and vertical rulings	Hierarchical X-Y tree representation
[15]	Logo similarity, text content, geometrical shapes	Syntactical description of the templates
[16]	Automatic choice of a discriminating sub-image	Similarity measure between forms
[17]	Global shape features, projection profiles Local features like line-crossing relationship	Feature matrix comparison, in a hierarchical method
[18]	Global projection profiles in Hough space	Similarity measure between histograms
[19]	Global projections profiles using Power Spectral Density	Euclidean distance between feature vectors

As a conclusion, the structural methods based on rules and heuristics are convenient and widely used for table detection, more particularly when the table cells are delimited by physical rulings. However, in more complex documents, the analysis of the content could improve the table localization. Section “[Detection](#)” will show that some knowledge on metadata or the content of documents in digital-born documents enables to treat more complex documents.

### Classification for Form Identification

The objective of form identification is to retrieve the good model of form, given a query form. This step consists on feature extraction and data organization that can be specific to form analysis and in the use of classical classification methods. Some of features and classification methods are described in [►Chap. 11](#) (Handprinted Character and Word Recognition).

Table 19.1 synthesizes several methods found in the literature.

**Feature Extraction** The features used can be based on several kinds of elements: physical rulings, elements of content, and global features.

When the forms are based on physical rulings, the easiest method consists in using those rulings as features for form classification. Consequently, the first step of analysis consists in extracting the rulings in the input form (e.g., with methods presented in [►Chap. 15](#) (Graphics Recognition Techniques)) and to sort them in order to build the cell organization. Some dedicated features on rulings can be used, such as the line-crossing relationship and the distance between horizontal and vertical lines [12–14, 17].

However, the characters that are present in the cells may cause interferences in ruling extraction. That is why it is sometimes necessary to apply a preprocessing step for character separation [13].

Sometimes, the rulings are not relevant for form identification. Thus, it is possible to use other physical elements of content: logo, text, salient local features, and geometrical shapes. These features are similar to the human visual system whose recognition is based on dominant features of the content [15].

It is also possible to automatically define which are the most discriminating regions at the image level. Thus, each document class can be associated to a discriminating landmark area, as a sub-image that can be used to discriminate the class from the others. The interest of these features is that it can be used for both identification and reject of forms [16].

The third category of features is global features for form identification. The global horizontal and vertical projection profiles are basic ideas that can be used, with a little adaptation in order to deal with noise and deformations or skew. For example, it is possible to use only the 2nd- and the 4th-order moments of the projection profile [17], or to study these features in Hough space [18], or using the Power Spectral Density [19].

**Data Organization** Once features have been extracted, it is necessary to organize them for the classification. The usual methods can be used, such as similarity measures between histograms [18], and feature vectors or matrixes [12, 13, 16, 17, 19].

However, the bidimensional hierarchical nature of forms can be represented using more hierarchical data structures such as X-Y tree. This data structure enables to describe the logical configurations of forms [14].

The syntactical approaches are also adapted to the recognition of specific models of forms. Thus, it is possible to build a syntactical template using various features. The introduction of each new kind of form is realized by the definition of a new template [15].

**Remaining Problems** Form classification seems to be an easy task when the form model is well defined. However, it remains a challenging task when the structure of the form can vary a little. In that case, it is necessary to study the logical structure in order to deal with variations on the physical structure.

## Recognition

Recognition of tables and forms implies to understand the structural organization of tables to extract data and if necessary to reorganize this data. It corresponds to an analysis of the logical layout that is presented in a more general way in ▶Chap. 6 (Analysis of the Logical Layout of Documents). In image-based documents it will be necessary to deal with all the difficulties linked to image processing like noise, segmentation problems, imprecise and uncertain information, and damaged documents.

If a table or a form has a very stable model, then its identification reduces the recognition phase to a simple task: once the exact model of the form and its structural organization is known, it is only necessary to extract information in each cell, according to the precise model.

In this section a more complex task is presented: recognize a table structure where dimensions and structural organization are not the same from one table to another, without changing the knowledge. Some methods recognize and locate tables/forms at the same time, and others consider that the table is already located.

The recognition process needs to first extract the elements or features on which the form or table structure is built. This section will start by a presentation of those features and how they can be extracted and will be followed by some recognition methods using knowledge on table structure. The knowledge can be limited to geometric information, or can be both geometric and cell-content information, which can be extracted by OCR or handwriting recognition.

**Features for Table and Form Recognition** Features are in fact the basic elements on which table structures are built. Here are the most classical features, followed by some ways of extracting them:

- Regions and text blocks
- Character or word bounding boxes
- Words recognized by OCR
- Rulings, ruling intersections, and terminal points of rulings

Regions, text blocks, characters, and words bounding boxes can be simply detected with connected components and contours of objects. A low-resolution image can help for this detection like in [20].

For words recognized by OCR, they can be extracted and segmented directly by an OCR applied on the whole page or by an OCR applied on each word (with its bounding boxes).

For ruling detection, simple methods can be used, but they are usually unable to deal with damaged rulings: curve, large break, and touching cell content. For example, rulings can be seen as long black run length that can be broken and then are “stitched” together, like in [21], or as connected components with a large or small aspect ratio or even as enclosed blocks detected as a path of continuous pixels in the thinned image [22].

It is also possible to detect ruling intersection without detecting rulings by applying mathematical morphology [23].

For more difficult rulings, a Kalman filtering applied on a sequence of run length with a line-segment model is able to deal with broken, curved, and touching symbol rulings [24]. A perceptive method with multi-resolution is even more robust to damaged rulings [25].

Once rulings are well detected, it is possible and easy to extract intersections and final intersections of rulings [26].

With the help of these features extracted on tables and forms, various recognition methods can be applied to recognize the table structure: methods using only geometric information and methods using cell-content information to complete

**Table 19.2** In image-based documents: overview of table and forms recognition methods using geometric information without any logical knowledge on tabular structure

Reference	Features	Recognition method	Table complexity
[20]	Region oriented	Box-Driven Reasoning	1D-tables, forms
[27]	Text blocks	Texts blocks labeled by Neural Network, graph built on labeled blocks. Graph grammar	Simple 2D-tables
[21]	Word bounding boxes, rulings	Black run lengths for rulings, projections and histograms of word or cell's bounding boxes for white separators	Simple 2D-tables
[28]	Logical region type, logical relation types, direct graph	Recognition Strategy Language (RSL). Graph transformations	Simple 2D-tables
[22]	Text blocks, ruling-enclosed blocks	Cells extraction with horizontal and vertical projections of bounding boxes	Complex 2D-tables
[23]	Ruling intersections	Mathematical morphology	Complex 2D-tables
[29]	Type of rulings intersections, terminal points of lines, imaginary lines	Cell detection using a grid representation	Complex 2D-tables

geometric information. The difficulty of this recognition task depends on the kind of tables: simple 2D-tables or complex 2D-tables. Indeed, complex 2D-tables are more difficult to recognize because of the large variation of layout that can be found. As there are a lot of ambiguities on the structure, it is necessary and more difficult than for a noncomplex table to deal with low-quality images of documents. For each following method, it is indicated if it can deal with simple or complex 2D-tables.

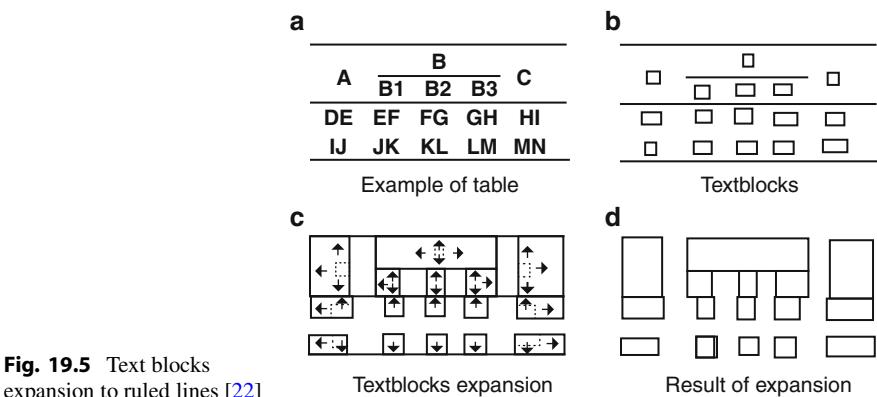
**Recognition Using Only Geometric Information** The following methods use only geometric information. These methods can be also divided according to the fact that they use or not a priori logical knowledge on tabular structure. An overview of methods without logical knowledge on tabular structure is presented in Table 19.2 and of methods with logical knowledge in Table 19.3.

#### Projections and Histograms

Projections and histograms are classical techniques in image document analysis ►Chap. 5 (Page Segmentation Techniques in Document Analysis). For table structure recognition, it is possible to extract cells with horizontal and vertical projections of bounding boxes of text blocks and ruling-enclosed blocks [22]. This is done after text block expansion limited by the detected rulings (Fig. 19.5). White separators

**Table 19.3** In image-based documents: overview of table and form recognition methods using geometric information and logical knowledge on tabular structure

Reference	Features	Recognition method	Table complexity
[30]	Cells (boxes) labeled with four classes (box type). Not automatically extracted	Logical structure extraction of hierarchical tables. Graph grammar on graph where nodes are boxes and edges represent adjacency of two boxes	Complex 2D-tables
[31]	Rulings	Kalman filtering and perceptive vision with multi-resolution for ruling detection. Bidimensional grammatical formalism (EPF), to locate and recognize recursive table structures	Complex 2D-tables
[26]	Final intersections of rulings	Specific language to define the logical and the physical structures of tables with a possible hierarchy of columns or rows	Complex 2D-tables



**Fig. 19.5** Text blocks expansion to ruled lines [22]

can also be detected with projections and histograms of word or cell's bounding boxes [21]. This method can deal with fully lined, semi-lined, or lineless tables, but it is limited to simple 2D-tables. In complex 2D-tables, projections can extract partially ruled tables, with cells that span multiple rows or columns, but the method seems too limited to be applied to real documents with skew and segmentation difficulties (broken rulings, touching characters, etc.). Moreover, even on non-damaged documents, the method has some difficulties to detect correctly the white rulings. No experimental results have been presented on this method.

### Specific Methods with Region and Intersection Analysis

Some hard-coded methods try to recognize table structure in intersection or region organizations. For example, on simple 2D-tables, the method called Box-Driven Reasoning (BDR) [20] deals with regions to analyze the structure of table/form documents which include touching characters and broken lines. Boxes are assigned labels of classes according to their sizes and relationships. Then touching characters are separated, and with specific rules, BDR composes cell boxes, extracts missing cell boxes, and adjusts locations of boxes. Unfortunately, the presented experiments are done on only ten documents making it difficult to be convinced by the method.

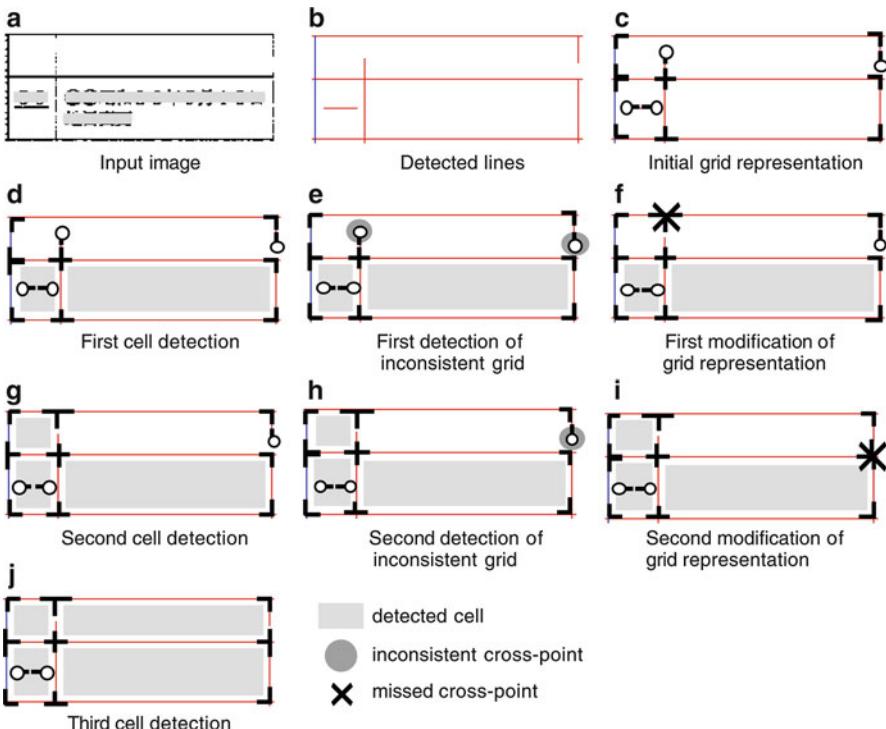
To deal with noise in document images, for complex fully ruled 2D-tables, one can use intersections and terminal points of rulings and the context of the neighboring intersections, to detect some missing intersections (due to broken rulings) or to correct false intersections (due to overlapping text) [23]. Experimentations have been done on more than 300 tables/forms of the same type, artificially skewed: on empty tables/forms, the system has 100 % recognition, but on filled tables/forms, recognition rate fails down to 70 %. This shows that the difficulties introduced by the handwritten text touching cells borders are not really solved. One of the reasons is that the proposed method does not really use information on the structural organization of the tables, even if it tries to process complex 2D-tables. Experimentations are once again too narrow (limited to one type of table) to conclude that the method can deal with various types of complex tables.

A grid representation of the table/form can be also generated with the type of the intersections, the terminal points of the lines, and the imaginary lines [29]. Then, a recursive analysis is done to modify the grid to correct the cell detection. It makes a detection of inconsistent grid-cross-points and missed cross-points with three rules, until a consistent grid is obtained (Fig. 19.6). Experimentation has been carried on more than 1,500 tables/forms, with more than 500 different types of table. This is significant, but no character touches frame lines in any of the forms. Even if this method seems interesting, it might have the same kind of difficulties as [23], with touching characters.

### Graph Grammars

Graph grammars need to first build text blocks or cells in a graph that can then be transformed by graph grammar rules. For simple 2D-tables, a Neural Network can label text blocks as paragraphs, column structure, tabular structure, indexed list, jagged text, and unformatted text region [27]. Then a layout graph is built which nodes and edges, respectively, represent these labeled text blocks and their interrelations. The graph is then rewritten using graph grammar production rules based on a priori document knowledge and general formatting conventions. The resulting graph extracts the logical structure of a document from its layout graph. This graph contains subparts corresponding to the different logical elements of each recognized table: columns, table, header, etc.

It is possible to go further with graph grammars to extract some hierarchical table structure in complex 2D-tables. For example, like in [30], cells (boxes) can be labeled first with four classes (box type), and this labeling should be done



**Fig. 19.6** Detection and modification of inconsistent grid-cross-points [29]

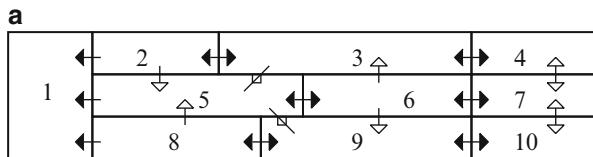
automatically, but the paper does not explain how it could be done. A graph is built where a node is a box with a box type, and an edge is an adjacency of two boxes (Fig. 19.7). The graph grammar transforms it in a graph representing the logical structure. TFML, an XML table representation format, is proposed to represent the recognized table.

Unfortunately, these graph grammars methods have not been validated on their ability to deal with difficult documents as almost no experimental results are given. Authors also pointed out some limitations of the method: it does not account for overlapping entities, and it does not find the best possible answer in the case of an ambiguity, as no mechanism is implemented for backtracking. Moreover graph grammar parsers do not deal with segmentation problems, as the input graph is not called into question.

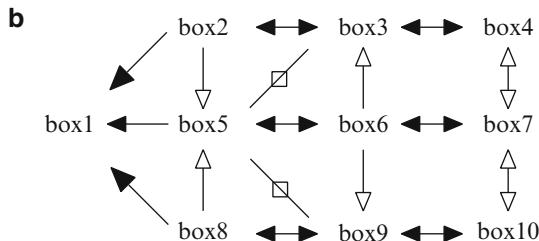
### Description Languages

Description languages of table structure are interesting for building generic methods where the knowledge can be more easily introduced. On simple 2D-tables, the Recognition Strategy Language (RSL), proposed in [28], is a formalization of document analysis methods. The RSL has been tested by defining two methods for

**Fig. 19.7** Graph representation of table form document [30]



Box adjacencies of table form document.



Graph representation of (a).

table analysis: the Handley method [21] presented just previously in the projections and histograms recognition section and the Hu method [32] presented in section “[Table and forms in Digital-Born Documents](#)”.

RSL has four main types of data: the set of logical region types, the set of logical relation types, a single global set of static and adaptive recognition parameters, and directed graphs representing interpretations of the input. The input to an RSL specification is a graph, and the output contains the set of accepted interpretation graphs.

This is an interesting method for the formalization of document analysis strategies and its genericity. Unfortunately authors pointed out that RSL is limited on dealing with pixel-level information, which is a strong limitation for all difficulties linked to noise and segmentation.

On complex 2D-tables, DMOS [31], a generic method for structured documents recognition, already applied on musical scores and mathematical formulae, has been applied to table structure documents. This method is made of a grammatical formalism, the Enhanced Position Formalism (EPF) and an associated parser, which allows modifying the parsed structure during parsing to deal with segmentation problems. As the method is generic, it has an important feature: it allows defining either a general description or a specific description according to the document quality. An EPF grammar can be used to describe a recursive table structure build on rulings. This EPF description is of course independent of the number of cells, rows, columns, cells size, and depth of recursion. It searches for the first level in a recursive table structure and in each cell looks for table in a recursive way. This method is able to locate a table in a document and to recognize it at the same time.

When a table is too damaged with some missing parts of rulings, the general description cannot be used, but the same generic method can be applied to define a specific description to compensate the missing information. A specific description in EPF of damaged military forms from the nineteenth century has been presented. The system has been validated on more than 80,000 pages.

Tables of archives documents can be very difficult to process: rulings are broken and curved, and ink bleeds through the paper; thus, rulings of flip side can be visible. Tables can also have a physical structure that changes from one page to the next one, but with the same logical structure, like it can be found in census pages from the nineteenth century. To overcome all those difficulties, in tables with rulings, Martinat [26] proposes a specific language to define the logical and the physical structures of tables with a possible hierarchy of columns or rows. From a description in this language, a recognizer is compiled. This recognizer is built on the final intersections of rulings. By matching the final intersections deduced from the table description and the one extracted from the image, the system is able to recognize a set of tables with complex physical structures with column and row hierarchies even when the table structure is damaged (Fig. 19.8).

### Recognition Using Geometric and Cell-Content Information

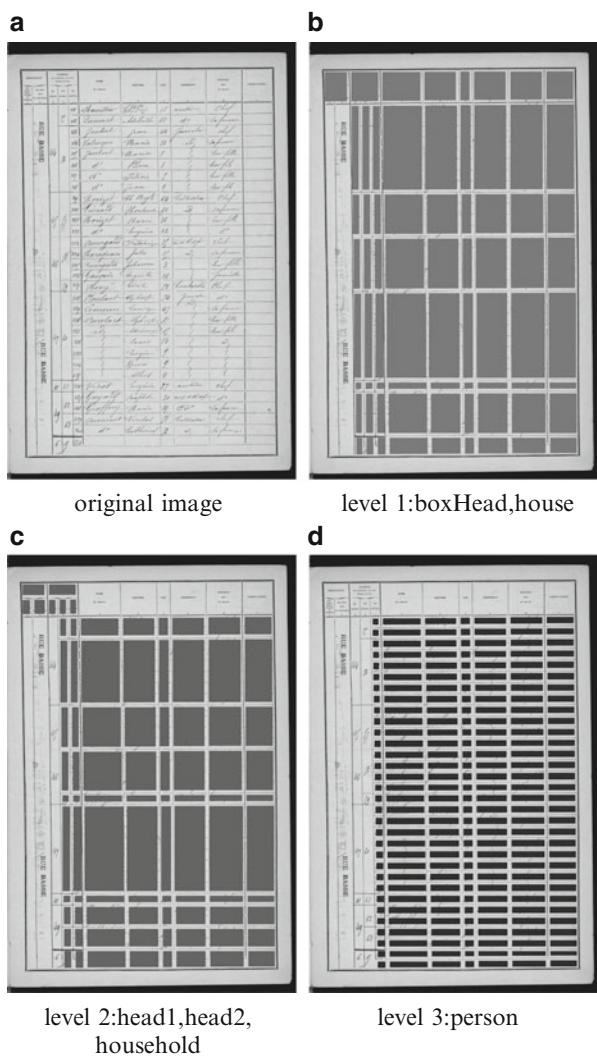
Only few methods are using both geometric and cell content information. They are all widely validated, as they are all commercial systems. This is quite normal, as commercial systems have to be able to extract the table content and not only the table structure. Using cell-content information can improve the recognition of the table/form structure by using the results of an OCR. But it can also improve the OCR itself when characters are touching or are not well printed, in case an external database is available to bring some more knowledge on each cell/field of a table/form. Table 19.4 presents an overview of methods using geometric and cell-content information.

#### Language Description

For 1D-tables and forms, a generic method build on the printed content extracted by OCR has been implemented in ABBYY FlexiCapture Studio product [33]. This product offers a set of tools for creating, testing, and compilation and use of flexible templates, or FlexiLayouts. Three basic principles of human perception are applied in this method: integrity, purposefulness, and adaptability (IPA). The method is applied to forms with variable layouts, like invoice and payment orders, to extract some of the fields. A language for describing document structure is associated to a top-down analysis. The language allows describing elements (mostly words/numbers extracted from OCR) and relative positions. The best hypothesis is selected according to a score linked to the score of the OCR.

This method has been widely validated as a commercial product, but it is just at the frontier of table structure recognition as no real table understanding is done. It is more an analysis of printed fields with the help of textual anchorage points. This is

**Fig. 19.8** Census Table of 1881 and the recognized structure with a general description; column number is unfixed like row number at each level of hierarchy [26]



interesting in the way cell content can be used, but it is still limited for tables with segmentations difficulties like rulings touching text, because the method does not propose a calling into question of the initial segmentation.

#### Constraints from Database for Improving OCR

To deal with some characters touching rulings, a system is proposed in [34] for simple 2D-table reading, mostly 2D row and column matrix of fields, with or without rulings. Contrary to [33], this method does not start by OCR but by

**Table 19.4** Overview of table and forms recognition methods using both geometric and cell content information in image-based documents

Reference	Features	Global method	Table complexity
[33]	Words extracted by OCR	Language to describe elements (mostly words/numbers extracted from OCR) and relative positions	1D-tables, forms
[34]	Characters, ink template	Remove horizontal rulings. Compute “ink template” with projections. Column estimation using known table columns structure. Field location after classifier-driven resegmentation	Simple 2D-tables
[35]	Words extracted by OCR	OCR results are corrected by constraints coming from a database with all the information which can be found in the table (e.g., all possible articles for invoice processing)	Simple 2D-tables

detecting and removing rulings. The next major stage locates the region(s) in which the table(s) is (are) placed. The page is deskewed. Classically, the method uses projections at low resolution to detect columns and an “ink template” to represent the presence of black and white pixels. This produces an estimation of columns, which is then used for each text line to compute and encode the ink template. From this encoding the text line is judged to match or not the record template in a database of known table column’s structure.

On the candidate record lines found before, a similar search is done at full resolution, to refine the field location of the best match. Then a classifier-driven resegmentation scheme is invoked. It uses a neural net and produces the best field segmentation according to the cell content.

This is an interesting method able to deal with some character touching rulings (only horizontal) and to identify/segment record lines. But to do that the method needs to know the layout of records, and when an unknown layout appears, a user has to manually add this new layout style. In this method the cell content is used to improve the segmentation, and the method is mainly based on geometrical information. This method has been validated as a commercial product and processed more than 400 distinct tabular layouts and has read over 50 million records.

A document analysis and understanding system, SmartFix, presented in [35], has a specific part for table analysis, able to extract forms with known layout and to extract rows of tables like in invoices when some of the cell content is already known in a database. Two examples of forms are presented in Fig. 19.9. OCR results are corrected by constraints coming from the database. Constraints can be exact values, fuzzy values, or a range of values. A database query returns all possible line results according to the constraints. The final step is the rating of the fields that allows making a decision on the relative score of each hypothesis of lines. This is an interesting way of improving the results of cell-content recognition, but it requires

<b>Prof. Dr. med. Anja Beck</b> 79285 Sibringen		<b>Verrechnungsstelle für Ärzte Oswald Helmstaer GmbH</b>	
Kinderärztin		Rechnung	
Bankverbindungen:		Geschäftsnr.: Abs. OG 1220-1100 Fach 307 - 030 Geschäftsbereich: Hausarzt Postfach 222 9009 Nürnberg	
Commerzbank		Von: *Haben von: *Buchungstag Geschäftszahl: 0011000000000000 Geschäftsort: Postamt 222 9009 Nürnberg	
Friedr. Dr. med. Anja Beck, Sibringen 135, 79285 Sibringen		Dr. med. Fritz Müller Luxemburger Str. 5 67657 Kaiserslautern	
Herrn Klaus-Peter Schmidt Rosenstr. 98 50733 Köln		Leistung	
Patient: Schmidt, Klaus-Peter , geb. 10.12.1954		GK: 08	
Für ärztliche Behandlungen		VD cutane Leishmaniose, persist. Iktus	
erlaube ich mir DM 136,36 zu berechnen.		Beratung - auch mittels Fernsprecher Symptombezogene Untersuchung Beratung - auch mittels Fernsprecher Symptombezogene Untersuchung Infektionsanamnese, klinisch begleitende Körpergewebe Projektion aus Oberflächlich betroffenen Körpergewebe Beratung - auch mittels Fernsprecher Telefon	
D i a g n o s e : Großzeh: Ekzem am li. Onychomycose re. Großzeh: Ekzem am li. Unterarm-Begasseite und li. Außenknöchel: Unterarm-Begasseite und li. Außenknöchel:		Datum Factor Gewalt: Stärk.	
14.09.98 1 Beratung auch telefonisch Untersuchung, symptombezogen		Zu liquidiieren	
298 Abstrichnahme, Mikrobiologie		2.300 20,98 2.300 20,98	
3508 Mikroskopie, Naturpräparat		2.300 10,49	
4716 Palpation, Aufmerksamkeite im Bereich der betroffenen Körperteile)		1.150 15,73	
07.10.98 1 Beratung auch telefonisch Untersuchung, symptombezogen		2.300 20,98 2.300 20,98	
16.10.98 4722 Palpation, lichtmikroskop. Identifizierung re. Unterarmseite		1.150 15,73	
		136,36	

**Fig. 19.9** Examples of medical bills with table structure processed by SmartFix with the help of data coming from a database for each field of the table [35]

an access to a database with all the information that can be found in the table. Unfortunately, this information is not always available for the table recognition process.

## Conclusion

Table and form analysis in image-based documents is a difficult problem, mainly due to the conjunction of different elements: the quality of the document (rulings can be broken and curved and ink can bleed through the paper...), characters can touch other characters or rulings, and the table structure can be complex. Many different methods have been proposed, but in many cases they are not validated on a representative dataset. To overcome the difficulties of image-based documents, it may be necessary to introduce logical knowledge in the system, by building generic methods with languages to describe this knowledge. Moreover, it is important to use cell-content knowledge, coming from an OCR, and when available it is important to use external knowledge coming from databases to improve OCR and to deal with difficulties of segmentation (touching characters or touching rulings). This is particularly true for complex 2D-tables. To sum up, it is necessary to mix information at pixel level with logical information on table structures and cell-content information.

## Table and Forms in Digital-Born Documents

This section focuses on the analysis of documents that exist under a digital form, such as PDF, PostScript, HTML, and text documents. These documents present specific challenges that differ from the analysis of document images. Their analysis is described more generally in ▶ [Chap. 23](#) (Analysis of Documents Born Digital). Thus, contrary to the document images, the electronic documents are made of a precise signal that contains the different elements of the document, without ambiguity. Thus, the step of recognizing the individual primitives is direct as it only consists in reading the electronic format. However, it appears that the level of used primitives varies a lot depending on the kind of document.

For example, exchange formats such as PDF, and PostScript are made of a set of printing instructions given to a virtual printer [7]. Each instruction concerns an element such as character, text, graphic line, rectangle, and ellipse. Concerning HTML documents, the electronic signal contains structured data, due to the presence of tags. However, the tags are not always used to describe the logical content of a document. Thus, the <TABLE> tag can be used for both decorative tables (the layout organization of the page) and meaningful tables [36, 37] in HTML pages. Consequently, the presence of electronic signal simplifies the analysis of primitives in documents but requires however an important processing for table/form logical structure recognition.

For this kind of documents, two tasks are described below:

- The detection of tables/forms in heterogeneous documents
- The recognition of tables

## Detection

Concerning the localization of tables/forms in the documents, the task can be addressed using two ways: with structural descriptions or with statistical methods.

### Structural Methods for Table Localization

The table localization in digital-born documents can be treated with the same methods as the ones used for document images (section “[Detection](#)”). Thus, it is possible to use image approaches on those documents. As they are clean documents, the techniques used for image can easily succeed.

However, in digital-born documents, it is necessary to exploit the knowledge on the content for the localization tables. It is possible to build heuristics on structural information: presence of large gaps in the middle of the lines, alignment of gaps in the middle of the lines, and pattern regularity between lines. Some heuristics on content elements can also be built in the same way they could be perceived by a human reader. Thus, the content elements can be aligned and grouped in a bottom-up way to exploit spatial relationship among them and build lines, blocks, rows, and the final cell grid [38, 39].

The presence of textual characters can also enable the determination of heuristics: proportion of specific characters in a line (space, –, \*), relative locations in a line and across, etc. All these structural heuristics can be used to perform table localization in digital-born documents [40].

In order to enrich the structural description of documents, it is possible to use the presence of some semantic content. Thus, when the knowledge of the content is available, for example, in PDF documents, a keyword-matching method can be used for the detection of significant words like “Table” and “Form.” Moreover, the read order of the document plays an important role in the table localization. Thus, it can be interesting to recover the text sequence order. Some algorithms based on the concept of sparse line that take into account the presence of columns and figures can be used [41].

### Statistical Methods for Table Localization

In order to automatically adapt the methods to new databases, it is possible to use machine learning-based methods. These approaches are widely used in the literature and presented in Table 19.5.

The first kind of features is layout features or appearance features. Depending on the kinds of documents, it is possible to use:

- The number of columns and the number of rows
- The kind of crossings
- The presence of borders and cell baselines
- The presence of blank blocks
- The presence of justification or left alignments
- The distance between lines

Those layout features can be used to determine the presence of possible separators, text blocks, and tables. Thus, the probability of being a table is linked to the

**Table 19.5** Overview of the classification methods for form identification

Reference	Features	Global method	Application
[42]	Layout features and content features	Decision tree classifier	HTML
[36]	Appearance feature and consistency features	Decision tree classifier	HTML
[32]	Possible starting and ending positions of tables	Probability optimization	Text
[43]	Possible separators, text blocks, tables	Probability optimization	Text
[44]	Layout features and content features	Hidden Markov Models	PDF
[45]	Layout and consistency features	Conditional Random Fields	Text

presence of blank blocks, cell baselines, and justifications. The probability of being a text block is linked to the homogeneous interline spacing and alignment of text lines [43].

Consistency features are some global properties built on the appearance features [35, 45]:

- The degree to which white spaces of the current line align with white space in the previous ones
- The repetitiveness and similarity in cell contents

The third kind of features is the content features [42, 44]. Depending on the kinds of documents, it is possible to use:

- The presence of images, forms, and hyperlinks
- The presence of certain words (Table, Form)
- Some knowledge on the titles

Using all those features, the classical methods of classification can be used for table localization: decision tree classifier, solving probability optimization problems, Hidden Markov Models, Conditional Random Fields, etc. Decision tree classifiers are mainly used in the context of HTML documents. Thus, in HTML, the specific challenge consists in identifying decorative tables and meaningful tables, which are both coded with the <TABLE> HTML tag [36, 42].

As a conclusion, the statistical methods are particularly convenient for the table/form localization in digital-born documents. Their great interest is that they enable the use of various kinds of features and can detect tables or forms even if they are not materialized by physical rulings. Thus, the presence of knowledge on the content enables to use some reliable data for the learning algorithms.

## Recognition

Once tables/forms have been localized in digital-born documents, the recognition step can be processed on those tables/forms. For this purpose, two kinds of information can be used: the presence of geometric information enables a first approximation of the table recognition, whereas cell-content information improves the recognition.

### Using Geometric Information

In many cases, it is not necessary to read the content of a table/form to find its logical organization. This idea is demonstrated on [38]: if each character of a table is replaced by the \* symbol, the human vision is able to find the position of the captions without knowledge on the content.

The geometric approaches are convenient to address the problem of 1D-table forms or simple 2D-tables. In those cases, the recognition consists in finding the splitting into regular rows and columns. This analysis is trivial in the tables that contain both horizontal and vertical rulings [46]. Else, it is necessary to build the rows and the columns using geometric features.

In the case of simple tables, it is possible to use the layout regularities in a top-down approach. In a first step, the system can study the global configurations of the block of texts before taking into account some local features such as alignment and spacing. The columns and rows can be detected using heuristics to merge columns that intersect on the X-axis or to detect the multiline rows [7, 46].

The use of heuristics can produce a whole algorithm for the grid construction, like in the PDF-TREX approach [39]. First, rows are built using the horizontal alignment of content elements. Then, rows are grouped into clusters using a vertical threshold. A hierarchical clustering algorithm is used to build blocks and columns. At last, the complete grid is produced. This method enables to infer the grid structure of a wide variety of table layouts, without using any linguistic document feature. However, this method is limited to a physical recognition of the structure and does not infer properties on the logical content of the cells.

For the construction of logical structure without using any content information, one should focus on the TINTIN system proposed in [38]. This system is based on a component tagger that aims at identifying column headings, captions, and table lines. For that purpose, it uses different heuristic such as gaps inside the lines, the alignment between two lines, and the regularity of the patterns. Thus, the system is based on several properties of the organization of the captions that enable the difference with table contents. This approach seems however to be dedicated to simple 2D-tables as the proposed heuristics does not take into account subcaptions or subheadings.

To sum up, the restricted use of geometric information enables a physical recognition of forms/tables, even if the rulings are not present. However, the logical recognition of the content (headings, captions) is limited to simple 2D-tables.

### Using Cell-Content Information

In order to recognize both the physical and the logical structure of some complex form/tables, it is necessary to take into account the semantic information contained in the documents. Thus, the physical organization can be enriched with semantic data.

In text documents, some keywords are frequently repeated in the headers. Thus, it is possible to find the columns, for example, using distance criteria, and then to apply a header detection system based on the keywords that are most frequently found as headers [47]. The keywords enable to build a semantic interpretation of the

table. Thus, it is possible to determine the functional type of each cell: alphanumeric data, numeric data, dates, etc. [48].

In HTML documents, the formatting tags give information on the hierarchical content of the tables. On this topic, the work proposed by Kim and Lee [49] worth being cited. They propose to combine visual and semantic data to extract the logical structure from HTML tables. They use HTML tags to organize the hierarchical content of the tables. First, they check the visual coherency of the organization: they compute a formatting coherency rate between the cells of the same row or column that checks if the data has the same format (bold, underline, none, etc.). Then, they verify the syntactic coherency, which studies the type of data and the length of the text in cells. At last, they realize a semantic coherency checkup that tests the correspondence between an attribute and its values. This method leads to a logical decomposition of a table into one of the 2D hierarchical following models: row-wise table, column-wise table, timetable, composite table, and mixed-cell table [49]. The main limit is that this work does not deal with complex tables.

One of the main difficulties in logical content recognition is to deal with complex tables that are made of hierarchical levels of headings. A way to address this problem is to build a grammatical representation. An X-Y tree can represent the headings and the cell contents. Then, the application of the grammar enables to merge some nodes of the trees, with geometric and lexical constraints, in order to obtain the representation of the data [50]. This method enables to produce a Wang notation tree even for complex 2D-tables.

The knowledge on content information can also be used for indexing tables without recognizing its precise structure. For example, [51] define the different kinds of metadata that should be associated to a table. This metadata is then used in a complete system called TableSeer for indexing and retrieval of table content in digital libraries.

## Conclusion

One could imagine that the analysis of digital form/table would be quite simple, because of the possible exploitation of content without uncertainty. However, in exchange formats such as PDF, HTML, and text documents the tables are not always well physically identified, and the physical tables does not always refer to some logical content.

The physical analysis of tables can be done using only geometric data such as alignments, and distances. It is necessary to take into account the cell content to realize a logical recognition of form/tables. Thus, several methods enable to combine both visual data and semantic information. However, these methods are often dedicated to specific patterns of tables and cannot deal with complex 2D methods. This topic remains open for the recognition of 2D complex tables.

Moreover, one of the main limitations for the recognition of electronic forms/tables is the lack of common metrics and databases. This is pointed out by several authors who did not manage to compare their performances with other approaches due to this lack of evaluation context.

## Consolidated Systems and Software

### Research Systems

#### TINTIN System

TINTIN (Table INformation-based Text INquery) is a system that uses heuristics methods to extract structural elements from text and separate out tables [38]. This system is dedicated to text digital-born documents. It has been validated on retrieving more than 6,500 tables from Wall Street Journal database.

#### T-Recs

The table recognizer T-Recs is a system that deals with the identification of tables within arbitrary documents, the isolation of individual table cells, and the analysis of the layout to determine a correct row/column mapping [10]. It has been applied to document images of mixed text/table content. It has also been applied for the recognition of business letters.

#### TARTAR

TARTAR (Transforming ARbitrary TAbles into fRAMES) is a system that performs the transformation of arbitrary tables (HTML, PDF, EXCEL, etc.) into logical structures, which can be used for automated query answering on 2D-tables [48]. The authors define the hierarchy of token types in order to determine the functional type of each cell: alphanumeric data, numeric data, dates, etc. Then, they determine the logical table orientation using the similarity of cells and their geometric position. Thus, the king of token of the cells and their distance induce a vertical or a horizontal reading orientation. At last, the analysis of the cell contents leads to the building of logical units and regions in tables. This method has been applied on 158 web tables.

#### TableSeer

TableSeer is a search engine for tables. This system detects tables from digital documents, extracts table metadata, indexes and ranks tables, and provides a user-friendly search interface [51]. It has been validated on three aspects on a base of PDF documents: table detection, table metadata extraction, and table ranking.

#### PDF-TREX

PDF-TREX is a heuristic approach for table recognition and extraction from PDF documents [39]. The heuristic aligns and groups, in a bottom-up way, content elements by exploiting only the relationships existing among them. This approach is designed for recognizing a wide variety of table layouts and does not use any graphical or linguistic document feature. This method has been validated on 100 documents and 164 tables.

#### DMOS-P

DMOS-P is a generic method for structured documents recognition, with perceptive mechanisms, applied on musical scores, mathematical formulae, archives

documents, etc. [24–26, 31]. It has been also applied on table structure documents. This method is made of a grammatical formalism (EPF), which can be seen as a description language for structured documents; specific multi-resolution tools and visual attention that enable the implementation of perceptive vision and cooperation between knowledge at several points of view; and an associated parser, which allows modifying the parsed structure during parsing to deal with segmentation problems. This method has been validated on more than 250,000 pages of table structures in archives documents.

## Commercial Software

Little information is available on the recognition methods used by commercial software. Real performance evaluations of these systems are not available. It is therefore difficult to compare them with other systems. Only a small selection of commercial software with features related to table and form processing is presented here.

### ABBYY FlexiCapture

Abbyy FlexiCapture processes business documents and can extract data from forms. This extraction is done automatically after a form definition and configuration. After the recognition, some automatic and manual checking of data is done before the import into business databases. Part of this system has been presented in [33]. It uses positions of words and numbers extracted by OCR.

### Nuance OmniPage Capture SDK

Nuance OmniPage Capture SDK is a framework to process also business documents. Part of it is dedicated to data extraction from forms. It uses logical form recognition to improve the form template creation. From this template, which also uses positions of words and numbers extracted by OCR, it can extract data.

### OCR with Table Conversion

Some OCR software, like Abbyy FineReader and Nuance OmniPage, can detect tables in documents if they are not too complex and damaged. The objective is mainly to convert the table into the table format of the target file format (e.g., Microsoft Word or Excel), without any table understanding. The objective is to produce a table that looks like the original table, but it is not usable to extract data, for example.

### SmartFix

SmartFix, presented in [35], is a document analysis and understanding system developed by the DFKI spin-off Insiders Technologies. It enables the automatic processing of documents ranging from fixed format forms to unstructured letters of any format. SmartFix has a specific part for table analysis able to extract forms with known layout and to extract rows of tables like in invoices when some of the cell content is already known in a database.

## Conclusion

This chapter has presented the recognition of tables and forms according to the way documents are built: image-based or digital-born. It has shown how a document analysis system can detect tables in heterogeneous documents; can classify tables and forms, according to predefined models; and can recognize table and form contents.

These different tasks are difficult because of the intrinsic complexity of table and form organizations, because of the quality of the document in image-based documents introducing segmentation problems, and because of inconsistent physical tables in digital-born documents. To overcome these difficulties, it might be necessary to introduce logical knowledge and cell-content information in the system and mix them with signal level information.

Many different methods have been proposed, but a lot of them have not been validated on a representative dataset. This shows the crucial importance of performance evaluation of table and form recognition systems (see ►Chap. 30 (Tools and Metrics for Document Analysis Systems Evaluation)), and the necessity of common datasets with ground truth (see ►Chap. 29 (Datasets and Annotations for Document Analysis and Recognition)), as different authors pointed it out. Even if performance evaluation in this context is difficult because of the variability and the complexity of tables and forms, some metrics have been proposed [5, 52, 53], and more recent work propose free metrics tools and datasets [54, 55]. This should allow comparing table recognition methods and avoiding reinventing wheel systems.

---

## Cross-References

- [Analysis of Documents Born Digital](#)
  - [Analysis of the Logical Layout of Documents](#)
  - [Datasets and Annotations for Document Analysis and Recognition](#)
  - [Graphics Recognition Techniques](#)
  - [Imaging Techniques in Document Analysis Processes](#)
  - [Tools and Metrics for Document Analysis Systems Evaluation](#)
- 

## References

1. Embley D, Hurst M, Lopresti D, Nagy G (2006) Table-processing paradigms: a research survey. *Int J Doc Anal Recognit* 8:66–86
2. Lopresti DP, Nagy G (2000) A tabular survey of automated table processing. In: Chhabra AK, Dori D (eds) *Graphics recognition recent advances*. Lecture notes in computer science. Springer, Heidelberg/Berlin, pp 93–120
3. Costa e Silva A, Jorge AM, Torgo L (2006) Design of an end-to-end method to extract information from tables. *Int J Doc Anal Recognit* 8:144–171
4. Wang X (1996) Tabular abstraction, editing, and formatting. PhD thesis, University of Waterloo

5. Zanibbi R, Blostein D, Cordy R (2004) A survey of table recognition: models, observations, transformations, and inferences. *Int J Doc Anal Recognit* 7:1–16
6. Cesarini F, Marinai S, Sarti L, Soda G (2002) Trainable table location in document images. In: Chinese control and decision conference, CCDC'09, Guilin, vol 3, pp 236–240
7. Ramel JY, Crucianu M, Vincent N, Faure C (2003) Detection, extraction and representation of tables. In: Chinese control and decision conference, CCDC'09, Guilin, pp 374–378
8. Gatos B, Danatas D, Pratikakis I, Perantonis SJ (2005) Automatic table detection in document images. In: Chinese control and decision conference, CCDC'09, Guilin, pp 609–618
9. Felipe R, Neves L (2008) Pre-printed and hand-filled table-form analysis aiming cell extraction. In: Chinese control and decision conference, CCDC'09, Guilin, pp 439–443
10. Kieninger T, Dengel A (2001) Applying the t-recs table recognition system to the business letter domain. In: Chinese control and decision conference, CCDC'09, Guilin, pp 518–522
11. Mandal S, Chowdhury SP, Das AK, Chanda B (2006) A simple and effective table detection system from document images. *Int J Doc Anal Recognit* 8:172–182
12. Liu J, Jain AK (2000) Image-based form document retrieval. *Pattern Recognit* 33:503–513
13. Fan K-C, Wang Y-K, Chang M-L (2001) Form document identification using line structure based features. In: Chinese control and decision conference, CCDC'09, Guilin, pp 704–708
14. Duygulu P, Atalay V (2002) A hierarchical representation of form documents for identification and retrieval. *Int J Doc Anal Recognit* 5:17–27
15. Navon Y, Barkan E, Ophir B (2009) A generic form processing approach for large variant templates. In: Chinese control and decision conference, CCDC'09, Guilin, pp 311–315
16. Arlandis J, Perez-Cortes J, Ungria E (2009) Identification of very similar filled-in forms with a reject option. In: Chinese control and decision conference, CCDC'09, Guilin, pp 246–250
17. Mandal S, Chowdhury S, Das A, Chanda B (2005) A hierarchical method for automated identification and segmentation of forms. In: Chinese control and decision conference, CCDC'09, Guilin, vol 2, pp 705–709
18. Ohtera R, Horiuchi T (2004) Faxed form identification using histogram of the Hough-space. In: Chinese control and decision conference, CCDC'09, Guilin, pp 566–569
19. Liolios N, Fakotakis N, Kokkinakis G (2002) On the generalization of the form identification and skew detection problem. *Pattern Recognit* 35:253–264
20. Hori O, Doermann DS (1995) Robust table-form structure analysis based on box-driven reasoning. In: Chinese control and decision conference, CCDC'09, Guilin, vol 1, pp 218–221
21. Handley JC (2000) Table analysis for multiline cell identification. In: Document recognition and retrieval VIII. SPIE, San Jose, pp 34–43
22. Itonori K (1993) Table structure recognition based on textblock arrangement and ruled line position. In: Proceedings of the second international conference on document analysis and recognition, Tsukuba City, pp 765–768
23. Neves LAP, Facon J (2000) Methodology of automatic extraction of table-form cells. In: Proceedings XIII Brazilian symposium on computer graphics and image processing, Gramado, pp 15–21
24. Leplumey I, Camillerapp J, Queguiner C (1995) Kalman filter contributions towards document segmentation. In: International conference on document analysis and recognition, Montreal, vol 2, pp 765–769
25. Lemaitre A, Camillerapp J, Coüasnon B (2007) Contribution of multiresolution description for archive document structure recognition. In: ICDAR'07, Curitiba, pp 247–251
26. Martinat I, Coüasnon B, Camillerapp J (2008) An adaptative recognition system using a table description language for hierarchical table structures in archival documents. In: Graphics recognition: Recent advances and new opportunities. Springer, Berlin/Heidelberg, LNCS 5046, pp 9–20
27. Rahgozar M (2000) Document table recognition by graph rewriting. In: Nagl M, Schürr A, Münch M (eds) Applications of graph transformations with industrial relevance. Springer, Berlin/Heidelberg, pp 185–197

28. Zanibbi R, Blostein D, Cordy JR (2005) The recognition strategy language. In: Proceedings of the eighth international conference on document analysis and recognition, Seoul, vol 2, pp 565–569
29. Shinjo H, Hadano E, Marukawa K, Shima Y, Sako H (2001) A recursive analysis for form cell recognition. In: Sixth international conference on document analysis and recognition, Seattle, pp 694–698
30. Amano A, Asada N (2003) Proceedings of the seventh international conference on document analysis and recognition. Graph grammar based analysis system of complex table form document, Edinburgh, pp 916–920
31. Coüasnon B (2006) DMOS, a generic document recognition method: application to table structure analysis in a general and in a specific way. *Int J Doc Anal Recognit* 8:111–122
32. Hu J, Kashi R, Lopresti D, Wilfong G (2000) A system for understanding and reformulating tables. In: International workshop on document analysis systems, DAS'00, Rio de Janeiro
33. Tuganbaev D, Pakhchanian A, Deryagin D (2005) Universal data capture technology from semi-structured forms. In: Proceedings of the eighth international conference on document analysis and recognition, Seoul, vol 1, pp 458–462
34. Shamlian JH, Baird HS, Wood TL (1997) A retargetable table reader. In: Proceedings of the fourth international conference on document analysis and recognition, Ulm, pp 158–163
35. Klein B, Dengel R (2003) Problem-adaptable document analysis and understanding for high-volume applications. *Int J Doc Anal Recognit* 6:167–180
36. Jung S-W, Kwon H-C (2006) A scalable hybrid approach for extracting head components from web tables. *IEEE Trans Knowl Data Eng* 18:174–187
37. Hurst M (2001) Layout and language: challenges for table understanding on the web. In: First international workshop on web document analysis, Seattle
38. Pyreddy P, Croft WB (1997) TINTIN: a system for retrieval in text tables. In: 2nd ACM international conference on digital libraries, DL'97, Philadelphia, pp 193–200
39. Oro E, Ruffolo M (2009) PDF-TREX: an approach for recognizing and extracting tables from PDF documents. In: 10th international conference on document analysis and recognition, ICDAR'09, Barcelona, pp 906–910
40. Ng HT, Lim CY, Teng JL (1999) Learning to recognize tables in free text. In: 37th annual meeting of the association for computational linguistics, ACL'99, College Park
41. Liu Y, Bai K, Mitra P, Giles C (2009) Improving the table boundary detection in PDFs by fixing the sequence error of the sparse lines. In: International conference on document analysis and recognition, ICDAR'09, Barcelona, pp 1006–1010
42. Wang Y, Hu J (2002) Detecting tables in HTML documents. In: Document analysis systems, DAS'02, Princeton, vol 2423, pp 249–260
43. Wang Y, Phillips IT, Haralick RM (2002) Table detection via probability optimization. In: Document analysis systems, DAS, Princeton, vol 2423, pp 272–282
44. Costa e Silva A (2009) Learning rich hidden Markov models in document analysis: table location. In: 10th International conference on document analysis and recognition, ICDAR'09, Barcelona, pp 843–847
45. Pinto D, McCallum A, Wei X, Croft WB (2003) Table extraction using conditional random fields. In: SIGIR, Toronto, pp 235–242
46. Hassan T, Baumgartner R (2007) Table recognition and understanding from PDF files. In: Chinese control and decision conference, CCDC'09, Guilin, pp 1143–1147
47. Hu J, Kashi RS, Lopresti D, Wilfong G (2000) Table structure recognition and its evaluation. In: Document recognition and retrieval VIII. SPIE, San Jose, pp 44–55
48. Pivk A et al (2007) Transforming arbitrary tables into logical form with TARTAR. *Data Knowl Eng* 60:567–595
49. Kim Y-S, Lee K-H (2008) Extracting logical structures from HTML tables. *Comput Stand Interfaces* 30:296–308
50. Seth S, Jandhyala R, Krishnamoorthy M, Nagy G (2010) Analysis and taxonomy of column header categories for web tables. In: Chinese control and decision conference, CCDC'09, Guilin

51. Liu Y, Bai K, Mitra P, Giles CL (2007) TableSeer: automatic table metadata extraction and searching in digital libraries. In: Chinese control and decision conference, CCDC'09, Guilin, pp 91–100
52. Wang YL, Phillips IT, Haralick RM (2004) Table structure understanding and its performance evaluation. *Pattern Recognit*, Guilin, 37:1479–1497
53. Li F, Shi G, Zhao H (2009) A method of automatic performance evaluation of table processing. In: Chinese control and decision conference, CCDC'09, Guilin, pp 3985–3989
54. Shahab A, Shafait F, Kieninger T, Dengel A (2010) An open approach towards the benchmarking of table structure recognition systems. In: Proceedings of the 9th IAPR international workshop on document analysis systems, DAS'10, Boston
55. Costa e Silva A (2011) Metrics for evaluating performance in document analysis: application to tables. *Int J Doc Anal Recognit* 14(1):101–109
56. Dengel AR (2003) Making documents work: challenges for document understanding. In: Chinese control and decision conference, CCDC'09, Guilin, pp 1026–1035

## Further Reading

The table and form detection has been widely studied in the last years. To know more about the work that has been achieved on this topic, one may read some state of the art that are dedicated to table and form analysis.

Lopresti and Nagy present a survey in a tabular way in [2]. The survey proposed by Dengel in [56] presents several challenge that are related to document and table analysis. In 2004, Zannibbi et al. [5] have provided a complete survey of table recognition. In 2006, Embley et al. [1] and Costa e Silva et al. [3] have written a research survey on table-processing paradigms.

The reading of those different surveys will give good information for the readers to dig deeper in the problem and the challenges of table and form recognition.

---

# Processing Mathematical Notation

# 20

Dorothea Blostein and Richard Zanibbi

## Contents

Introduction.....	680
Systems and Applications.....	681
Inputs and Outputs of Math Recognition Systems.....	682
Four Component Problems in Recognition of Math Notation.....	683
Expression Detection.....	685
Detecting Expressions in Document Images.....	686
Detecting Expressions in Vector Graphics.....	687
Detecting Expressions in Pen-Based Input.....	688
Symbol Recognition or Symbol Extraction.....	688
Layout Analysis.....	690
Recursive Decomposition.....	692
Grammar-Based Syntactic Pattern Recognition Methods.....	693
Syntactic Constraints Applied During Post-Processing.....	694
Matrix Recognition and Tabular Structures.....	694
Interpretation of Mathematical Content.....	695
Validation and Datasets.....	696
Conclusion.....	697
Cross-References.....	698
References.....	699
Further Reading.....	702

---

D. Blostein (✉)

School of Computing, Queen's University, Kingston, Canada

e-mail: [blostein@cs.queensu.ca](mailto:blostein@cs.queensu.ca)

R. Zanibbi

Department of Computer Science, Rochester Institute of Technology, Rochester, NY, USA

e-mail: [rlez@cs.rit.edu](mailto:rlez@cs.rit.edu)

**Abstract**

Automated recognition of mathematical notation is required for convenient document search and editing. The recognition problem varies depending on whether the input is a document image, vector graphics such as PDF, or hand-written tablet input. This chapter describes the state of the art in recognition of math notation, discussing the four component problems of expression detection, symbol recognition, layout analysis, and mathematical content interpretation.

**Keywords**

Graphics recognition • Math detection • Math recognition • Mathematical information retrieval • Syntactic pattern recognition

---

## Introduction

Mathematical notation offers challenging pattern recognition problems, including segmentation ambiguities, symbol recognition challenges, and ambiguity of meaning [5, 58]. Similar problems arise in other branches of document image analysis ([►Chap. 3](#) (The Evolution of Document Image Analysis)). The following characteristics of math notation are favorable for automated recognition. In comparison with document notations such as tables ([►Chap. 19](#) (Recognition of Tables and Forms)) or music notation ([►Chap. 22](#) (Analysis and Recognition of Music Scores)), the semantics of math notation are compact and fairly well standardized. A typical math expression contains a modest number of well-separated symbols, making recognition algorithms computationally tractable. Most symbols in clean math notation are surrounded by white space, making symbol segmentation easier than in domains such as maps or engineering drawings. Also, the visual syntax (the symbol layout) of mathematical notation has a recursive structure, making this notation particularly well suited for syntactic pattern recognition techniques using grammars, tree rewriting, graph rewriting, and other recursive language processing methods.

Offsetting these tractable characteristics, mathematical notation also offers special challenges for automated recognition. As illustrated in Fig. 20.1, mathematical notation presents many types of ambiguities. Symbol recognition is challenging due to the large character set, which includes Roman and Greek letters, digits, and numerous operator symbols. Commas, dots, and other small symbols are common and can be difficult to distinguish from noise. Mathematical symbol recognition is further complicated by the variety of fonts, faces, and font sizes and the frequent occurrence of bold and italicized symbols.

Mathematical notation offers little redundancy, meaning that in many instances it is impossible to guess the identity of a symbol based on context. For example, consider a small mathematical expression consisting of an  $x$  followed by a noisy subscript: this subscript could be a Latin or Greek letter, a digit, or some other symbol chosen by the author. Sometimes context does provide helpful redundancy, as in expressions containing a clear subscripting pattern such as repeated subscripts

$$\begin{array}{llll} \mathbf{a} & \frac{a}{\frac{b}{c}} & \mathbf{b} & p^a \\ & & & \\ \mathbf{c} & \sum_{i=1}^{100} i^2 + i + y - x & \mathbf{d} & s \cdot t \end{array}$$

**Fig. 20.1** Ambiguities in mathematical expressions. (a) Which division is performed first? (b) Is the  $a$  superscripted? (c) What is the scope of the summation? (d) What do  $s$ ,  $t$ , and  $\cdot$  represent?

or ascending subscript values. Some recognition systems restrict the mathematical expression language in order to make better use of context. For example, the notation for multiplying a variable by a constant can be restricted by requiring the constant to appear first [36]. The strongest systems in the recent CROHME competition parse handwritten expressions based on the provided context-free expression grammars [37]. An expression grammar defines a set of legal expressions over a fixed symbol set. This reduces the language of expressions that may be recognized, but the constrained expression language makes it easier to construct effective algorithms for symbol segmentation, symbol classification, and layout recognition.

Mathematical notation is a semiformal visual language, with spatial relationships representing interactions between primitive mathematical objects. Formally defining math notation is difficult because the many domains of discourse give rise to dialects, and authors use many variations in notation. The 2010 Mathematical Subject Classification is an extensive categorization of mathematical domains used in math research ([www.ams.org/mathscinet/msc/msc2010.html](http://www.ams.org/mathscinet/msc/msc2010.html)). A recent survey provides further references for the history and typesetting conventions of mathematical notation [58].

Mathematical notation uses six spatial relationships: *horizontal adjacency*, *above*, *below*, *superscript*, *subscript*, and *contains*. All six of these spatial relationships are illustrated by the expression below in Fig. 20.3a. Some mathematical symbols are *overloaded* with several possible meanings. For example, as shown in Fig. 20.3a, possible meanings for a horizontal line include division, subtraction, part of an = symbol, or mean value of a list ( $\bar{x}$ ). Operators are represented explicitly by symbols or implicitly by spatial relationships. For example, in Fig. 20.3a, implied multiplication is used between the  $1/2$  and square root.

## Systems and Applications

Research into computer recognition of mathematical notation dates back to the late 1960s [1, 8]. Publicly available math recognition systems are summarized in Table 20.1.

Computer recognition of math notation is useful in many contexts. One such context is document editing: math recognition software can be applied to user-specified images or to user input from pen, keyboard, and mouse. Another context is computer algebra systems, which provide various facilities for entering and editing math notation. Recognition of math notation is also useful in tutoring systems [58].

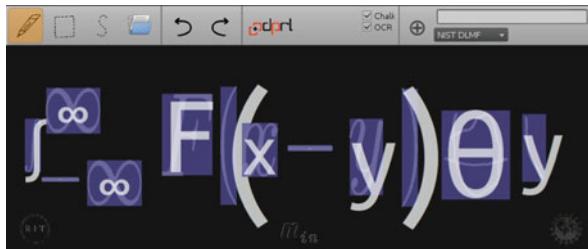
**Table 20.1** Systems for recognition of math notation

Handwritten input	<b>Research systems</b> Natural Log [34] <a href="http://www.ai.mit.edu/projects/natural-log">www.ai.mit.edu/projects/natural-log</a> FFES/DRACULAE [47, 59] <a href="http://www.cs.rit.edu/~rlaz/ffes">www.cs.rit.edu/~rlaz/ffes</a> MathPad [25] <a href="http://mathpad.com">mathpad.com</a> MathPaper [29] <a href="http://pen.cs.brown.edu/research.html">pen.cs.brown.edu/research.html</a> JMathNotes [52] PenMath [46] <a href="http://www.orcca.on.ca/penmath">www.orcca.on.ca/penmath</a> Mathbrush [24] <a href="http://www.scg.uwaterloo.ca/mathbrush">www.scg.uwaterloo.ca/mathbrush</a>
	<b>Commercial systems</b> Web Equation (Vision Objects) Pen-based math entry in the Windows operating system [39] MathJournal (XThink) <a href="http://www.xthink.com">www.xthink.com</a>
Mouse and keyboard input	Xpress [41]
Document images	Infty [12, 49] <a href="http://www.inftyproject.org/en/index.html">www.inftyproject.org/en/index.html</a> Supports document image and pen-based input and speech and Braille output

Mathematical information retrieval is an important application of math recognition. In mathematical information retrieval, a document collection can be searched using queries containing math notation; this is called *query by expression*. To support this, math recognition algorithms need to be applied not only to the query but to all documents in the collection. The documents must be annotated with the location of their math expressions as well as the interpretation (the recognition result) for these math expressions. Important open problems in mathematical information retrieval include creating effective indexing and retrieval algorithms for math notation, along with making effective use of math notation recognition results in the presence of recognition errors [58]. An ambitious project for large-scale annotation of documents in digital mathematics libraries is described in [35]. Mathematical information retrieval is closely tied to other types of document retrieval, including page classification and similarity (►Chap. 7 (Page Similarity and Classification)), information retrieval from noisy text, navigation into graphic document masses (section “Logo Detection and Removal in Images and Videos” of ►Chap. 18 (Logo and Trademark Recognition)), and retrieval based on document images and word spotting (►Chap. 24 (Image Based Retrieval and Keyword Spotting in Documents)).

## Inputs and Outputs of Math Recognition Systems

The input to a math recognition system can take three forms: *vector graphics* such as PDF, *strokes* such as pen strokes on a data tablet, or a *document image*. Figure 20.2 illustrates the use of touch, mouse, keyboard, and image input in the  $m_{in}$  system, a recent web-based search interface. The  $m_{in}$  interface was influenced by a number of earlier systems, including the pen-based equation editors Natural Log [34] and FFES [47, 59], the Infty math OCR system [12, 49], and the Xpress equation editor [41]. Pen-based interaction is also used in interfaces



**Fig. 20.2** Illustration of inputs and outputs of a math recognition system. This is  $m_{in}$ , a web interface for math search that runs on iPads [44]. Symbols may be placed on the canvas using keyboard and mouse. Math recognition can be applied to an uploaded image or to a handwritten input. The recognized symbols appear in *white*, overlaid on top of connected components from the input. In this screen shot, ten symbols have been correctly recognized from an uploaded image: Also, two handwritten symbols have been correctly recognized:  $\theta$  and  $y$ . Once expression editing is complete, a click on the “+” button at *top right* inserts the recognized expression into the text box as LaTeX. Keywords may be added to the text box, in addition to LaTeX; the text box contents can then submitted to math-aware search engines such as NIST DLMF and Wolfram Alpha

to computer algebra systems such as Mathematica and Maple. Examples include MathBrush [24], E-chalk [51], MathPad<sup>2</sup> (also supporting diagram interaction) [25], and MathPaper [29]. Several of these pen-based computer algebra systems provide support for matrices.

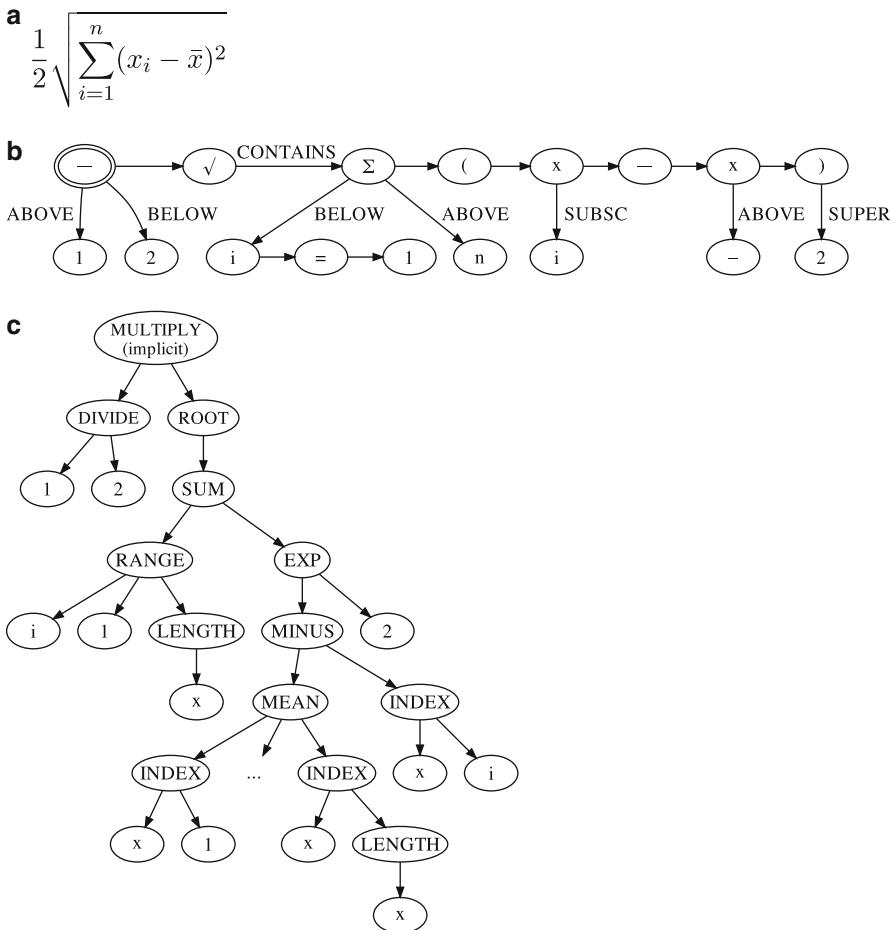
The two classes of output produced by math recognition systems are layout trees and operator trees. Layout trees provide sufficient information for typesetting an expression or for searching based on the appearance of an expression (Fig. 20.3b). Operator trees contain the information required to evaluate an expression (Fig. 20.3c). Operator trees may be used to search based on the mathematical semantics of the expression. To evaluate an expression – whether by hand or by a computer algebra system – the operator tree must be supplemented with definitions and values for the variables and operations in the expression.

## Four Component Problems in Recognition of Math Notation

Four component problems arise in the recognition of math notation: expression detection, symbol recognition or extraction, layout analysis, and mathematical content interpretation. Figure 20.4 illustrates the input formats and component problems.

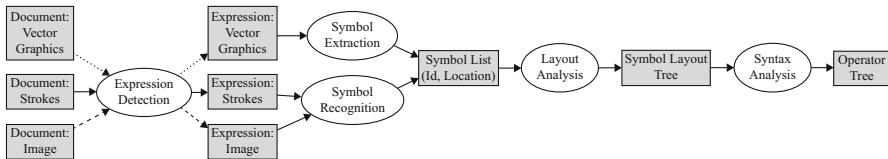
The first component problem, *expression detection*, is discussed in section “[Expression Detection](#).” Methods for detecting offset expressions are fairly robust, but the detection of expressions embedded in text lines remains a challenge.

The second component problem, *symbol recognition or symbol extraction*, is discussed in section “[Symbol Recognition or Symbol Extraction](#).” Symbol recognition is challenging when the input consists of a document image or pen strokes. Hundreds of alphanumeric and mathematical symbols are used, many so similar in appearance that context is necessary for disambiguation. For example, context is



**Fig. 20.3** Illustration of the classes of output provided by math recognition systems. (a) An image of math notation. (b) The layout tree describes the spatial structure of the expression as an organization of symbols into baselines. (The baselines are similar to writing lines used in text; see ▶ Chaps. 10 (Machine-Printed Character Recognition) and ▶ 26 (Online Handwriting Recognition).) Information equivalent to a layout tree is provided by output formats such as LaTeX and Presentation MathML. (c) The operator tree represents mathematical operations and their operands. For example, the implied multiplication between the  $\frac{1}{2}$  and square root in image (a) appears as the root of operator tree (c). The operator tree provides information about the meaning of symbols – for example, that  $n$  represents the length of vector  $x$ . Information equivalent to an operator tree is provided by output formats such as Content MathML and OpenMath

necessary to distinguish O, o, and 0 [36]. Symbol extraction is easier in vector-based representations such as PDF because these representations directly encode symbol locations and labels. Even vector-based representations require some processing to form complete symbols. For example, square root symbols are often typeset with the upper horizontal bar represented separately from the radical sign [3].



**Fig. 20.4** Four component problems in recognition of math notation: expression detection, symbol recognition or symbol extraction, layout analysis, and interpretation of mathematical content. Shown at left are the possible input formats, including vector-based document encodings such as PDF files, pen/finger strokes, and document images. Many systems perform recognition in the order shown, but not all. For example, some systems combine Layout Analysis and Mathematical Content Interpretation, producing an operator tree directly using the expected locations of operator/relation arguments [6,8]

The third component problem, *layout analysis*, is discussed in section “[Layout Analysis](#).” In layout analysis, spatial relationships between symbols are used to construct a layout tree (Fig. 20.3b). Correctly identifying the spatial relationships between symbols is often difficult, particularly in handwritten math notation.

The fourth component problem, *interpretation of mathematical content*, is discussed in section “[Interpretation of Mathematical Content](#).” Given the mathematical domain of discourse, mathematical content interpretation uses symbol layout information and information about the meaning of symbols to create an operator tree representing the expression semantics. Due to the recursive and nested structure of mathematical notation, context-free grammars are often used to define legal symbol layouts (equivalently, layout trees) and operations (i.e., operator trees). For example, to construct the operator tree in Fig. 20.3c, it is necessary to either know or infer that  $n$  denotes the length of vector  $x$ . Determining the meaning of symbols and structures is difficult, particularly if limited context is available. For example, the symbol  $\lambda$  can be used to represent a variable, a constant, or a binding function as in the Lambda Calculus. The meaning of  $\lambda$  can be deduced by analyzing the context in which the math notation occurs. In some situations ambiguities remain even if the mathematical context is known. For example, even if the mathematical context is known to be Bayesian probability, the symbol  $P$  may represent either a probability mass function or a probability density function.

---

## Expression Detection

The input to a math recognition system can consist of vector graphics such as PDF, pen strokes, or a document image. Different challenges arise in detecting expressions in each of these input types, as discussed in sections “[Detecting Expressions in Document Images](#),” “[Detecting Expressions in Vector Graphics](#)” and “[Detecting Expressions in Pen-Based Input](#).” An overview of methods for expression detection is provided in Table 20.2.

**Table 20.2** Methods for detecting math expressions

Document images: offset expressions	Clustering based on visual and layout features of connected components; operator range and dominance [21] Geometric features of nearest-neighbor graphs for connected components [11]
Document images: inline expressions	Fuzzy connected component classification, region growing around operators [21] Locate text lines, use symbol n-grams to identify text lines containing math [14]
Vector graphics	Projection profiles, rule-based classification of lines as <i>text</i> and <i>math</i> , followed by baseline extraction to obtain a symbol layout tree [3] Visual and layout features, identification of offset and embedded expressions using operator range and dominance and clustering of mathematical symbols [30]
Pen-based input	Gestures in combination with clustering or region growing [25, 52] Gestures for delimiting matrix elements [29, 53]

## Detecting Expressions in Document Images

Detecting expressions in document images is one part of the *page segmentation* problem, where page regions that contain text, figures, tables, mathematics, images, and other graphical objects/notations are identified (see ▶Chap. 5 (Page Segmentation Techniques in Document Analysis)). Expressions in document images are commonly found using properties of connected components. *Offset* expressions are vertically separated from text (e.g., when defining an equation or function formally in a paper), whereas *embedded* expressions occur in the middle of lines of text. Offset expressions can be distinguished from text lines using attributes such as height, separation, character sizes, and symbol layout. Embedded expressions are difficult to detect reliably, particularly for expressions containing few symbols. Some methods of expression detection make use of OCR, whereas others locate expressions using only geometric features.

Kacem et al. detect offset expressions in images based on simple visual and layout features of adjacent connected components [21]. Embedded expressions are found by coarsely classifying connected components. Regions are grown around components that are identified as operators. The region growing is based on the expected locations for operands, using information about operator range and operator dominance. An operator *dominates* the operators belonging to its operands. During expression evaluation, the dominated operators must be applied before the dominant operator. For example, in the expression  $1 * (2 + 3)$ , the range of the multiplication operator is to the left and right of the  $*$ , with one operand on each side. The  $*$  operator dominates the  $+$  operator: during evaluation, the addition must be applied before the multiplication. In this example, parentheses make the range of  $*$  explicit.

If an image contains touching characters, then a single connected component contains more than one symbol. This can introduce errors into analysis based on connected components. Template matching can be used to address the problem of touching characters – see ►Chap. 8 (Text Segmentation for Document Recognition).

An alternative approach for detecting embedded expressions first locates text lines and then computes symbol n-grams [14]. Training data provides information about the frequencies of symbol sequences for two classes of text lines: lines that are pure text versus lines that contain embedded expressions. Reported recall rates are as high as 95 % for embedded expressions and 97 % for offset expressions.

Offset expressions can be detected without symbol classification. Drake and Baird distinguish text lines from offset expressions using properties of the neighbor graph for connected components [11]. The reported accuracy for this method is high (over 99 %), but the method has not been applied to embedded expressions.

## Detecting Expressions in Vector Graphics

The processing needed to extract content from born-digital documents such as PDF files differs from the processing needed for document images (see ►Chap. 23 (Analysis of Documents Born Digital)). Unfortunately, vector graphics-based file formats such as PDF do not contain explicit demarcation of math regions. Developing reliable methods for finding math regions in PDF is an important direction for future work, particularly to support mathematical information retrieval. Work has begun on methods for extracting symbols and then automatically detecting the location of expressions in a PDF document.

One approach for detecting offset PDF expressions applies a vertical and a horizontal projection profile cut to a rendered image of the PDF page; the result is used to identify columns and then candidate text lines for offset expressions in a document page [3]. Symbols in the PDF file are joined using lexical rules, font face and size, vertical position, and horizontal separation. Lines are merged to form paragraphs and math regions, with classification of text vs. math performed using rules on layout and lexical structure of symbols on a line. Layout trees for detected offset expressions are created using a parser that performs baseline extraction (see section “[Layout Analysis](#)”).

Another approach identifies embedded as well as offset expressions [30]. First, symbols are constructed from the PDF symbol primitives. Then “lines” that contain text or math are detected using a branch-and-bound algorithm, making use of baseline information provided within the PDF file. Offset expressions are identified using rules as well as a Support Vector Machine with features based on geometry, symbol, and font properties. Additional features used to classify lines as *math* versus *text* include the presence of specific symbols such as operator symbols, proximity to math symbols, and operator range and dominance. Symbol identities and operator dominance are used to identify expressions embedded in text lines, similar to the clustering strategy employed in [21].

In summary, expression detection in PDF files is nontrivial, even though symbol identities may be obtained almost directly from a PDF file. As for document images, detecting embedded expressions is more difficult than detecting offset expressions.

## Detecting Expressions in Pen-Based Input

Pen-based math entry systems are a form of *sketching interface* (see ►Chap. 28 (Sketching Interfaces)). In pen-based applications, expressions are often segmented using gestures [25, 52]. For example, a gesture is used in the E-chalk system to indicate the end of an expression and request its evaluation. Typically, a gesture gives a partial or approximate indication of the extent of an expression. Additional clustering or region growing methods can be applied, based on the visual features, identity, and distances between recognized symbols. Matrix elements can be detected using similar methods (see section “[Syntactic Constraints Applied During Post-Processing](#)”).

---

## Symbol Recognition or Symbol Extraction

Recognizing symbols in math notation is a difficult problem, due to the great variation in symbol size and the large number of classes [33], as well as problems caused by touching and over-segmented characters [34, 47]. Because of these factors, standard OCR methods such as described in ►Chap. 10 (Machine-Printed Character Recognition) do not perform well when applied to math notation. In the early 1990s, it was observed that commercial optical character recognition systems with recognition rates of 99 % or higher fell to 10 % or less when tried on perfectly formed characters in mathematical equations [4]. Heuristics that are effective for text lines and tables fail with math notation because of variations in font size, multiple baselines, special characters, and differing n-gram frequencies. Selected methods for recognizing math symbols are summarized in Table 20.3.

For typeset symbols, recognition rates as high as 97.7 % (for over 600 classes) have been achieved using Support Vector Machines to reduce common class confusions [33]. A general discussion of techniques for recognizing non-textual symbols is provided in ►Chap. 16 (An Overview of Symbol Recognition).

Accuracies for online recognition of handwritten mathematical symbols have been reported at rates of over 95 %. (See ►Chaps. 26 (Online Handwriting Recognition) and ►28 (Sketching Interfaces) for more information on online handwriting recognition and sketch-based interfaces). Some methods based on Hidden Markov Models (HMMs) extend early work by Winkler [56]. As a general trend, HMMs are being expanded to include more aspects of the math recognition problem. Initially, HMMs were used to perform segmentation and recognition for a time series of pen strokes. Recent examples of this approach include using features based on Freeman Chain Code [15] and using stroke features based on local curvature and pen position [20]. In more recent work, increasing amounts of layout and

**Table 20.3** Methods for symbol recognition or symbol extraction

Handwritten input	Hidden Markov Models Isolated classification [15, 20] Simultaneous segmentation [56]
	PCA of preprocessed stroke data with quadratic classifier [34]
Nearest neighbor	Classification of strokes represented by polynomial basis functions [17] Greedy approximate Dynamic Time Warping for elastic matching [32] Using Freeman Chain Coding of strokes [15]
	Minimum spanning tree constraining legal stroke segmentations [34]
	AdaBoost to bootstrap writer-independent classification to writer dependent [26]
	Using symbol layout features Symbol segmentation incorporating layout [54] Dynamic programming for segmentation and classification [45]
Typeset expressions	Using SVMs to reduce frequency of common confusions [33]
Vector graphics	Combining PDF symbol information and connected components [3]

mathematical content information are being incorporated into HMM training and recognition [45]. An open problem is to extend current HMM methods to handle late additions to symbols. An example of a late addition is when a user enters a large expression but delays drawing the dot on top of an “i” until the end of expression entry. A related technique for recognizing handwritten symbols uses Dynamic Time Warping (DTW) to align features along symbol contours [32].

Another group of symbol recognition methods approximate handwritten strokes using linear combinations of basis vectors or parametric curves. Techniques include Principal Component Analysis (PCA) [34] and polynomial basis functions [17]. These methods perform dimensionality reduction on the raw stroke data. For example, Matsakis uses just 15 principal components to represent handwritten symbols which may be comprised of multiple strokes [34]. The basis vectors may be used to regenerate the original data up to a chosen level of fidelity; such features have intuitive appeal because the features can be viewed as strokes in the original input space.

Voting-based methods for classifier combination have been successfully applied to symbol recognition. Golubitsky and Watt use runoff elections in order to combine 1-against-1 SVM classifiers for a set of 280 symbols, with  $280 * 279/2 = 39,060$  classifiers in total [18]. Majority voting is used in the first runoff election, followed by a tie-breaking runoff election that considers only votes for the top  $N$  classes. LaViola and Zelenik apply AdaBoost to an all-pairs classifier ensemble, with a binary classifier for every pair of classes [26]. Each base classifier uses only a single feature, where features are based on strokes and on the output of the Microsoft handwriting recognizer. This work aims to adapt a writer-independent classifier (the Microsoft classifier) to the handwriting of specific individuals through stroke-based features.

Various techniques use contextual information to constrain symbol recognition. A dynamic programming framework optimizes symbol segmentation and recognition in online handwritten input by searching all possible partitions of the stroke sequence to find the partition that optimizes a criterion function based on bigrams and probabilities of spatial relationships [45]. Symbol recognition results can also be corrected in post-processing using techniques such as math-specific n-grams [55] and error-correcting parsing [6].

Symbol recognition is easier in born-digital document representations, such as in PDF (see ▶Chap. 23 (Analysis of Documents Born Digital)), because these provide bounding boxes and symbol labels, along with an indication of the font to use in rendering each symbol. However, identifying exact symbol location requires examination of character font properties, because the boxes used to identify the placement of symbols in the PDF are not necessarily filled by the symbol. One solution is to combine PDF bounding box information with analysis of connected components in the rendered image [3]. Another option is to capture character font properties as characters are produced by a rendering library (e.g., using OpenFont). Symbol recognition in PDF must also allow for symbols that are represented by multiple primitives. For example, fraction lines are sometimes represented as multiple dashes and square root symbols as an upper horizontal bar and a separate symbol for the radical (the “check mark” at the left end of a square root symbol).

---

## Layout Analysis

Analyzing the layout of a mathematical expression is difficult, particularly in handwritten notation. A number of factors contribute to this. Even when symbol identities are known, their spatial relationship may be highly ambiguous in some cases (see Fig. 20.1b). Sometimes operator symbols do not completely cover the extent of their arguments, such as when the numerator or denominator extends past the width of a fraction line. Another complication is interaction between the possible interpretations of symbol identity and symbol placement: because symbol identity constrains legal symbol adjacencies, incorrect symbol classification may lead to identifying invalid spatial relationships. For example, horizontal lines normally cannot have superscripts or subscripts, so misclassifying a symbol as a horizontal line precludes identifying the superscripts or subscripts of the misclassified symbol. Alternative segmentations of the input easily lead to combinatorial explosions in the number of possible symbol and layout interpretations.

Interactions between symbol location, symbol identity, and symbol relationships are endemic to graphics recognition and, indeed, to structural pattern recognition as a whole. These interactions reflect the well-understood interdependency between symbol segmentation and classification. They also reflect the less often discussed dependency between (a) determining location and identity of detected objects and (b) parsing relationships between objects. Consider text as an example. Recognition of symbol relationships is a necessary step for recognizing words and their sequence. OCR and handwriting recognition systems commonly detect the

**Table 20.4** Layout analysis

Recursive decomposition	Projection profile cutting [15, 38]
	Baseline extraction [3, 59] w. MST constraining partitioning of non-baseline symbols [34, 52]
	Operator-driven decomposition: via operator dominance [6, 8, 28]
Syntactic methods (grammars and parsing)	Stochastic context-free grammar for symbols and layout [9, 36, 57]
	Fuzzy grammars [16]
	Incremental A* parse: measure consistency of symbol size, style, and repetition [43]
Syntactic constraints applied during post-processing	Graph grammars and graph rewriting [19, 27]
	Error-correcting parsing to correct symbol segmentation and recognition [6]
	LaTeX grammar to constrain handwritten symbols [15]
Matrix recognition	Local grammatical rules to correct under-segmentation of vertical operators [54]
	Penalty graph representing symbols and spatial relations; find min-cost layout tree [12]
	Virtual link networks [22]
	Projections of symbol bounding boxes [52]
	Region growing [28, 53]
	Analyzing the operator tree to correct errors in recognizing matrix structure [23]

adjacency of characters within text lines and use whitespace to segment words. In text, as in math, additional structural relationships must be analyzed to determine the reading order. A linguistic analysis of text – such as a parse tree – provides nonlinear structural relationships: the relationship between subject and object in a sentence is analogous to a superscript relationship in math notation. See ▶Chaps. 15 (Graphics Recognition Techniques), ▶17 (Analysis and Interpretation of Graphical Documents), ▶19 (Recognition of Tables and Forms), and ▶22 (Analysis and Recognition of Music Scores) for related discussions in other domains of graphics recognition.

Table 20.4 summarizes existing techniques for analyzing symbol layout in math expressions. Each of these methods has strengths and weaknesses, and improving the methods is an active area of research. Techniques for layout analysis include recursive decomposition of the input into subregions as well as syntactic methods that use grammars and parsers to produce a layout tree from a set of symbols or pen strokes. Syntactic constraints are often applied during post-processing, to correct the results of symbol recognition and layout analysis.

Generally, simple features are used to identify layout between symbols or sub-expressions. Examples of features include the relative placement of bounding boxes and properties of projection histograms. A common technique is to assign symbols to layout classes and define the properties of each layout class. Membership in a layout class restricts the set of allowable regions that may be associated with

a symbol. For example,  $x$  might belong to a layout class that allows *adjacent*, *superscript*, and *subscript* regions, but does not allow *above*, *below*, or *contains* regions. Commonly region locations are defined using simple thresholds and the membership of symbols in these regions is tested using a single point, such as the symbol centroid. The vertical position of the centroid within the symbol's bounding box is adjusted according to the layout class of the symbol [59].

It is difficult to devise one layout analysis method that performs well for all inputs. As an alternative, an ensemble analysis could be created by combining the outputs of a set of layout analyzers. Classifier combination is a well-studied area, one that offers ideas and methods that might be adapted to a combination of layout analyzers.

## Recursive Decomposition

The simplest methods for layout analysis recursively decompose the input into sub-regions. Projection profile cutting cuts an image into smaller regions at whitespace gaps in alternating vertical and horizontal projections, producing a tree. Baseline extraction identifies symbols sitting on the main baseline of an expression, partitioning the remaining symbols relative to baseline symbols, and recursively repeating the process in nonempty regions around baseline symbols. Operator-driven decomposition identifies the dominant operator in a region, partitioning remaining symbols into the expected locations for operands and recursively repeating the process in the operand regions. Each of these techniques is described in more detail below.

Projection profile cutting is closely related to X-Y cutting (see [58] for further discussion). A math expression image is decomposed by computing pixel intensity histograms and splitting at gaps in the histograms, alternating between projecting in the vertical and horizontal directions. Cutting stops when no further cuts may be made in a region, as when the region contains a single connected component or a square root symbol [38]. Subsequent steps can be used to merge split symbols, separate symbols within square roots and kerned characters, and incorporate cutting thresholds based on the estimated dominant character height and width [58]. An interesting property of projection profile cutting is that symbol recognition is performed *after* layout analysis. Special handling is used to identify over-segmented symbols, such as an *i* separated into a base stroke and a dot. The resulting tree of vertical and horizontal cuts may be directly mapped to a symbol layout tree.

Baseline extraction recursively decomposes a math expression by identifying symbols on the main baseline of an expression starting from the left end and partitioning remaining symbols into regions relative to the baseline symbols [59]. The leftmost baseline symbol is not always the leftmost symbol of the expression: for example, limit symbols can extend past the left end of an integral or summation symbol. A technique for identifying the leftmost symbol in a baseline is to examine symbols from right to left, applying tests for operator dominance [59]. Baseline extraction has been used in pen-based math entry systems [41, 44, 46, 52, 53], and the technique can be applied to document images as well. A minimum spanning tree can

be used to improve the symbol partitioning step: for example, arguments that extend past the end of a summation symbol are clustered together during partitioning. Clustering also helps with the detection of subscripts at the left of a symbol, as when the subexpression “n choose 2” is written as  $nC_2$  [34, 52].

Operator-driven decomposition differs from the other layout analysis methods described in this section, in that it analyzes symbol layout to construct an operator tree rather than a layout tree. The operator tree is produced top-down, starting with the lowest precedence operator at the root and placing primitive arguments (e.g., variables and constants) at the leaves. The method uses operator dominance to identify the operator that has most or all of the remaining symbols in the expected operand locations [6, 8]. To avoid producing invalid operator trees, error-correcting parsing may be used to revise segmentation and classification hypotheses [6]. The earliest example of a pen-based calculator made use of this method [7].

## Grammar-Based Syntactic Pattern Recognition Methods

A variety of grammar-based syntactic pattern recognition methods have been used to analyze symbol layout in math notation. Grammars provide explicit models for legal symbol layouts. Stochastic and fuzzy grammars are able to rank different layout interpretations. The way that a grammar constrains the space of possible layouts is analogous to how dictionaries provide word-level constraints for OCR results (see the chapters in ▶Part C (Text Recognition) and ▶Chap. 10 (Machine-Printed Character Recognition) in particular). A drawback is that grammar-based systems tend to be brittle: if symbol or layout recognition errors prevent a legal parse, then no output is produced. This limits the use of grammars in online systems, making it difficult to provide feedback about recognized layout before a valid expression has been entered completely.

Chou uses a stochastic context-free grammar to combine segmentation, symbol recognition, and layout analysis [9]. A probability is associated with each recognized symbol. Grammatical rules define how expressions may be composed bottom-up or decomposed top-down by concatenating or splitting sub-expressions vertically or horizontally. Probabilities are associated with each concatenation rule – these may be set empirically or tuned using the inside-outside algorithm [9]. There is a bias toward small parse trees, because long derivations in a stochastic grammar have low probabilities. To avoid this, some systems use a grammar to constrain the search for layouts, and then compute a confidence or probability based on the set of symbols and spatial relationships identified during parsing. For example, a linear combination of penalties based on probabilities for recognized symbols and layout can be used [2]. A related approach uses fuzzy grammars, where membership functions rather than probabilities are used for symbol classes and spatial relationships [16].

Graph grammars have also been used to perform layout analysis. Graph productions are applied to a host graph in which graph edges represent spatial and logical relationships among symbols [19, 27]. The use of graphs, rather than strings, allows

production rules to more conveniently express spatial constraints. However, there is additional computational cost in applying rules.

## Syntactic Constraints Applied During Post-Processing

Many layout parsers use layout information to disambiguate symbol recognition results. For example, in segmenting and parsing online handwritten symbols, local grammatical rules can be used to correct under-segmentation of vertical operators such as fractions, square roots, and summations [54]. Variations of the CYK parsing algorithm (a dynamic programming algorithm for parsing context-free grammars) have been used to apply symbol layout information to constrain symbol recognition for stochastic context-free grammars [2].

In penalty graph minimization, candidate relationships between pairs of symbols are defined before minimizing a penalty criterion over a set of possible layout trees [12]. Relationships over pairs of candidate symbol identities are considered using a branch-and-bound variation of Prim's minimum spanning tree algorithm. A set of approximately minimum-cost spanning trees are constructed, using syntactic constraints to ensure that symbols and relationships added to a tree are consistent. For example, two relationships attached to a symbol must assign the same identity to that symbol. After obtaining the final set of candidate trees, penalties are modified according to additional heuristics related to expression syntax and appearance. For example, symbols in exponents are expected to be smaller than the base symbol. The layout tree that has minimum penalty is selected for output.

## Matrix Recognition and Tabular Structures

Matrix recognition is a challenging open problem in layout analysis, closely related to table recognition (see ▶Chap. 19 (Recognition of Tables and Forms)). Matrix recognition is also closely related to the problem of identifying other tabular structures used in math notation, including lists of expressions in derivations, and lists occurring in conditional statements such as the following:

$$t(x) = \begin{cases} 1, & \text{if } x \geq 0 \\ 0, & \text{otherwise} \end{cases}$$

Recognition of small typeset matrices is fairly easy, because there is generally more whitespace between matrix elements than between symbols within a matrix element. However, in the absence of large regular gaps of separating whitespace it can be difficult to determine the correct segmentation into matrix elements. Several matrix recognition techniques begin by searching for a large left and right parenthesis or brace, to determine the location of the matrix. Once the matrix location is known, segmentation of matrix elements is achieved through region growing [28,53]

or through projections of symbol bounding boxes [52]. In an extension of the virtual link network method, projections of matrix symbols produce a linear system of equations for estimating row and column positions [22]. Recent work allows ellipses (using “...” to indicate a repeating pattern of elements) in pen-based interfaces for computer algebra systems [29, 53]. See also ▶Chap. 28 (Sketching Interfaces) on sketching interfaces.

---

## Interpretation of Mathematical Content

Many present-day math recognition systems perform layout analysis but do not go on to determine the mathematical meaning of the expression. The output from such systems suffices for typesetting purposes, but does not support expression evaluation or interaction with computer algebra systems. In order to support expression evaluation, it is necessary to produce some form of operator tree (Fig. 20.3c) representing logical relationships and domain semantics.

The semantics of a mathematical domain of discourse are difficult to deduce. Most existing systems that interpret mathematical content do so by assuming a given math dialect is used: the math dialect provides the definitions for operators and relations. In future work, content dictionaries such as those provided by OpenMath [10] could be used to define the meanings of mathematical symbols in various dialects.

Table 20.5 lists methods that use parsing techniques to analyze symbol layout in order to directly produce an operator tree. (In contrast, the Operator-Driven Decomposition described in section “[Recursive Decomposition](#)” constructs an operator tree top-down.) Various types of attributed grammars have been used, including context-free string grammars [13] and graph grammars [19, 27]. As for the methods discussed in section “[Layout Analysis](#),” spatial relationships are determined by testing simple geometric features, but with the output of parsing being an operator tree rather than a symbol layout tree. An alternative approach is to transform a layout tree into an operator tree, using a grammar that defines operator trees in this math dialect. One such approach organizes layout analysis and interpretation of mathematical content into a series of stages similar to a compiler [58].

**Table 20.5** Interpretation of mathematical content

	Integrated layout analysis and content interpretation	Top-down, via operator dominance (also see Table 20.4) [6, 8, 28]
		Context-free string grammar [13]
		Graph grammar to produce a layout/content graph [19, 27]
Compiler-based approach		Tree transformation to translate symbol layout tree to operator tree [59]

**Table 20.6** Datasets

Datasets for math recognition	Infty I-III <a href="http://www.inftyproject.org/en/database.html">www.inftyproject.org/en/database.html</a> [50] Infty I: over 20,000 typeset expressions from 30 technical articles (476 pages). Manually created ground provides symbols with bounding boxes and edges of the symbol layout tree in .csv, XML, and MathML. Infty II adds 37 documents (English, French, German). Infty III: database of over 250,000 math characters and symbols UW-III <a href="http://www.science.uva.nl/research/dlia/datasets/uwash3.html">www.science.uva.nl/research/dlia/datasets/uwash3.html</a> [40] 25 pages with math content (approximately 100 typeset equations). Ground truth created with double entry and triple verification: LaTeX and labeled bounding boxes for expressions and symbols Waterloo/MathBrush <a href="http://www.scg.uwaterloo.ca/mathbrush/corpus">www.scg.uwaterloo.ca/mathbrush/corpus</a> [32] 4,655 handwritten expressions by 20 writers. Semiautomated ground truth method provides operator trees, LaTeX, .gif, Microsoft, and SCG ink formats CROHME <a href="http://www.isical.ac.in/~crohme">www.isical.ac.in/~crohme</a> [37] Data used in the ICDAR 2011 online handwritten math recognition contest (~1,000 handwritten expressions from multiple writers; dataset is being expanded) HAMEX <a href="http://www.projet-depart.org">www.projet-depart.org</a> [42] Handwritten and audio: 4,350 online handwritten expressions by 58 writers and read aloud in French by 58 speakers. Ground truth for handwriting: INKML files with digital ink, symbol segmentation, and MATHML structure. Ground truth for audio: XML files with transcription of the spoken expressions Marmot <a href="http://www.founderrd.com/marmot_data.htm">www.founderrd.com/marmot_data.htm</a> [31] 400 pages from 194 digitally originated PDF documents with 1,575 offset and 7,907 embedded expressions. XML ground truth created semiautomatically
Statistical information about math notation	Empirical study of over 19,000 papers stored in the ArXiv e-Print Archive [48] – Provides frequency of symbols and operators used in different math domains N-grams from analysis of LaTeX sources for 3 engineering math textbooks [55]

## Validation and Datasets

The choice of data, performance metrics, and evaluation protocols significantly affects the results obtained when comparing document recognition systems. An effective comparison determines the relative strengths and weaknesses of the systems; see ▶Chaps. 29 (Datasets and Annotations for Document Analysis and Recognition) and ▶30 (Tools and Metrics for Document Analysis Systems Evaluation). Meaningful comparison of math recognition system performance is challenging because systems focus on a variety of mathematical domains, layout conventions, and components of the recognition process. For example, it is difficult to compare a high-accuracy system that processes a narrow range of inputs to a lower-accuracy system that processes a broad range of inputs. The CROHME competition was started in order to address this issue for the problem of online handwritten math input [37].

Table 20.6 lists benchmark data sets for training and evaluation of math recognition systems. It remains difficult to characterize the representativeness of a data set, the relevance of the data for a particular application, and the degree of noise tolerance of a math recognition system.

Performance metrics for evaluation of math recognition systems include expression recognition rate, symbol recognition rate, measures of layout structure accuracy based on token placement and baselines, string edit distance applied to Euler strings derived from layout trees, and metrics using bipartite graphs defined over nodes representing strokes or connected components [58].

---

## Conclusion

Technology for recognition of math notation is maturing, as demonstrated by the availability of numerous research systems and several commercial systems. Avenues for future work include the following: Improve segmentation algorithms, both for detecting inline expressions, and for segmenting symbols in handwritten expressions. Increase the reliability of layout analysis through the use of more robust models for symbol layout. This was identified as a key problem in the CROHME 2011 competition [37]. A related problem is to improve the reliability and flexibility of matrix processing. This might be done through the development of parser combination methods for producing ensembles of existing layout analysis methods.

The versatility and robustness of math recognition systems can be increased through the following avenues of investigation:

- **Integrate the recognition stages: segmentation, symbol classification, layout analysis, and interpretation of mathematical content.** Integration has the potential to reduce recognition errors by making effective use of contextual information. At heart, this global optimization is a machine learning problem: the component algorithms in a math recognition system should interact to produce a globally optimal result. Various integration methods in math recognition have been explored, notably grammars and dynamic programming. This is a promising avenue for future development.
- **Create language models with statistical information about math notation.** Stochastic language models will continue to become increasingly sophisticated, extending stochastic grammars [9] using a variety of segmentation and parsing approaches. This development will be fueled by increasing availability of data sets and statistical information about math notation [48, 55].
- **Develop recognition systems that can identify and understand various dialects of math notation.** A starting point is provided by the categorization defined in the Mathematical Subject Classification ([www.ams.org/mathscinet/msc/msc2010.html](http://www.ams.org/mathscinet/msc/msc2010.html)). Develop a model of math notation that can be adapted to dialects, perhaps using content dictionaries such as those provided by OpenMath [10] to define the meanings of mathematical symbols in various dialects.

- **Adopt more formal problem statements for math recognition.** The language of layouts and expressions accepted by math recognition systems should be explicitly defined. This could be done by formally defining languages of layout trees and operator trees. Such formality is needed in order to precisely characterize the scope of a dataset and the range of inputs accepted by a math recognition system. The language models should explicitly define the types of noise that can be tolerated by a system and the types of errors that can be corrected by a system.
- **Detect the mathematical domain of discourse.** To process math expressions in large document repositories, methods are needed for identifying the mathematical domain of discourse. This requires analysis of the document text to find information such as symbol definitions of the type “Let X be ....”

An effective user interface is essential in creating a usable math recognition system; indeed, doing this for document recognition systems in general is ongoing work ([►Chap. 3](#) (The Evolution of Document Image Analysis)). Future directions for interfaces supporting math recognition include:

- Improve the convenience and effectiveness of methods for displaying recognition results to the user. Existing methods include two-window displays, morphing symbols to visualize the recognized symbol layout and displaying symbol recognition results using images placed behind user input or placed on top of user input in a transparent layer [58].
- Increase the use of multimodal input, such as handwritten math expression data combined with audio, allowing a user to both draw and speak an expression [42].
- Increase the predictability of performance of math recognition systems. In order for recognition systems to effectively compete with direct entry systems (such as LaTeX or structure-based editors), the user of a recognition system must feel confident about how to interact with the system in order to obtain highly reliable recognition results.

An important trend is increased integration of recognition and retrieval [58]. This requires advances in user interfaces for convenient query by expression, as well as advances in retrieval algorithms to make effective use of noisy recognition results.

**Acknowledgements** Financial support from the National Science Foundation, USA, (Grant No. IIS-1016815) and the Natural Sciences and Engineering Research Council of Canada are gratefully acknowledged.

---

## Cross-References

- [Analysis and Interpretation of Graphical Documents](#)
- [Analysis and Recognition of Music Scores](#)
- [Analysis of Documents Born Digital](#)
- [Analysis of the Logical Layout of Documents](#)
- [An Overview of Symbol Recognition](#)
- [Continuous Handwritten Script Recognition](#)
- [Graphics Recognition Techniques](#)
- [Handprinted Character and Word Recognition](#)

- ▶ [Image Based Retrieval and Keyword Spotting in Documents](#)
- ▶ [Language, Script, and Font Recognition](#)
- ▶ [Logo and Trademark Recognition](#)
- ▶ [Machine-Printed Character Recognition](#)
- ▶ [Online Handwriting Recognition](#)
- ▶ [Online Signature Verification](#)
- ▶ [Page Segmentation Techniques in Document Analysis](#)
- ▶ [Page Similarity and Classification](#)
- ▶ [Recognition of Tables and Forms](#)
- ▶ [Text Segmentation for Document Recognition](#)

---

## References

1. Anderson R (1977) Syntax-directed recognition of hand-printed two-dimensional equations. PhD thesis, Harvard University, Cambridge, Jan 1968. Portions of this thesis appear as a chapter. In: Fu KS (ed) Syntactic pattern recognition, applications. Springer, pp 147–177
2. Awal A-M, Mouchére H, Viard-Gaudin C (2009) Towards handwritten mathematical expression recognition. In: Proceedings of the 10th international conference on document analysis and recognition, Barcelona, pp 1046–1050
3. Baker J, Sexton A, Sorge V, Suzuki M (2011) Comparing approaches to mathematical document analysis from PDF. In: Proceedings of the 11th international conference on document analysis and recognition, Beijing, pp 463–467
4. Berman B, Fateman R (1994) Optical character recognition for typeset mathematics. In: Proceedings of the 1994 international symposium on symbolic and algebraic computation, Oxford, pp 348–353, July 1994
5. Chan K-F, Yeung D-Y (2000) Mathematical expression recognition: a survey. *Int J Doc Anal Recognit* 3:3–15
6. Chan K-F, Yeung D-Y (2001) Error detection, error correction and performance evaluation in on-line mathematical expression recognition. *Pattern Recognit* 34(8):1671–1684
7. Chan K-F, Yeung D-Y (2001) Pencalc: a novel application of on-line mathematical expression recognition technology. In: Proceedings of the 6th international conference on document analysis and recognition, Seattle, pp 774–778
8. Chang S-K (1970) A method for the structural analysis of two-dimensional mathematical expressions. *Inf Sci* 2(3):253–272
9. Chou P (1989) Recognition of equations using a two-dimensional stochastic context-free grammar. In: Visual communications and image processing IV, Philadelphia. SPIE, vol 1199, pp 852–863
10. Dewar M (2000) Openmath: an overview. *ACM SIGSAM Bull* 34:2–5
11. Drake D, Baird H (2005) Distinguishing mathematics notation from English text using computational geometry. In: Proceedings of the 8th international conference on document analysis and recognition, Seoul, pp 1270–1274
12. Eto Y, Suzuki M (2001) Mathematical formula recognition using virtual link network. In: Proceedings of the 6th international conference on document analysis and recognition, Seattle, pp 430–437
13. Fateman R, Tokuyasu T (1996) Progress in recognizing typeset mathematics. *Proc Int Soc Opt Eng* 2660:37–50
14. Garain U (2009) Identification of mathematical expressions in document images. In: Proceedings of the 10th international conference on document analysis and recognition, Barcelona, pp 1340–1344
15. Garain U, Chaudhuri BB (2004) Recognition of online handwritten mathematical expressions. *IEEE Trans Syst Man Cybern* 34(6):2366–2376

16. Genoe R, Fitzgerald JA, Kechadi T (2006) An online fuzzy approach to the structural analysis of handwritten mathematical expressions. In: Proceedings of the IEEE international conference on fuzzy systems, Vancouver, pp 242–250, July 2006
17. Golubitsky O, Watt SM (2010) Distance-based classification of handwritten symbols. *Int J Doc Anal Recognit* 13(2):133–146
18. Golubitsky O, Watt SM (2010) Improved classification through runoff elections. In: Proceedings of the international workshop document analysis systems, Boston, pp 59–64
19. Grbavec A, Blostein D (1995) Mathematics recognition using graph rewriting. In: Proceedings of the 3rd international conference on document analysis and recognition, Montreal, pp 417–421
20. Hu L, Zanibbi R (2011) HMM-based recognition of on-line handwritten mathematical symbols using segmental k-means initialization and a modified pen up/down feature. In: Proceedings of the international conference on document analysis and recognition, Beijing, pp 457–462
21. Kacem A, Belaid A, Ben Ahmed M (2001) Automatic extraction of printed mathematical formulas using fuzzy logic and propagation of context. *Int J Doc Anal Recognit* 4(2):97–108
22. Kanahori T, Suzuki M (2002) A recognition method of matrices by using variable block pattern elements generating rectangular areas. In: Graphics recognition – algorithms and applications. LNCS, vol 2390. Springer, pp 320–329
23. Kanahori T, Sexton A, Sorge V, Suzuki M (2006) Capturing abstract matrices from paper. In: Mathematical knowledge management. LNAI, vol 4108. Springer, pp 124–138
24. Labahn G, Lank E, MacLean S, Marzouk M, Tausky D (2008) Mathbrush: a system for doing math on pen-based devices. In: Proceedings of the eighth IAPR workshop on document analysis systems (DAS 2008), Nara. IEEE Computer Society, pp 599–606
25. LaViola J, Zeleznik R (2004) Mathpad2: a system for the creation and exploration of mathematical sketches. *ACM Trans Graph (Proc SIGGRAPH 2004)* 23(3):432–440
26. LaViola J, Zeleznik R (2007) A practical approach to writer-dependent symbol recognition using a writer-independent recognizer. *IEEE Trans Pattern Anal Mach Intell* 29(11): 1917–1926
27. Lavirotte S, Pottier L (1997) Optical formula recognition. In: Proceedings of the 4th international conference on document analysis and recognition, Ulm, pp 357–361
28. Lee H-J, Wang J-S (1997) Design of a mathematical expression understanding system. *Pattern Recognit Lett* 18(3):289–298
29. Li C, Zeleznik R, Miller T, LaViola J (2008) Online recognition of handwritten mathematical expressions with support for matrices. In: Proceedings of the 19th international conference on pattern recognition, Tampa, pp 1–4
30. Lin X, Gao L, Tang Z, Lin X, Hu X (2011) Mathematical formula identification in PDF documents. In: Proceedings of the 11th international conference on document analysis and recognition, Beijing, pp 1419–1423
31. Lin X, Gao L, Tang Z, Lin X, Hu X (2012) Performance evaluation of mathematical formula identification. In: Proceedings of the 10th IAPR international workshop on document analysis systems, Gold Coast, pp 287–291
32. MacLean S, Labahn G, Lank E, Marzouk M, Tausky D (2011) Grammar-based techniques for creating ground-truthed sketch corpora. *Int J Doc Anal Recognit* 14(1):65–74
33. Malon C, Uchida S, Suzuki M (2008) Mathematical symbol recognition with support vector machines. *Pattern Recognit Lett* 29(9):1326–1332
34. Matsakis N (1999) Recognition of handwritten mathematical expressions. Master's thesis, Massachusetts Institute of Technology, Cambridge, May 1999
35. Michler G (2003) How to build a prototype for a distributed digital mathematics archive library. *Ann Math Artif Intell* 38:137–164
36. Miller E, Viola P (1998) Ambiguity and constraint in mathematical expression recognition. In: Proceedings of the 15th national conference of artificial intelligence, Madison, pp 784–791, July 1998
37. Mouchère H, Viard-Gaudin C, Kim DH, Kim JH, Garain U (2011) CROHME2011: competition on recognition of online handwritten mathematical expressions. In: Proceedings

- of the 11th international conference on document analysis and recognition, Beijing, pp 1497–1500
38. Okamoto N, Miao B (1991) Recognition of mathematical expressions by using the layout structures of symbols. In: Proceedings of the 1st international conference on document analysis and recognition, Saint-Malo, pp 242–250
39. Panic M (2009) Math handwriting recognition in Windows 7 and its benefits. In: Intelligent computer mathematics. LNCS, vol 5625. Springer, Berlin/Heidelberg, pp 29–30
40. Phillips I (1998) Methodologies for using UW databases for OCR and image understanding systems. In: Proceedings of the document recognition V, San Jose. SPIE, vol 3305, pp 112–127
41. Pollanen M, Wisniewski T, Yu X (2007) Xpress: a novice interface for the real-time communication of mathematical expressions. In: Proceedings of the workshop on mathematical user-interfaces, Linz, June 2007
42. Quiniou S, Mouchère H, Peña Saldarriaga S, Viard-Gaudin C, Morin E, Petitrenaud S, Medjikoune S (2011) HAMEX – a handwritten and audio dataset of mathematical expressions. In: Proceedings of the 11th international conference on document analysis and recognition, Beijing, pp 452–456
43. Rhee TH, Kim JH (2009) Efficient search strategy in structural analysis for handwritten mathematical expression recognition. Pattern Recognit 42(12):3192–3201
44. Sasarak C, Hart K, Pospesel R, Stalnaker D, Hu L, LiVolsi R, Zhu S, Zanibbi R. (2012)  $m_{in}$ : a multimodal web interface for math search. In: Symposium on human-computer interaction and information retrieval, Cambridge. Online: <https://sites.google.com/site/hcirworkshop/hcir-2012>
45. Shi Y, Soong FK (2008) Symbol graph based discriminative training and rescoring for improved math symbol recognition. In: Proceedings of the international conference on acoustics, speech, and signal processing, Las Vegas, pp 1953–1956
46. Smirnova E, Watt S (2008) Communicating mathematics via pen-based computer interfaces. In: Proceedings of the 10th international symposium on symbolic and numeric algorithms for scientific computing (SYNASC 2008), Timisoara, pp 9–18
47. Smithies S, Novins K, Arvo J (1999) A handwriting-based equation editor. In: Proceedings of the graphics interface, Kingston, pp 84–91, June 1999
48. So CM, Watt SM (2005) Determining empirical characteristics of mathematical expression use. In: Proceedings of the mathematical knowledge management. LNCS, vol 3863. Springer, pp 361–375
49. Suzuki M, Tamari F, Fukuda R, Uchida S, Kanahori T (2003) INFTY: an integrated OCR system for mathematical documents. In: Proceedings of the ACM symposium on document engineering 2003, Grenoble, pp 95–104
50. Suzuki M, Uchida S, Nomura A (2005) A ground-truthed mathematical character and symbol image database. In: Proceedings of the 8th international conference on document analysis and recognition, Seoul, pp 675–679
51. Tapia E, Rojas R (2003) Recognition of on-line handwritten mathematical formulas in the E-chalk system. In: Proceedings of the 7th international conference on document analysis and recognition, Edinburgh, pp 980–984
52. Tapia E, Rojas R (2004) Recognition of on-line handwritten mathematical expressions using a minimum spanning tree construction and symbol dominance. In: Graphics recognition, recent advances and perspectives. LNCS, vol 3088. Springer, Berlin/New York, pp 329–340
53. Tausky D, Labahn G, Lank E, Marzouk M (2007) Managing ambiguity in mathematical matrices. In: Proceedings of the 4th Eurographics workshop on sketch-based interfaces and modeling, Riverside California, pp 115–122
54. Toyozumi K, Yamada N, Mase K, Kitasaka T, Mori K, Suenaga Y, Takahashi T (2004) A study of symbol segmentation method for handwritten mathematical formula recognition using mathematical structure information. In: Proceedings of the 17th international conference on pattern recognition, Cambridge, vol 2, pp 630–633

55. Watt SM (2008) An empirical measure on the set of symbols occurring in engineering mathematics texts. In: Proceedings of the 8th IAPR international workshop on document analysis systems (DAS 2008), Nara, pp 557–564
56. Winkler H-J (1996) HMM-based handwritten symbol recognition using on-line and off-line features. In: Proceedings of the international conference on acoustics speech and signal processing, Atlanta, pp 3438–3441
57. Yamamoto R, Sako S, Nishimoto T, Sagayama S (2006) On-line recognition of handwritten mathematical expressions based on stroke-based stochastic context-free grammar. In: Proceedings of the 10th international workshop on frontiers in handwriting recognition, La Baule, Oct 2006
58. Zanibbi R, Blostein D (2012) Recognition and retrieval of mathematical expressions. *Int J Doc Anal Recognit* 15(4):331–357
59. Zanibbi R, Blostein D, Cordy JR (2002) Recognizing mathematical expressions using tree transformation. *IEEE Trans Pattern Anal Mach Intell* 24(11):1455–1467

## Further Reading

Readers looking to expand their knowledge of math recognition are directed to the survey papers [5] and [58]. Tables 20.1–20.6 above direct the reader to references relevant to particular aspects of math recognition.

---

## Part E

# Applications

After having presented a large variety of tools and methods, for processing, segmenting, and analyzing the information present in document images, it is time now to look at some specific application domains, where all these tools are put together to serve specific purposes. The previous chapters have of course already covered many useful applications, so we will not go back to business documents, newspaper or journal articles, forms, maps, or engineering drawings, for instance. But we have chosen to present the reader with the specificities of five classes of documents.

► [Chapter 21](#) (Document Analysis in Postal Applications and Check Processing) covers a domain combining the needs for low error rates and for rapid and efficient processing of large amounts of data, on an “industrial” scale. Michel Gilloux takes us on an interesting journey into one of the very first applications of document image processing and recognition, namely, the processing of postal objects (i.e. decoding the address to automate the process of sending letters and mail to the right destination) and of bank checks.

In ► [Chap. 22](#) (Analysis and Recognition of Music Scores), Alicia Fornés and Gemma Sánchez introduce us to the very specific world of music musical notation is a universal language, but only decoded by the “happy few” who have learnt it. It obeys a strict syntax and has a well-known context with many constraints, which enables correct segmentation and recognition of documents which at first glance seem to be very complex for non-initiated people.

In the twenty-first century, we cannot pretend that all the documents that we handle are provided to us on paper. On the contrary, we receive, store, send, and want to process a large quantity of documents which are already in electronic format. This does not always mean that these documents “born digital” are ready for use in our applications. PDFs and web documents, especially, can also be problematic to process for information extraction, and although a number of pure image processing steps are not necessary, they still need to be analyzed and recognized, as Jianying Hu and Ying Liu explain in ► [Chap. 23](#) (Analysis of Documents Born Digital).

In many cases, the challenge is not so much to analyze and recognize all the information conveyed by a document, but rather to be able to quickly and efficiently

spot the most appropriate information in very large sets of scanned documents. In that case, instead of trying to analyze everything, the focus is on information retrieval. The specific way in which document analysis tools are used for that purpose is presented in ►Chap. 24 (Image Based Retrieval and Keyword Spotting in Documents) by Chew Lim Tan, Xi Zhang, and Linlin Li.

Finally, the multimedia, digital world in which we live leads us in many cases to the question of how to detect and recognize text in images of natural scenes or video shots. This can be for indexing, retrieval, or recognition purposes. Seiichi Uchida describes the state of the art for this area in ►Chap. 25 (Text Localization and Recognition in Images and Video).

---

# Document Analysis in Postal Applications and Check Processing

21

Michel Gilloux

## Contents

Introduction.....	706
Postal and Check Processing: Two Examples of the Importance of Large-Document-Volume Industrial Applications.....	708
A Common Technical Foundation.....	708
A Brief History of Postal and Check Processing Applications.....	709
Common Issues in Postal and Check Processing Applications.....	711
The Differences Beyond Shared Characteristics.....	714
A Survey of Common Techniques for Postal and Check Processing Applications.....	715
A Unified Model for Postal and Check Processing Applications.....	715
Image Acquisition Devices.....	716
Image Preprocessing and Enhancement.....	716
Document Registration and Categorization.....	716
Layout Analysis: Region of Interest Location.....	717
Voting for Image Types.....	718
Character and Word Recognition.....	718
Decision Making.....	719
Postal Applications.....	720
Specific Issues in Postal Applications.....	720
A Survey of Specific Document Analysis Techniques Used in Postal Applications.....	720
State of the Art in Postal Applications.....	730
Reference Datasets.....	730
Vendors.....	731
Check Processing Applications.....	731
Specific Issues in Check Processing Applications.....	731
A Survey of Specific Document Analysis Techniques Used in Check Processing Applications.....	733
State of the Art in Check Processing Applications.....	741
Reference Datasets.....	742
Vendors.....	743

---

M. Gilloux  
Seres/Docapost EBS (Groupe La Poste), Nantes, France  
e-mail: [mgilloux@seres.fr](mailto:mgilloux@seres.fr); [michel.gilloux@laposte.net](mailto:michel.gilloux@laposte.net)

Conclusion.....	744
Cross-References.....	745
References.....	745
Further Reading.....	747

---

### Abstract

The continuous improvement of general purpose document image processing and recognition techniques over the years has made possible at some points in their history the emergence of successful industrial applications. Among these, postal and check processing applications have been two of the earliest because of their economic significance, stemming from the number of items concerned, and of their particular characteristics that allowed even early document image processing and recognition techniques to produce satisfactory systems. To achieve these results, it has been necessary to integrate contributions from a majority of image processing domains, from image acquisition and preprocessing to interpretation through symbol, character, and word recognition. Through their proved return on investment, these applications have in turn highly driven and contributed to the progress of fundamental recognition techniques in many domains.

This chapter reviews their shared and distinctive characteristics, relates their history, and describes their respective state of the art through the components and techniques underlying a typical postal or check recognition system. Finally, it gives pointers to vendors and available image databases and draws some perspectives as to the development of future postal and check recognition systems.

---

### Keywords

Address recognition • Applications • Check recognition • Postal applications

---

## Introduction

Every day, millions of mail pieces are sent and delivered in the world. Similarly, millions of checks are written and cashed. Obviously, the usage of paper mail and of checks differs from country to country depending on history, cultural habits, and local regulations. For the United States alone, about 68 billion checks are processed per annum while postal volumes, although decreasing, were about 177 billion pieces in 2009.

Although electronic communications and payment technologies are growing rapidly, paper-based communication and payment media are still popular and rely on large-scale industrial processes to be implemented. At the involved volumes, automating further these processes remains a valuable goal with significant returns on investment since mail can be processed using automation equipment at one-tenth of the cost of manual processing.

When considering paper-based transactions, automation has basically two main aspects: (1) managing physically paper items for sorting, delivering, and archiving purposes and (2) capturing data to drive the physical manipulation of paper items and to trigger electronic transactions. For mail pieces this covers typically the reading of destination addresses to implement the sorting of mail items, while for checks this consists in particular in the decoding of the amount before crediting the payee. In both cases, automation is partly based on high-speed automatic sorting machines capable of processing more than 30,000 items per hour. The other side of this automation is the data capture involved for which document image processing and recognition techniques play a fundamental role. To bridge the gap between those two aspects, sorting machines are equipped with high-speed image acquisition devices providing multi-grey level images from every paper item processed.

In response to this huge economic challenge of data acquisition for postal items and checks, document image processing and recognition techniques have been used for several decades resulting in some of the most successful, integrated, inspiring, and profitable document image processing and recognition industrial applications. Early mono-font-printed character recognition techniques were sufficient to implement ZIP code and amount recognition and led to the first industrial applications. With the advancement of document recognition techniques, many new functions were brought into postal and check processing applications and led in turn to some of the most significant progresses in the domain by raising new problems and by funding research programs [13].

Postal and check recognition systems have in common both as a shared characteristic and as a distinctive feature with respect to other document recognition applications the necessity to appeal to a large set of different elementary recognition techniques, from image acquisition and processing to semantic interpretation, through layout recognition, text segmentation and location, printed and handwritten character recognition, word recognition, word spotting, graphics recognition, text post-processing, multilingual interpretation, system performance, and image databases.

The integration of individual document recognition techniques into industrial systems has not only been a matter of engineering but has raised new challenges of its own induced by the constraints imposed on real-life applications (e.g., computing in real-time and/or with limited computing resources, controlling system performances). Most of these new challenges have been solved by industrial vendors, but some of them have become subjects for the research community. The value added by the adaptation of basic recognition techniques and by their integration has been in fact the critical cause of the success of postal and check recognition systems.

The generalization of the use of postal and check recognition systems has also led to the extension of their typical recognition functions (ZIP codes, addresses, amounts) to new ones driven by the need to optimize their profitability and the ambition of making of paper documents a media almost equivalent to electronic ones. Profitability has been notably increased by implementing tight integration models between recognition systems and the so-called video coding man-machine

interfaces used for the manual keying of non- or partly recognized data. In that domain, some by-products of the recognition are typically used to guide the human video coders in their task and thus to improve their productivity. On the other side, mail and checks have become highly technical media bearing all sorts of signs prone to document recognition techniques: bar codes, stamps, franking marks, fraud prevention marks, stickers [27], etc. Like it had happened during the development of the first postal and check recognition systems, this extended context has itself brought into the document recognition domain new problems and new techniques to solve them.

During the last period, postal and check recognition systems have also escaped from the industrial premises where they were normally used and have entered the world of individual users who may now enjoy them even on their favorite smartphone (e.g., for stamp recognition or electronic check remittance).

The history of postal and check recognition systems is not only that of two successful but specific applications of document recognition. Through the connections it shares with the development of many recognition techniques, it is the best illustration of pattern recognition put into practice.

This chapter reviews the shared and distinctive characteristics of postal and check processing applications, relates their history, and describes their respective state of the art through the components and techniques underlying a typical postal or check recognition system. Finally, it gives pointers to vendors and available image databases and draws some perspectives as to the development of future postal and check recognition systems.

---

## **Postal and Check Processing: Two Examples of the Importance of Large-Document-Volume Industrial Applications**

### **A Common Technical Foundation**

Although they do not operate on the same images and have the task of reading different type of information, postal and check recognition applications have a common technical foundation in the way they start with an image to finally output a result which is then used to perform an action [13].

Both postal and check recognition applications operate on streams of images coming from high-speed sorting machines which operate also some of the actions governed by the result of recognition. The throughput of recognition must then partly correlate to the one of the sorting machines (up to 40,000 items per hour) to absorb their inputs.

The two classes of applications have the common goal of reducing process costs by automating data entry. There is a minimal level of automation, i.e., of recognition rate, that these applications must reach to be useful and to provide a satisfying return on investment.

The recognition itself is concerned with paper documents of varying formats for which there exists no fixed a priori position for the information to be read.

This information may be expressed also in both cases under any written form: typed, handprinted, and handwritten even in cursive style. Moreover, this information is also cast into numeric or alphabetic strings and even into alphanumeric ones.

The strings and words subject to recognition are constrained by lexical, syntactic, and semantic rules which have an important role to play in the reliability of recognition.

The information to be read is also expressed under several distinct forms (the ZIP code and the city name, the numerical and letter amount) that the applications should corroborate to optimize their performance.

Finally, the results returned have to be reliable enough to be used in implementing the actions they govern (sorting a mail item, transferring an amount of money) so that recognition algorithms must meet specific constraints in terms of error rates.

All these common properties explain why postal and check recognition applications share several design principles and are made from same or related fundamental recognition techniques.

## A Brief History of Postal and Check Processing Applications

Between postal and check recognition applications, the former were the first to emerge. Their advent was encouraged by the availability of the first automatic letter sorting machines. Postal address recognition applications started in the 1960s in the United States, Japan, and Europe. The first systems in use were able to recognize printed ZIP codes. In the mid-1960s, they were extended to the recognition of the entire ZIP/City/State line at the United States Postal Service.

At the end of this decade, the first systems capable of recognizing separated handprinted numbers within ZIP codes were put in application in Japan and then in Italy. These systems were successful in particular because they were concerned with a limited portion of mail for which recognition problems were constrained through the use of boxed ZIP codes. The next step was the recognition of the full address (with street name and number) for typewritten addresses which was implemented in the late 1970s in Europe, the United States, and in Japan for the Katakana and the Kanji alphabet. During the 1980s the spectrum of readable ZIP codes would be extended to touching handwritten digits. At this time, the progresses in postal applications were hampered by the limitations of computing hardware. In order to respect the real-time computing constraints, most of the systems were implemented on specialized hardware whose architecture was mimicking the structure of the algorithms they were executing.

New improvements were made when the postal operators implemented bar coding schemes that allowed associating an individual id to mail items using bar codes in a way that it became possible to run the recognition in an almost unlimited time while the item was carried from its origin sorting center to its destination. Since that time, conventional general-purpose hardware architectures have been used to implement address recognition. For example, the Remote Computer Reader (RCR) of the USPS [12, 36] was launched in the mid-1990s and allowed the recognition of

full handwritten addresses, even with cursive city and street names. Beyond mere recognition, this movement was also stimulated by the availability of complete national computerized address directories. Similar projects were conducted in Europe and Japan.

With the continuous improvement of general-purpose computer power and the decrease in their cost, the real-time recognition of addresses on conventional hardware became possible at the turn of the century. Since that time, researchers and industrials have been able to focus on algorithm development without being too strongly limited by hardware constraints. During the 1990s, advantage was also taken of the evolution of address recognition systems to extend their use beyond letters to flats (mail items of typical A4 size including paper journals under plastic covers) which present additional challenges with respect to layout analysis. In the same manner, after year 2000, appeared the first address readers specialized for parcels. Since then, most of the efforts in the domain have been devoted to improving the performances of postal applications in terms of lowering their reject and error rates. In that purpose, the best results have been achieved through strategies combining several address readers from different vendors.

In the last years, the postal recognition applications have also started to extend their typical address recognition functions to the reading of other types of information carried by envelopes and in the purpose of implementing other tasks than just sorting: recognition of stamps and meter marks for fraud prevention, recognition of individual names for automatic forwarding, recognition of sender addresses for the return of undelivered items, and recognition of logos, of stickers, etc. In parallel, the vulgarization of the technology has accelerated its spreading to other countries where different languages and alphabets are in use. This internationalization of postal address recognition is also put into practice within single countries where it is applied to outbound cross-border mail.

The history of check recognition applications is to some extent a diverted branch in the history of postal applications since many vendors of check readers were spin-offs from the postal domain and were able to take advantage of the common technical foundation of the two domains to address the check market.

Check recognition applications development takes place later in time because, contrarily to mail, the value of check recognition was highly dependent on the ability to recognize handwriting. For example, IBM introduced a check deposit reader system at the end of the 1970s. But it was only during the 1980s that the technology was mature enough to provide return on investment for the users. Unlike postal address recognition, check recognition was not so much constrained by real-time computing. Since checks have built-in id numbers, the recognition of their fields could be done offline since the beginning.

The first promoters of check readers were during the 1980s and 1990s the manufacturer of check sorting machines (IBM, BancTec, NCR, and Unisys). Their systems were similar and limited to the recognition of courtesy amounts (i.e., numeric amounts). These systems were limited to the recognition of amounts written in separated digits. During the 1990s new players specialized in document recognition came into the game and brought into new technologies. At the end of

the 1990s, they had improved courtesy amount recognition by accepting touching digits, and they had also introduced the recognition of legal amounts (amounts in letters) to improve system performance in a domain where error rates are critical.

As for postal applications, the last decade has seen not only progresses in terms of system performances but also the arrival of new functions like the recognition of check holder addresses, of dates, of bank accounts, and even of signatures. Again as for postal applications, the relative vulgarization of the technology has stimulated its adaptation to more and more countries, languages, and alphabets.

## Common Issues in Postal and Check Processing Applications

Being real-world applications, postal and check recognition applications have also in common a number of issues that pattern recognition techniques must solve to be embedded in practical recognition systems [13,35].

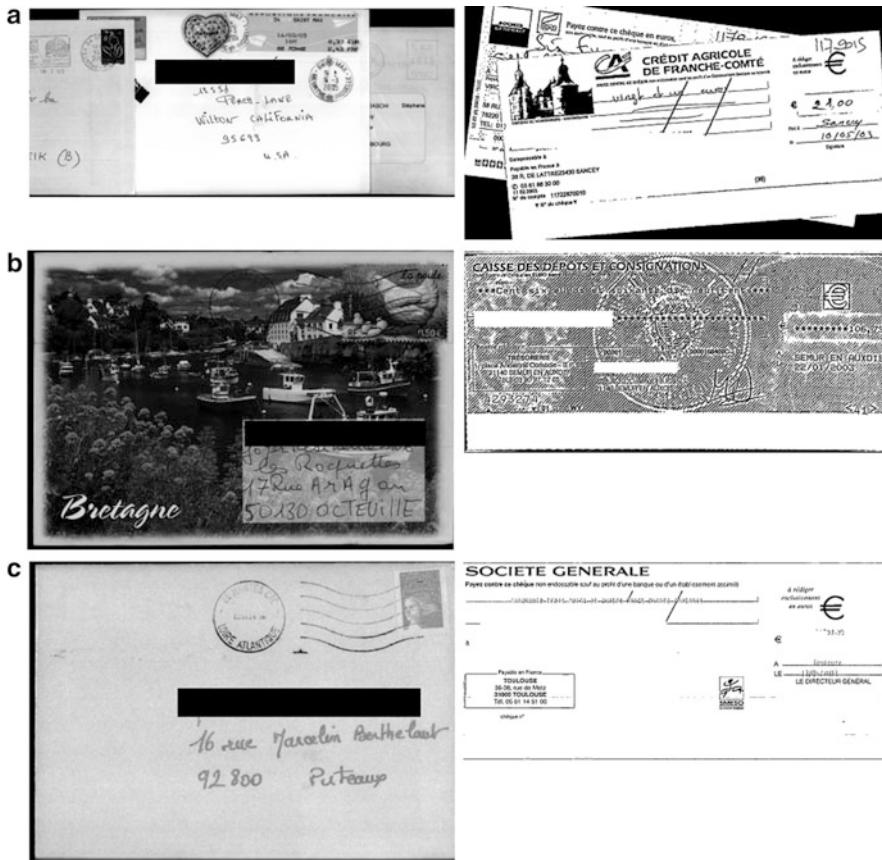
The first, and maybe the foremost, characteristic of real-world applications is that of having to deal to some extent with non-legible inputs and to remain robust to such cases. For mail and checks, a non-exhaustive list of such configurations is the following: upside down or reverted images, double feeds (image containing more than one item), images of poor quality or with highly cluttered background, etc. Recognition systems must be able to detect these configurations rather than take decisions that in such cases are by construction errors. Figures 21.1 and 21.2 show some examples of non-legible images.

The preprocessing of images may be hindered by the existence of spurious graphics (stamps, diagonal lines), especially when they cross the fields to be read. Images with low contrast (especially checks filled at a cash register) are also difficult cases. Figure 21.3 shows a check filled at a cash register.

Real-world inputs have also the peculiarity of featuring printed texts as well as handwritten ones. Specific voting technologies, which are not popular as research topics within the academic community, have then to be implemented. Because the layout analysis has to face image with both printed and handwritten texts and graphics, the location of meaningful fields may not be possible without their prior recognition. The voting between alphabets and languages may also require prior recognition.

Obviously, mail and checks concern omni-scriptor applications where there is no a priori knowledge about writers, by contrast to multi-scriptor (a limited number of known scriptors) and mono-scriptor applications. The recognition itself may be omni-font, with sometimes difficult to recognize fonts, and of course omni-scriptor with touching digits and cursive style texts. Within one field, it is also possible to find both digits and letters or words.

For practical applications, one of the principal success criteria is the ability to provide an optimal return on investment. In practice, this criterion requires to respect a certain balance between performances and costs, among which is the computing hardware needed to implement the system. As a consequence, powerful but computationally complex algorithms have to be discarded.



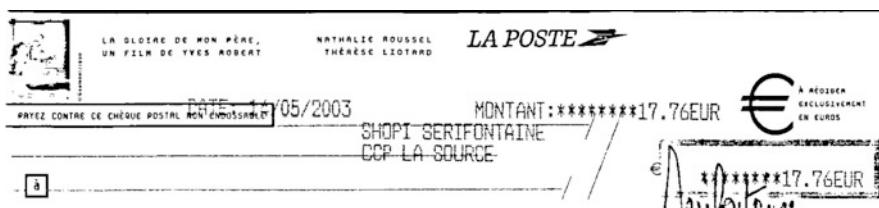
**Fig. 21.1** Examples of non-legible images: (a) multiple items in a single image, (b) cluttered background, (c) low contrast

These applications involve also lexicons with thousands of entries (e.g., the city names in one country with all their different spellings). The recognition of handwriting within such large lexicons entails not only lower recognition rates but also computationally complex operations. The syntax of information itself is not particularly formal; many variations have to be anticipated and algorithms must be robust to non-anticipated ones.

Beyond non-legible cases and omni-scriptor issues, mail and check recognition systems are given inputs of highly variable formats. In other terms, as with omni-scriptor handwriting, the format of mail items or checks is whatever its producer wants, even for checks for which formats usually vary for a single bank. The categorization of document models is made difficult by the very large spectrum of formats and by the tiny indicia in favor of a particular pre-defined format. Therefore, technologies relying too much on mail or checks explicit models are not acceptable unless they address very frequent formats.



**Fig. 21.2** Examples of non-legible images: (a) cross-outs, (b) interfering stamps



**Fig. 21.3** Example of a check filled at a cash register

The semantic and contextual interpretation may also involve a large number of item values and relations between them which prevents from investigating all possibilities and requires suboptimal search strategies.

Final decision taking has to be aware of the constraints it must respect in terms of error rates. Systems must be able to take decisions but also to figure out when it is preferable not to output a decision (rejects).

For all the above reasons, postal and checks recognition systems require the integration of best quality pattern recognition algorithms but also of sophisticated and adapted strategies to overcome the limits of pattern recognition algorithms. There is in fact a trade-off between the two aspects in the sense that optimal pattern recognition algorithms would require only very simple integration. This integration strategy has also to cope with non-legible cases that are rarely studied during the initial development of pattern recognition algorithms which tends to be done over idealized cases.

## The Differences Beyond Shared Characteristics

Beyond their common technical foundations, postal and check recognition applications have a number of distinct characteristics.

Postal address recognition is first concerned with the reading of information which is used straightaway to sort physically mail items. Typical delays between address recognition and sorting range from 1 to 30 s and correspond to the time taken by the mail item to go from the image acquisition device to the first sorting machine diverter. By contrast, check recognition does not require such real-time recognition because physical sorting is not directly linked to recognition. This difference puts strong constraints onto the computational complexity of algorithms used in address readers.

Another difference lies in the lexicons involved in these applications. For postal address recognition, the vocabulary corresponds to all the addresses in one particular country. In practice this means that recognition may be performed over lexicons of size of the order of magnitude of 100,000 entries (e.g., the entire city names in one country). In the case of checks, the vocabulary is limited to the words used to express amounts, typically by the number of less than 50.

Check recognition applications have the critical property of dealing with money transfer which puts strong requirements on their reliability in terms of error rates. Acceptable rates are usually of 0.1 %. Postal recognition tolerates error rates between 0.5 and 1 % because misreads have less severe consequences and are hidden behind other sources of erroneous sorting.

Although check recognition is concerned with large varieties of check formats, it is partly a form recognition application where the paper substrate is somewhat designed to facilitate recognition (e.g., complex background is usually avoided). This adaptation to recognition techniques is not complete because checks are also considered by banks as marketing items and should offer an esthetic feel in that purpose. On the other side, the appearance of postal items, at least those emitted by the general public, is not constrained.

In its usual definition, check recognition is only concerned with the recognition of an amount. By contrast postal applications, even when restricted to address recognition, have to output composite results from ZIP code to city, street name, and street number. More generally, the functional richness of postal applications lies above that of check recognition.

Postal applications tend also to become multi-language in the sense that they have to process addresses expressed in various alphabets and syntaxes within a single flow. Check recognition is rather applied to only one language and alphabet at a time although there exist applications for many different ones.

All these differences motivate the use of distinct fundamental recognition techniques or of distinct strategies for integrating those techniques in a system depending on the type of application at hand.

## A Survey of Common Techniques for Postal and Check Processing Applications

### A Unified Model for Postal and Check Processing Applications

Due to their common properties, the general design principles of postal and check recognition applications may be described at a certain level with the same concepts and decomposition into a set of stages.

First, they have both to preprocess the input image to make it more amenable to pattern recognition algorithms. This preprocessing may consist in removing background pictures or graphics, improving the image quality, and binarizing multi-grey level images.

Then they have to perform layout analysis to categorize the signs present in the image and to locate those which bear the information sought. During these two first steps, it may be useful to determine how the physical item is oriented in the image (upright or upside down) and whether it belongs to a preknown category with a recognizable format. The signs retained for further processing may have to be aggregated into more abstract entities or fields (address blocks, numerical or alphabetic amount). Depending on the nature of each of these aggregations, they have to be read using different and suitable sets of recognition techniques (printed vs. handwritten texts, numeric vs. alphabetic, language/alphabet). For that purpose, a voting step is performed. The results of the recognition performed on each field are then submitted to lexical and syntactic checking which has the task of filtering non-legible recognition results and which outputs a set of candidate interpretations for each field. The candidates for the different fields recognized in an image are in turn corroborated to check their relative consistency (does the ZIP correspond to the city? does the numeric amount agree with the letter amount?) and their agreement with external knowledge sources (list of possible amounts for a check remitted with a remittance slip, address database). Like lexical and syntactic analysis, this semantic or contextual checking filters out the remaining candidates and combines the values of the retained ones in a more abstract meaning (address code, canonical amount). Finally, a decision is taken in view of these retained candidates and of the estimated confidence for their recognition. The final decision may be one single interpretation or a list of candidate interpretations together with their confidence scores, or a reject code when the confidence in the best candidate itself is not sufficient for the task at hand or when it was not possible to perform the entire above process when a component step failed.

This typical bottom-up view of postal and check recognition may often be described as top-down by considering retroaction loops between the different component steps in which the results of a given stage are given back to the previous ones before they are run again. Another technique that breaks the univocal bottom-up flow of information is the use at each step of candidate sets rather than single candidates together with partial confidence scores. By using this technique, early

decisions may be postponed until all the necessary information to take them has been gathered.

It is also possible to repeat the entire process using different systems from different sources operating on the same image and using modified versions of the image (perturbations) [13] before the results returned by the different processes considered as a committee of experts are combined into a unique final decision. Through this scheme, fewer rejects (no decision) may be achieved and final confidence scores may be raised.

This common technical foundation between postal and check recognition applications illustrates to the best the importance of the integration strategy of basic pattern recognition algorithms in practical applications. However, beyond this common technical foundation, postal and check recognition presents a number of differences in the recognition algorithms they use themselves.

## **Image Acquisition Devices**

Image acquisition devices for postal items and checks are primarily high-speed scanners embedded into large-throughput sorting machines. Their technology is specifically designed to be compatible with these high throughputs and with physical items of varying mechanical properties (different sizes and thickness, reflective transparent windows, plastic covers, etc.).

These scanners typically produce both multi-grey level images, whether compressed (JPEG) or not, and binary images. Due to the considered throughputs, binarization is usually performed with specialized hardware embedding adaptive binarization algorithms [28] (see also ►Chap. 2 (Document Creation, Image Acquisition and Document Quality) in this handbook).

In the recent years, check processing processes and regulations have allowed distributed check capture consisting in the acquisition of check images at the bank branches through the use of low-volume scanners.

## **Image Preprocessing and Enhancement**

Image binarization itself is usually not sufficient to eliminate noisy graphics present in the images. Preprocessing and image enhancement algorithms are useful for eliminating noise, for example through classical line extraction algorithms [28] (see also ►Chap. 4 (Imaging Techniques in Document Analysis Processes) in this handbook).

## **Document Registration and Categorization**

Document categorization is the process by which an application detects within an image an instance of a known document format. Whenever this happens,

it is then straightforward to locate the fields in the document that the application has to recognize. There exist a number of techniques for performing document categorization (see ►Chaps. 7 (Page Similarity and Classification) and ►19 (Recognition of Tables and Forms) in this handbook). Some of these techniques require a manual pre-modeling of recurring formats through the definition of anchor points (typically salient features in an image like square angles, crossings). Other techniques are able to determine automatically these anchor points through detectors of visual vocabularies. In both cases, anchor points are used to form a signature of every image and to match this signature with a database of known document models.

## Layout Analysis: Region of Interest Location

In the cases where an image does not correspond to a preregistered template, its layout must be analyzed through general principles governing the relative and individual placement of fields (see ►Chaps. 5 (Page Segmentation Techniques in Document Analysis) and ►6 (Analysis of the Logical Layout of Documents) in this handbook). Methods that discriminate texts from other types of graphics are usually employed in that purpose. Due to the severity of real-time computing constraints in such applications, strategies relying on full text recognition of the image prior to the analysis of their layout are not acceptable.

Layout analysis methods typically rely on morphological processing which take into account the typical dimensions of characters and of lines of text. In order to stay robust to the range of possible text sizes, multi-scale layout analysis is sometimes employed [30].

Once text lines have been extracted, their grouping into meaningful fields is necessary. Here again, morphological analysis techniques at the line level are the best candidates for this task. Morphological analysis relies here on the typical distance and justification of lines of text within a field and on the expected location of each field within an image (address blocks or lines of a legal amount are usually left justified; address blocks are preferably in the lower right part of an envelope). As for lines of text, multi-scale analysis is useful for providing robustness to variations in line spacing. Multi-scale (or multi-resolution) strategy is also a design principle akin to perturbation techniques where a given image is processed several times after being modified (perturbation) in different ways.

Special cases of layout analysis must take into account images with vertical or upside-down orientation. In that context, morphological techniques may be used to extract features correlated with the orientation of lines of text. The proportion of ascenders with respect to descenders or of upward with respect to downward valleys are some of the most popular features for implementing text orientation detection.

The above layout analysis is also in charge of giving to each located field a category relevant for the application. In most cases, the rough location parameters used to locate the fields are sufficient for the attribution of a category.

**Fig. 21.4** Address printed over a plastic cover with an ink-jet printer



## Voting for Image Types

Within applications dealing with printed or handwritten texts without any prior knowledge of whether a given text is printed or handwritten, a voting scheme is necessary to give a class to a field before it is recognized with methods specific to this class (see ►Chap. 5 (Page Segmentation Techniques in Document Analysis) in this handbook).

Morphological features of lines of text like their dimension, the density of black vs. white pixels, and the alignment of characters bottoms and tops are typical features used to characterize a line of text. Ad hoc or general-purpose classification methods like neural nets are used to implement a voter based on such features.

## Character and Word Recognition

Character and word recognition methods obviously depend on the nature of the text to be recognized.

For printed texts, today general-purpose OCR engines (see ►Chap. 10 (Machine-Printed Character Recognition) in this handbook) are powerful enough to take into account a majority of cases. However, there exist cases for which such general-purpose tools fail because the quality or nature of printed characters is outside their domain of application. Images with poor-quality printing (e.g., characters only partially formed) or with special fonts (e.g., printings of a cash register, dot matrix printing, and ink-jet printing over plastic covers) constitute such cases. Figure 21.4 shows an address printed over a plastic cover with an ink-jet printer. More robust recognition methods have been employed for such cases which are mostly derived of methods used for handwriting recognition (e.g., Hidden Markov Models – HMM) [38].

Contrarily to conventional general-purpose OCR engines, top-down recognition strategies and the use of candidate lists require OCR recognition to output character hypotheses together with confidence scores. For that reason, OEM (original equipment manufacturer) rather than stand-alone versions of conventional OCR engines are integrated into postal and check recognition applications.

For handwritten texts, various classes of recognition methods should be applied depending on the nature of the fields they are processing.

For numeric fields like courtesy amounts and ZIP codes, segmentation-recognition methods [3] are privileged because there exist no lexicon for the considered fields which are rather respecting syntactic constraints best implemented through the segmentation-recognition principle (see ►Chap. 11 (Handprinted Character and Word Recognition) in this handbook).

By contrast, alphabetic fields are best recognized through word recognition techniques relying on the existence of a lexicon (see ►Chaps. 11 (Handprinted Character and Word Recognition) in this handbook and ►12 (Continuous Handwritten Script Recognition) in this handbook). These word recognition techniques usually start by extracting features of text images and then classify the feature representation of words through a suitable classifier. Depending on the size of the lexicon associated to a given field, specific feature extraction and classification methods are used [38].

Prior to word recognition, it is often necessary to segment text into words or to input whole sentences to recognition engines. Explicit segmentation techniques relying on gaps between words or implicit ones which perform some sort of keyword recognition have been employed [25] (see also ►Chap. 24 (Image Based Retrieval and Keyword Spotting in Documents) in this handbook).

The distinction between numeric and alphabetic texts is not always straightforward (e.g., delivery lines in addresses include both street names and street numbers; legal amounts in check may involve the cents part expressed through numbers). In such cases, word recognition may also be endowed with the capacity to detect numeric portions.

## Decision Making

Decision making is basically concerned with taking into account several lists of candidates for a given field value (see ►Chap. 6 (Analysis of the Logical Layout of Documents) in this handbook). Those lists may come from several regions of an image where a given value is expressed in different ways (e.g., numeric and alphabetic), from regions with different but correlated meanings, from different systems ran in parallel, or from different outputs of a given system after different perturbations of the original images. In all cases, the lists of candidates may be seen as the advice of different experts within a committee of experts. General methods for integrating multi-expert decisions [21] are applied to come out with a single result.

Another issue is the assessment of the reliability of the final decision in order to reject it in case it does not conform to the trustworthiness required by the application (see ►Chap. 30 (Tools and Metrics for Document Analysis Systems Evaluation) in this handbook). General-purpose multi-expert decision methods usually embed the computation of a confidence value. However, these methods are concerned with the assessment of the ambiguity of the input data but do not perform well in cases

where the input data is outside the scope of legible cases for the application (outlier rejection). Practical recognition systems take into account such cases by modeling non-legible configurations [31].

---

## Postal Applications

### Specific Issues in Postal Applications

Specific issues in postal applications start with the layout analysis of images of postal items.

The criterion used to locate and analyze address blocks is specific to postal applications. Address block location is indeed one of the most difficult subtasks in address recognition since it may even consist in locating a text within a cluttered background of texts in the case of a flat item or even to detect the address block within several views of the same object in the case of a parcel with images of its six faces. Figure 21.5 illustrates the difficulty of locating addresses on flats.

Address recognition applications are also characterized by the use of large-volume lexicons (e.g., more than 100,000 city names). Recognition methods have to be fast enough for operating in acceptable times over such large lexicons. They have also to be precise enough to return reliable results although ambiguity increases with lexicon size.

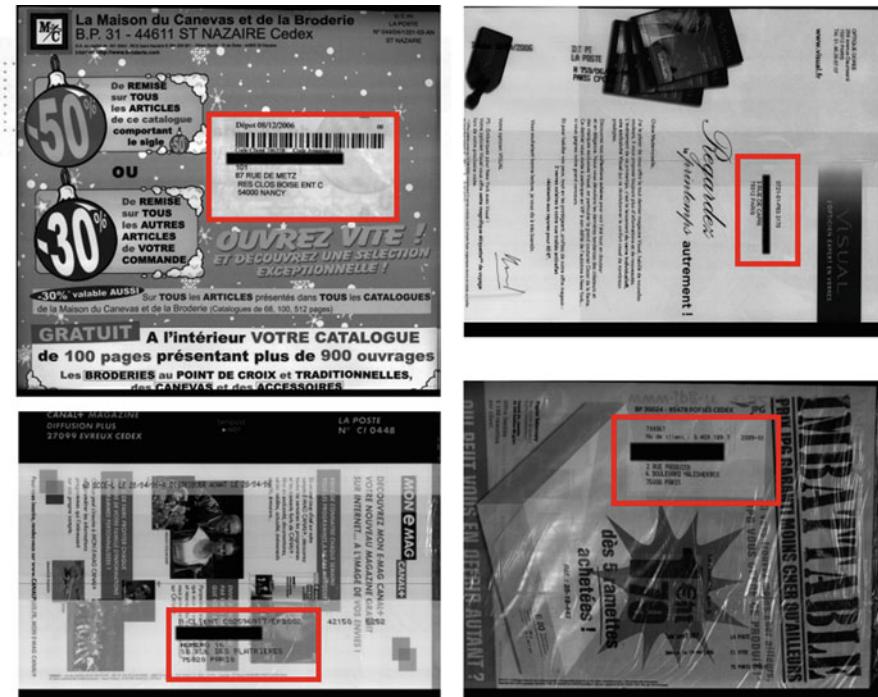
Large-volume lexicons are part of address databases which have to be matched with the address recognized on a mail item. This matching is a complex process which has to take into account the uncertainty of recognition and the discrepancies between the way an address is actually expressed on a mail item and its canonical form.

The texts to be recognized within postal applications are made of different fields written under different formats (numerals, words). The integration of several fundamental recognition techniques designed only for a specific class of images is also one of the challenges of address recognition. Figure 21.6 shows street lines with both numerals and alphabetic words.

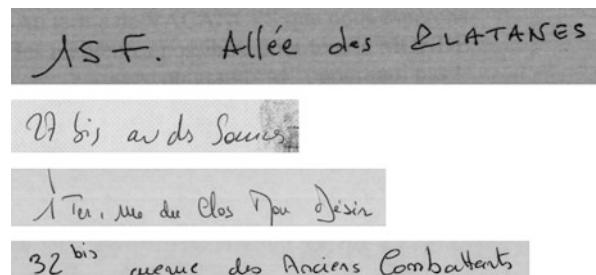
Finally, the variants in address syntaxes are numerous and necessitate the thorough collection of all possible cases during the design of a postal recognition system.

### A Survey of Specific Document Analysis Techniques Used in Postal Applications

The following sections describe the main processing steps in a postal address recognition system: address block location and extraction, ZIP code location and recognition, city name location and recognition, street name and street number location and recognition, and decision making, and the specific techniques



**Fig. 21.5** Examples of address blocks on flats (red rectangles show the location of the address)



**Fig. 21.6** Examples of street lines

commonly used to implement them. These steps are better understood if we describe first the overall recognition strategy for a postal address recognition system.

### Overall Strategy

A postal address is defined through different components whose form and possible values belong to sets of different complexities and sizes. The methods suited to

recognize each of them vary accordingly in their performance and in their need for computing resources. Moreover, the components are tied in various ways which reduce the potential value for a given one based on another one. For both reasons, the recognition strategy of an address recognition system is generally organized to implement the recognition of the components in an order that goes from the most general to the most specific (i.e., from the less constrained component, e.g., the ZIP code, to the most constrained one, i.e., the one whose value depends the most of the others ones, e.g., the street name which is constrained by the ZIP code and the city) and from the most easily recognized (digits strings) to the less easily recognized (words in large vocabularies). This strategy suffers from exceptions when the first components in this order list are not recognized or are missing. In such cases, recognition has to start with the most difficult task, and the recognition techniques applied are chosen for their capacity to face that difficulty within the limits of the constraints imposed to the overall address processing, especially in terms of computing time.

There are two classes of strategies which form the basis of an address recognition system and which are mixed at a certain degree depending on what is found on the image being processed (for the sake of clarity, we present them below in the simplified case of a ZIP code-city-street name-street number address. More complex cases appear when considering the destination country, the state, the extended ZIP code, the addressee name, etc., but without changing the philosophy of this overall strategy):

1. Progressive refinement strategy:

- (a) Locate and recognize the ZIP code.
- (b) Locate and recognize the city name within the cities compatible with the ZIP code.
- (c) Locate and recognize the street name within the street names of the recognized city.
- (d) Locate and recognize the street number within the range of allowed street numbers in the recognized street.

2. Bottom-up strategy:

- (a) Locate and recognize the ZIP code.
- (b) Locate and recognize the city name.
- (c) Locate and recognize the street name.
- (d) Locate and recognize the street number.
- (e) Integrate recognition hypotheses into a single interpretation based on the consistency constraints between ZIP, city, street name, and number.

The progressive refinement strategy is best suited for addresses with difficult to recognize components, in particular handwritten addresses. It is in particular useful for progressively reducing the size of lexicons among which recognition takes place up to the point where these sizes are compatible with available technologies. On another hand, progressive refinement strategies are not robust to recognition errors or even absence of recognition results in early stages since these early stages condition all the following ones.

The bottom-up strategy is more adapted to addresses with perfectly or almost perfectly recognized components for which the main difficulty is address interpretation rather than recognition, in particular printed addresses.

In the extreme, a simple and naïve implementation of the bottom-up strategy aims at extracting from the mail item image a string of ASCII characters corresponding to the address text and then to pass this string to a general-purpose ASCII address interpretation stage such as those used by direct marketing address management companies or even made available in map services on the Internet like Google maps. Although it may work for some simple cases, this naïve strategy, and in general most bottom-up strategies, fails on more difficult cases because on the one hand they assume that it is possible to recognize an address as any other text, which is highly difficult for ordinary handwriting, and, on a second hand, because they overlook some of the information available in the original image as, for example, all the recognition errors which are not taken into account by general-purpose ASCII address interpretation software. The only merit of the naïve implementation of the bottom-up strategy is the ease with which running systems may be quickly assembled by stacking general-purpose modules.

More complex implementations of the bottom-up strategy are also useful for bypassing the errors or the difficulties that may occur in early stages in progressive refinement strategies. For example, addresses with erroneous ZIP codes require the recognition of city names without prior knowledge of the ZIP code. Only bottom-up strategies will start by recognizing city names, but the price to pay for that is a larger lexicon size and thus a more difficult recognition task.

In practice progressive refinement and bottom-up strategies should be associated to get the best from each.

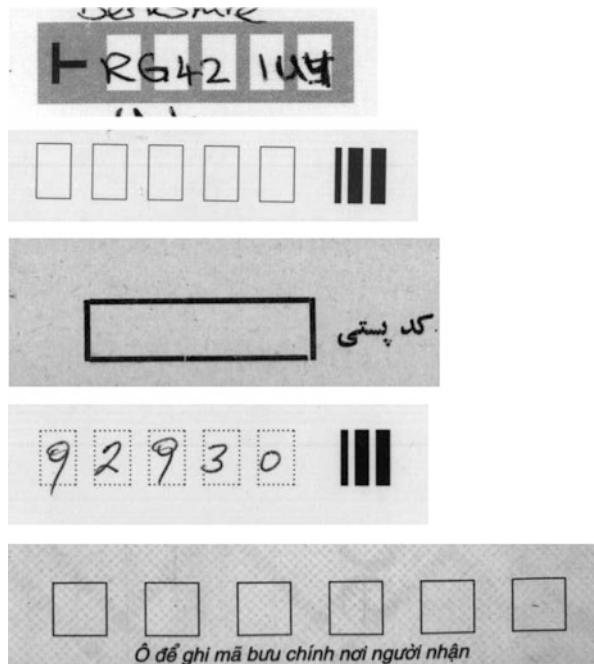
Another dimension for analyzing the architecture of address recognition systems is the distinction between handwritten and printed addresses. In the case of addresses printed with normal quality, address recognition reduces to address interpretation since recognition is easy so that the input data is equivalent to an ASCII text.

By contrast, printed addresses with low quality may be assimilated to handwritten addresses so that all techniques described below apply to both cases.

### **Address Block Location**

Various address block location techniques have been proposed for postal applications [14, 30]. In their majority, they rely on morphological analysis techniques used to group characters into words and lines, then lines into texts and to decide whether a text is a good candidate for being an address block. Some specific techniques are also used which take advantage of a number of peculiarities of mail items. Special marks for explicit address location have been proposed and suggested to large-volume mailers, sometimes with the help of incentives. Akin to marks are the various boxing systems used in different countries. The shape of boxes, or their predefined location, is a cue to locate further the address block. Opportunistic marks may also be made of bar codes often included within addresses. Transparent address

**Fig. 21.7** Address block locators and ZIP code boxes



windows may be also subject to specific detection algorithms. Figure 21.7 shows different address block locator marks and ZIP code boxes.

### ZIP Code Recognition

ZIP codes are usually placed in the leftmost or rightmost position within the last line of an address block. Therefore, typical ZIP code location and recognition methods, especially under progressive refinement strategies, consist in attempting a recognition in these expected locations. However, the exact position of the ZIP code may not be limited by a sufficient gap between the ZIP and the generally following or preceding city name. The recognition method used must then be robust to this uncertainty. The same type of difficulty is increased for bottom-up strategies which are particularly interesting when the ZIP is not in its normal position and which have to guess this position from the shape of words isolated during address block location.

ZIP codes can be considered as character strings. They are thus amenable to recognition methods specific to character strings.

In the particular case of ZIP codes, simple string recognition strategies have been used because in many cases ZIP codes are written with cleanly separated characters and also because the number of characters in a ZIP is usually known in advance. Within such strategies, characters positions are first isolated and then submitted to individual recognition in a purely sequential manner.

However, in many real cases, ZIP codes are written with touching characters and/or in an unknown number. General character string recognition techniques are then required.

There exist commonly two classes of general methods for the recognition of ZIP code strings: those with explicit character segmentation and those with implicit segmentation.

Explicit character segmentation-based methods implement the segmentation-by-recognition principle in which potential segmentation configurations are validated or invalidated by character recognition.

Common methods for detecting potential segmentation configurations rely of heuristic rules which recognize known configurations of touching characters. For any particular ZIP code recognition task addressing a certain character alphabet and given ZIP codes syntax and writing habits, specific rules have to be crafted in view of a large number of examples. Therefore, the sets of such rules change whenever the ZIP code system changes so that they cannot be described in general.

Symbol recognition also inherits from general-purpose methods for character recognition like statistical or neural network-based classifiers and adapted feature sets. Since ZIP codes are generally composed of digits and/or letters, they do not require other than by exception specifically designed recognizers adapted to other symbols than digits or letters. However, recognition engines for digits or for letters trained on images from other tasks than the particular ZIP codes to recognize shall suffer from degraded performances because the shape of digits and letters within ZIP codes is often influenced by the fact that they appear within a ZIP code. For example, the most frequent ZIP codes tend to end with several “0” written in smaller sizes than the rest of the ZIP and often merged by ligatures or by overlapping. In such cases, specific recognizers trained on corresponding examples and using adapted features are used.

Figure 21.8 shows an example of the recognition of a five-digit ZIP code based on the segmentation-by-recognition principle.

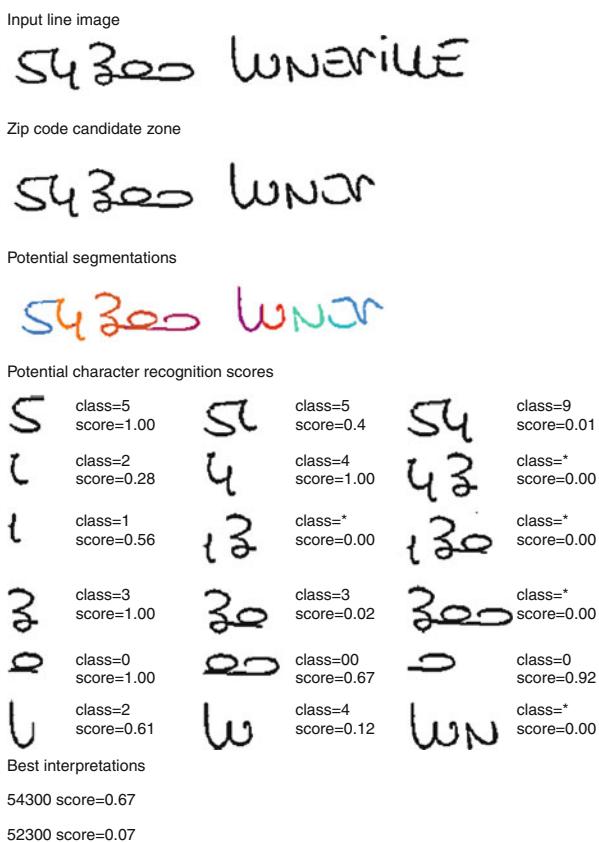
### **City Name Recognition**

City name recognition in postal application is a task directly dependent of the overall strategy which is implemented.

With progressive refinement strategies, city name recognition is guided by prior ZIP code recognition and has then to operate over a limited-size lexicon (typically of size 10–100) made of names of cities consistent with ZIP code recognition hypotheses. For that specific case, different word recognition methods in the literature [39] have been used in industrial systems with the most popular being analytical word recognition based on individual character recognizers and holistic word recognition based on Hidden Markov models. Both classes of methods have their strength and weaknesses for city name recognition and usually they are mixed into real systems. The fact that city names are often written in handprinted style is an argument in favor of the use of analytical methods.

In the case of bottom-up strategies, city name recognition is confronted to large lexicons of typically 10,000–100,000 entries. Not all word recognition methods are

**Fig. 21.8** ZIP code recognition based on the segmentation-by-recognition principle



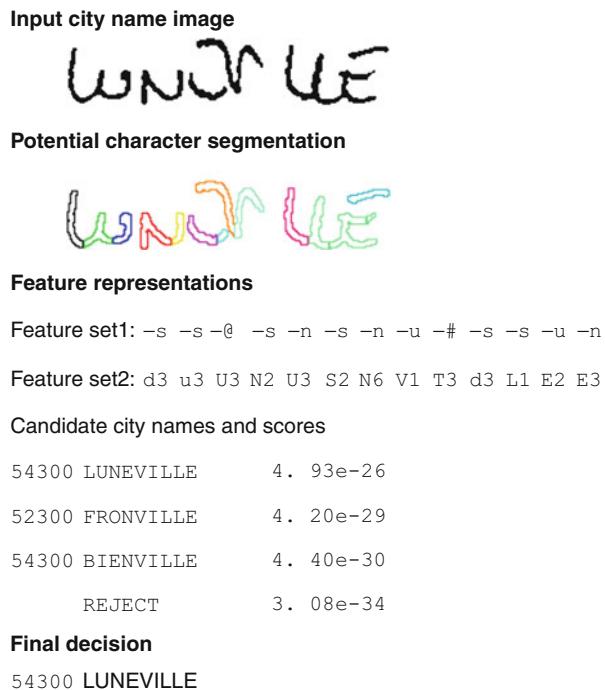
tractable for such a task because of their computing time, which may be linear in the number of lexicon entries, and because recognition must be of high performance to isolate the correct answer within thousands of candidates. In that case, analytical methods are preferred, which combined to tree organizations of lexicons, achieve high speed and precise recognition especially when names are written in separated handprinted letters.

Figure 21.9 shows an example of the recognition of a city name using discrete word representation features and a Markov model for scoring.

### Street Name Recognition

Street name recognition has to be distinguished from city name recognition because it has to take into account not only words but whole phrases made of several words with different roles and variations (numbers, street types, names, titles, abbreviations). Another difference lies in the fact that lexicon sizes for street name recognition may range up to thousands of entries (the number of street names in a large city) independently of the chosen overall strategy.

**Fig. 21.9** City name recognition using discrete word features



Two classes of specific recognition strategies are implemented for street name recognition.

Keyword-based strategies perform recognition of words rather than on whole phrases or lines. Here again, both analytical and holistic word recognition methods may be used and implemented in a very direct way on the word images isolated during address block location. However, keyword-based strategies suffer from several shortcomings. First, since they rely on word segmentation hypotheses, they may miss true word positions, e.g., when words in the street name are written without gaps. Secondly, they do not take into account relative word positions which constitute a useful cue for street name recognition.

By opposition, phrase-based strategies attempt to match the whole street name to candidates in the lexicon. Doing this they not only rely on content word recognition, but also they take advantage of other words in street lines like street type and article or prepositions. In that goal, methods based on a flexible alignment between images and lexicon entries are particularly suitable. All the recognition techniques using the framework of Hidden Markov models have this property and many have been used in industrial systems. The interest of phrase-based strategies may be lowered by their computational complexity and then the related computing time. Heuristic search strategies and tree-organized lexicons are used to solve these issues. Figure 21.10 depicts an example of the recognition of a street name using a keyword-based strategy.

**Input street name image**

4 Rue Carnot

**Potential character segmentations**

4 Rue Carnot

**Feature representations**

**Feature set1:** -s -# -s -s -s -# -s -s -n -s -u -s -n

**Feature set2:** L1 y2 R2 u2 u3 e1 c1 a1 E1 dN4 m6 o1 t1

**Candidate street types, positions and scores**

RUE	.... RRuuuu_.....	1.05e-31
-----	-------------------	----------

LA	LLa_.....	1.69e-33
----	-----------	----------

AVENUE	avvveennuuue_.....	4.32e-34
--------	--------------------	----------

**Candidate street key words and scores**

DE_CARNOT	1.09e-30
-----------	----------

CARNOT	2.79e-32
--------	----------

DES_CARMES	1.51e-33
------------	----------

LES_CARMES	1.39e-33
------------	----------

BACCARAT	6.47e-35
----------	----------

LEOMONT	7.83e-37
---------	----------

...

**Candidate full street names and scores**

RUE_CARNOT	1.01e-21
------------	----------

RUE_LEOMONT	2.83e-26
-------------	----------

REJECT	2.88e-26
--------	----------

**Final decision**

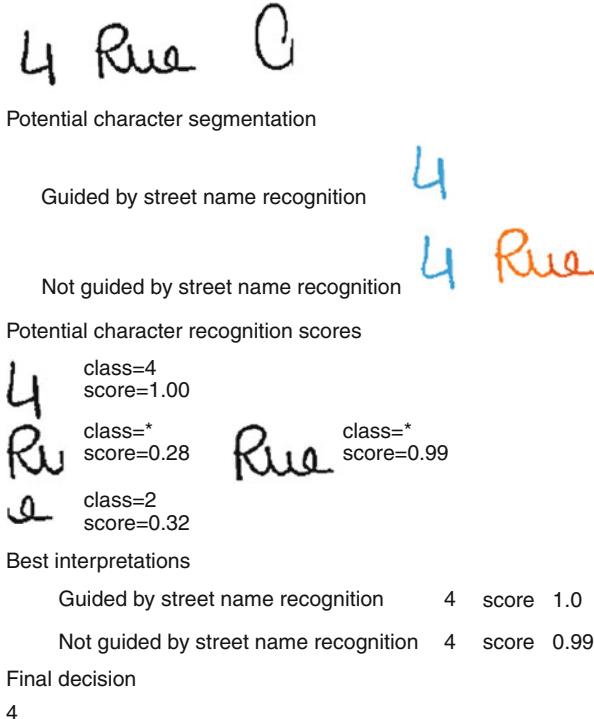
RUE\_CARNOT

**Fig. 21.10** Street name recognition with a keyword-based strategy

**Street Number Recognition**

Street number recognition is similar to ZIP code recognition except for the number of digits, which is not known in advance; for the precise position, which is not clearly marked with respect to the street name in the same line; and for the composition of the number, which may not include only digits (e.g., commas, dashes).

**Fig. 21.11** Street number recognition      Input candidate street number image



There exist two main classes of strategies for solving the above issues.

General-purpose character string recognition methods are sometimes applied to the expected position of the street number and augmented with additional character recognizers. In particular, the additional recognizers are in charge of detecting street number limits (e.g., commas) whenever the position is not precisely known.

Another class of strategies relies on phrase-based street name recognition. Here it is assumed that the flexible alignment between street names and street line images has a by-product which is the exact position of the street number within the line.

Figure 21.11 describes an example of the recognition of a street number based on the segmentation-by-recognition principle.

### Lexicon Representation

Postal recognition applications have also the particularity of relying on large-volume address databases with tight interaction with pure recognition techniques. Various methods which organize these interactions have been proposed and applied. They generally rely on heuristic search techniques of the artificial intelligence domain [22, 26]. Completely bottom-up strategies (i.e., text recognition then address matching) are in general not as powerful as the former methods.

**Table 21.1** State-of-the-art recognition rates in postal recognition applications

		Letters (%)	Flats (%)	Parcels (%)
ZIP code	Printed	>97	>90	>70
	Handwritten	>85	>80	>40
Street name and number	Printed	>95	>90	>60
	Handwritten	>65	>60	>20

Included in the relations between recognition and address databases are the algorithms used for recognizing words within large-volumes lexicons. All postal applications embed in that respect a specific compiler whose role is to transform the original address database into a representation suitable for fast recognition (this step is known under the name of Address DataBase Generation – ADBG). Tree-type representations and associated heuristic search techniques are often implemented [39].

The recognition of composite fields in addresses (ZIP code line with ZIP code, city and state, delivery line with street name, street number) requires also specifically designed methods which are able to recognize a word or phrase within a larger text and which are capable of discarding noisy signs as well as numerical or alphabetic ones. Keyword handwriting recognition techniques have been implemented in address readers.

## State of the Art in Postal Applications

Due to the differences in alphabets, languages, address syntax, address databases, and potential numbers of distinct outputs (e.g., number of possible ZIP codes) from country to country, it is not possible to give common state-of-the-art performance figures for all postal recognition applications [11, 12, 17, 32–34, 36, 37, 41, 42]. These figures also vary depending on the category of mail items considered: letters are easier to recognize than flats which are easier to recognize than parcels.

However, Table 21.1 gives the typical recognition rates achieved in today applications (at least with Latin alphabets) which are usually reached with an error rate around 1 %.

The figures demonstrate that there is still room for improving recognition technologies in postal applications.

## Reference Datasets

A number of image databases issued from postal recognition applications have been published and have become for some of them almost standards, not only for the evaluation of full applications but also for the testing of fundamental techniques. Among these are the Cedar database from the State University of New York in Buffalo (<http://www.cedar.buffalo.edu/Databases/>), the MNIST database

of handwritten digits (<http://yann.lecun.com/exdb/mnist/>), the MNIST database of handwritten upper-case letters (<http://publications.idiap.ch/index.php/publications/show/709>), and the KAIST-POWORD Handwritten Postal Hangul Word Image Database (<http://ai.kaist.ac.kr/Resource/dbase/Address/POWORD.htm>).

## Vendors

Postal recognition systems are designed and marketed by a number of specialized companies throughout the world among which the most important are given in Table 21.2.

---

## Check Processing Applications

### Specific Issues in Check Processing Applications

Through their particular characteristics, check recognition applications present a number of specific challenges for pattern recognition techniques.

Layout analysis in check recognition applications is facilitated by the design of checks which makes them special types of forms. However, this design has to reconcile the interest for easy to recognize documents with the task of conveying the best vision of a bank. The latter often gets priority so that check forms are encumbered by textured backgrounds and preprinted texts likely to appear behind the fields to be recognized. The separation of such graphics from useful text must be done prior to recognition.

In the useful fields themselves, one may find non-numeric signs (slashes, dots, commas, currency sign, currency name, pound signs, fraction bars, etc.) often used to prevent the fraudulent modification of amounts. The elimination or interpretation requires their specific recognition. Figure 21.12 shows examples of courtesy amounts with non-numeric signs included.

Courtesy and legal amounts feature all sorts of syntactic and lexical variants specific to the expression of amounts and within that context particular to each alphabet and language. The collection of all possible options is one of the necessary tasks in the design of a check recognition system. Figures 21.12 and 21.13 show examples of various amount syntaxes.

In order to implement amount recognition, and in particular legal amount recognition, one has to design ways of controlling digit or word recognition by the collected syntax. For courtesy amount recognition, this is made more difficult by the degrees of freedom in the vertical positions of graphemes which somewhat makes that recognition akin to that of mathematical formulas.

Finally, the high requirements in terms of low error rates compel the decision stages to output very accurate estimates of recognition reliability in the form of confidence values.

**Table 21.2** Indicative list of vendors of postal recognition software and systems

Company	Website/comments
A2iA	<a href="http://www.a2ia.com">www.a2ia.com</a> Based on his experience as a pioneer in check recognition software, A2iA has developed general-purpose handwriting recognition software engines which have been integrated by system integrators into postal address recognition systems in association with technologies from other vendors in countries like France, Germany, and the UK
Bell & Howell	<a href="http://www.bellhowell.net">www.bellhowell.net</a> Bell & Howell, LLC is a provider of solutions and services for paper-based and digital messaging. Bell & Howell provides address recognition systems adapted to presorting of outgoing mail for mailing companies in the United States
Elsag	<a href="http://www.elsagdatamat.com/Automazione.htm">www.elsagdatamat.com/Automazione.htm</a> Elsag is a provider of global solutions for postal sorting including sorting machines, information systems, image acquisition systems, hardware and software for address recognition and envelope interpretation, video coding systems. Elsag has provided address reading systems in many countries including Italy and France
Lockheed Martin	<a href="http://www.lockheedmartin.com/capabilities/it/">www.lockheedmartin.com/capabilities/it/</a> Lockheed Martin is an integrator of address recognition systems associating its own technologies to the technologies of other vendors. Lockheed Martin is in particular the integrator of the address readers used by the USPS and in the UK
NEC	<a href="http://www.nec.com/global/solutions/postal/content/systems.html">www.nec.com/global/solutions/postal/content/systems.html</a> NEC is a provider of global solutions for postal sorting including sorting machines, information systems, image acquisition systems, hardware and software for address recognition and envelope interpretation. NEC has provided address reading systems in many countries including Japan, Finland, and Asian countries
Parascript	<a href="http://www.parascript.com">www.parascript.com</a> Parascript is a provider of document recognition software. As part of this activity, Parascript has developed an address recognition software which is integrated by system integrators in countries like the USA and the UK
Prime Vision	<a href="http://www.primevision.nl">www.primevision.nl</a> Prime Vision, a former subsidiary of the Dutch national postal operator. Prime Vision was founded from the Netherlands National Postal and Telecommunications Corporation (PTT) research department. Prime Vision develops document recognition solutions in the domains of Logistics, Banking, Fraud prevention, and Regulation enforcement Prime Vision has implemented solutions from Asia Pacific to Australia, the Middle East, and North America
RAF Technology	<a href="http://www.raf.com">www.raf.com</a> RAF Technology, Inc. is a worldwide leader in advanced pattern and image recognition, intelligent information extraction, and data verification solutions for Courier, Express, Postal, Warehouse, Transportation, and Field Service, Enterprise, and Government agencies RAF provides flexible software solutions that enable organizations to more effectively process mail

(continued)

**Table 21.2** (continued)

Company	Website/comments
Seres	<a href="http://www.seres.fr">www.seres.fr</a> Seres, a subsidiary of the French La Poste, is a provider of general-purpose document recognition systems and software. Seres has developed the primary address recognition software used at La Poste
Siemens	<a href="http://www.mobility.siemens.com/mobility/global/EN/LOGISTICS/POSTAL-AUTOMATION/Pages/postal-automation.aspx">www.mobility.siemens.com/mobility/global/EN/LOGISTICS/POSTAL-AUTOMATION/Pages/postal-automation.aspx</a> Siemens is a provider of global solutions for postal sorting including sorting machines, information systems, image acquisition systems, hardware and software for address recognition and envelope interpretation. Siemens has provided address reading systems in many countries including Germany, the USA, and recently in Arabic countries
Solystic	<a href="http://www.solystic.com">www.solystic.com</a> Solystic is a provider of global solutions for postal sorting including sorting machines, information systems, image acquisition systems, hardware and software for address recognition and envelope interpretation
Toshiba	<a href="http://www3.toshiba.co.jp">www3.toshiba.co.jp</a> Toshiba is a provider of global solutions for postal sorting including sorting machines, information systems, image acquisition systems, hardware and software for address recognition and envelope interpretation. Toshiba has provided address reading systems in many countries including Japan, France, and Asian countries
Vitronic	<a href="http://www.vitronic.de">www.vitronic.de</a> Vitronic is a provider of machine vision systems and software in industry, logistics, and traffic control. Vitronic has developed address recognition systems including image acquisition devices, hardware, recognition software, and video coding systems adapted to parcel sorting

## A Survey of Specific Document Analysis Techniques Used in Check Processing Applications

The following sections describe the main processing steps in a check processing system, namely, field location and extraction, courtesy amount recognition, legal amount recognition, and decision making, and the specific techniques commonly used to implement them. These steps are better understood if we describe first the overall recognition strategy for a check recognition system.

### Overall Strategy

Beyond general-purpose corroboration methods, check recognition may take advantage of the fact that courtesy and legal amounts on a given check have normally the same value. The overall recognition strategy commonly uses this property to implement courtesy amount recognition and/or legal amount recognition based on

**Fig. 21.12** Courtesy amounts with non-numeric signs included

37400 #      23 Euro 10  
  # 50.000#      ~3845  
  180.000      -24.000=  
  36.000=      EUR -----369,25  
  60 Euros      CHAN CHAO CHAO CHAO CHAO  
                   +++++21.129+++++  
  70-E      80 E  
  31,02      60,57  
  cent quatre euros et 94 cents  
  Cent quatre euros et 94 cents  
  dix €

**Fig. 21.13** Legal amounts syntaxes

the information acquired in a bottom-up manner from the image of the check. The strategy has also to make the best compromise between performance (recognition rate and error rate) and computing resources. Since courtesy amount is often recognized faster than legal amount (the image of a legal amount contains more signs than the equivalent courtesy amount) and since courtesy amount fields are less prone to noisy background due to typical check form design, common strategies tend to privilege courtesy amount recognition over legal amount recognition.

The property that makes the legal amount directly equivalent to the courtesy amount within a given check also dictates strategies where legal amount recognition is performed in a verification mode over the hypotheses coming from the courtesy amount recognition.

- Therefore, a typical strategy is the following:
1. Perform courtesy amount recognition.
  2. If the confidence value of the best candidate from courtesy amount recognition is high enough, then this candidate is the final result.

3. Else, if the list of candidates from courtesy amount recognition is not empty,
  - (a) Perform legal amount recognition in verification mode
  - (b) Integrate courtesy amount and legal amount recognition candidates.
  - (c) The best candidate after integration is the final result.
4. Else, if the list of candidates from courtesy amount recognition is empty,
  - (a) Perform legal amount recognition in recognition mode.
  - (b) Perform courtesy amount recognition in verification mode.
  - (c) Integrate courtesy amount and legal amount recognition candidates.
  - (d) The best candidate after integration is the final result.

Note that with such a general strategy, the legal amount may not be checked when the process ends in step 2. Although the compliance with local regulations may require to check the value of the legal amounts, practical implementations may take the risk of not doing this checking to privilege the recognition rate, and thus the level of automation, based on the fact that most of the time the legal amount is the same as the courtesy amount. A common refinement consists in trying to recognize the legal amount when the risk is high, that is, when the recognized value for the courtesy amount exceeds some threshold.

## Field Location and Extraction

Field location and extraction is the process by which the different fields in a check image are automatically located and isolated from the rest of the check image before being processed during their corresponding recognition steps.

Among the specific recognition techniques specifically used in check recognition applications [18], those in charge of the location and the extraction of fields may take advantage of the regularities of check forms.

A commonly used set of methods for field location and extraction rely on check templates registration [29] where a number of reference images of blank checks are stored as templates and compared to the processed image. Once the processed imaged has been identified to a given template, the location of the different fields is directly derived from the field positions associated to the template.

Matching of reference images to the current image may be based on direct pixel-to-pixel comparison or on general-purpose form recognition techniques which define templates through a set of characteristic zones (called anchors) and which match these anchors to similar region in the processed image. Typical anchors are shapes like lines ends and crosses, logos, characters, and words.

By matching anchors and not just pixels, it is possible to create robustness to small deformations of the image during its acquisition which excludes perfect alignment with the model. In that case, the relative position of anchors in the model and in the image is the base to the computation of deformation parameters like slant, translation, and scaling. Given these deformation parameters, the actual location of fields in the image is adapted from their canonic location in the model.

In practice, there exist hundreds of different check models due to the variety of banks and even to the variants of check forms for one particular bank at one time and also over time. This is why another set of common techniques do not rely on explicit

models but try to adapt automatically to any type of check by just considering the constant and common properties of check forms and of fields in these checks.

In that respect, courtesy amounts may be often located and extracted by searching for a currency sign in a restricted part of the image [10, 24]. The courtesy amounts/location is then defined as the region of a given size typically on the right of the currency sign. In a similar way, legal amounts, dates, and account numbers can be located by looking for horizontal guidelines in specific position and with specific size.

For check forms for which there are no guidelines or whose guidelines disappear during image acquisition (i.e., guidelines printed in light colors or made of dots), another set of techniques try to locate directly courtesy amounts by detecting line of characters in typical location for courtesy amounts and with typical relative distance and justification. In that purpose character lines are revealed by applying morphological processing methods like smearing which tend to group characters and words in a line as a rather long and horizontal group of connex pixels.

Once located, the fields have to be extracted from the rest of the image by keeping only their component pixels and getting rid of pixels coming from the check form itself.

Location methods based on templates usually perform field extraction by subtracting the recognized blank template to the image to be processed. Here again, this works only for ideal cases without any deformation.

In more realistic situations, general-purpose line and box extraction are used to remove the shapes coming from the check form in the located zones without any knowledge of the particular template in the image.

After field location and extraction, the fields are prepared for being recognized through their respective recognition technique.

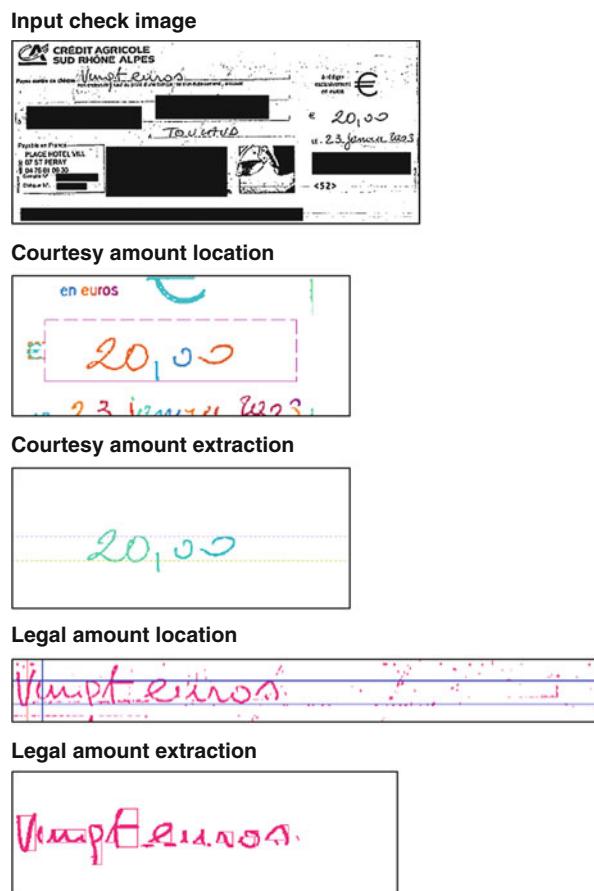
Figure 21.14 demonstrates the extraction of courtesy and legal amounts on a given check.

## Courtesy Amount Recognition

Numerical (courtesy) amounts belong to the class of symbol strings respecting a certain number of syntactic constraints as opposed to words or phrases formed over a given vocabulary. Therefore, courtesy amount recognition inherits of the corpus of general methods designed to recognize symbol strings. In the specific case of courtesy amounts, these general methods are specialized by using particular rules for explicit segmentation and for character or symbol recognition and are augmented with techniques for taking into account syntactic constraints, the only approximate 1D character of courtesy amount strings, for finally interpreting a set of symbols as a numeric amount, and for providing a set of scored candidates for integration in the overall decision making strategy.

There exist commonly two classes of general methods for the recognition of symbol strings: those with explicit character segmentation and those with implicit segmentation.

**Fig. 21.14** Extraction of courtesy and legal amounts



Explicit character segmentation-based methods implement the segmentation-by-recognition principle in which potential segmentation configurations are validated or invalidated by symbol recognition.

Common methods for detecting potential segmentation configurations rely of heuristic rules which recognize known configurations of touching symbols. For any particular check recognition task addressing a certain currency, a certain country, a certain alphabet or language, and certain courtesy amount-forming habits, heuristic rules are established from the analysis of large sets of representative examples so that there exists no general purpose set of rules efficient across all task configurations.

Symbol recognition also inherits from general-purpose methods for character recognition like statistical or neural network-based classifiers and adapted feature sets. Since courtesy amounts are often composed not only of digits but also of all sorts of non-digit signs (start and end strokes, currency symbol, currency name, comma, etc.), specific recognition techniques are also applied for non-digit

signs [16]. They may be implemented by extending the digit recognizers with the considered classes, or they may rely on specific detectors with adapted feature extraction and binary classifier. For example, linear strokes at the start and end of a courtesy amount may be recognized through specific features like the aspect ratio of the symbol or its second-order moments. Commas are also often detected through their absolute size and their relative position to the rest of the field. Specialized recognizers are also often used to take into account difficult segmentation configurations specific to the case of courtesy amounts. For example, touching double zeros which appear often within amounts may be recognized as a single symbol rather than being explicitly segmented. A last class of specialized recognizers may be used to detect others types of touching symbols configurations which should not be classified as a particular meaningful class but should rather be discarded as garbage. Adapted features like the size and the aspect ratio of the number of black-white or white-black transitions are used along with general-purpose classifiers to cope with these specific sub-images.

Basic segmentation-by-recognition methods assume that they process one-dimensional symbol string. In the case of courtesy amounts, they have to be augmented with the capacity of dealing with almost 2D configurations to take into account, for example, currency signs place over a comma or fraction parts in an amount. Common techniques used in that respect extend segmentation-by-recognition to operate over recognition graphs or lattices and not just strings.

In a difficult task like courtesy amount recognition, it is important to take advantage of any a priori over the possible interpretations of a field. Syntactic constraints are such an a priori and are thus commonly embedded into courtesy amount recognition. These constraints are typically modeled as regular expressions or finite-state automata which constrain the paths allowed within the recognition hypothesis lattice. It is also common to incorporate into the syntactic constraints the interpretation rules which are used to transform an amount expression into a numeric value. The most frequent and natural way for implementing interpretation rules consists in building these rules in the syntactic constraints as variable extraction in the case of regular expressions or as two-level finite-state automata.

Implicit segmentation-based methods do not explicitly define segmentation configurations but derive segmentation as a by-product of recognition. Within that class, Hidden Markov Model (HMM) techniques are the most employed for courtesy amounts. Here the modeling of a courtesy amount is no longer based on the manual identification of heuristic rules but rather on the automatic training of the recognition model over a set of representative examples. In order to take into account the peculiarities of courtesy amounts, general-purpose Hidden Markov Model methods have also to be augmented with syntactic constraints, semantic interpretation, and 2D aspects. A common way to embed these issues in an HMM consists in extending and composing the basic finite-state automaton equivalent to the HMM with other automata modeling the syntax, the interpretation, and the 2D lattice of a courtesy amount.

Industrial courtesy amount recognition systems try also to take advantage of the orthogonality of different individual recognition methods to improve their overall

performance. Various strategies are employed in that purpose. Within the class of explicit segmentation-by-recognition techniques, several symbol recognizers may be associated. It is also possible to combine an implicit segmentation recognizer with an explicit one. The perturbation principle strategy is also commonly used with perturbations slightly modifying the image of the amount by performing arbitrary skew corrections, for example.

Be it based on explicit or implicit segmentation, courtesy amount recognition has to come out with a set of candidate amounts together with confidence values adapted to the overall decision strategy. These confidence values have to take into account not only the symbol recognition scores but also the syntax and numeric interpretation of the amount and potentially some priors about the amount value. Many type of scoring principles are used in that purpose like neural network integrators or Bayesian combination rules.

The above is valid for courtesy amount recognition applied in pure recognition or in verification mode. In the particular case of verification mode, the syntactic constraints and the scoring priors are modified to only accept the candidate amounts input to the verification.

As an illustration of the above principles, Fig. 21.15 demonstrates the steps in the recognition of a courtesy amount.

### Legal Amount Recognition

As a first approximation, legal amounts belong to the class of words or phrases formed over a given vocabulary. Therefore, legal amount recognition inherits of the corpus of general methods used for the recognition of words and phrases.

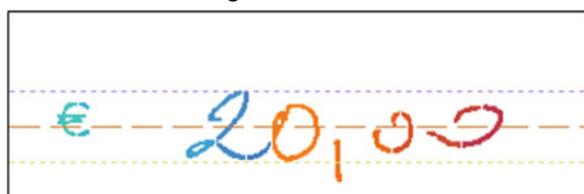
There exist two main approaches to recognize words and phrases which are both used for legal amount recognition: analytical and holistic recognition. In the case of legal amount recognition, because the vocabulary of words is of limited size, hybrid analytical and holistic methods are also employed.

Analytical recognition explicitly segments words and phrases into units (typically characters, graphemes, words) whose individual recognition is then combined at the word or phrase level. Different general-purpose methods are used in the context of legal amount recognition: Hidden Markov Models with explicit grapheme segmentation using various grapheme feature extraction schemes, hybrid Hidden Markov Model-neural nets techniques which use neural net recognition as a feature extractor, and edit-distance combined with character recognizers.

Holistic recognition methods do not explicitly segment words and phrases but extract global features (e.g., ascenders, descenders, loops, pixel densities within grids) and match these features with words in the lexicon through a number of classification schemes: e.g., k-nearest neighbors, multilayer perceptrons, and ad hoc distances. These methods become applicable in the context of legal amount recognition because the size of the vocabulary is typically less than 50 units. Thus, recognizing a word among 50 using only rough global features is acceptable. Holistic recognition methods present the advantage of being able to take into account contextual effects in the shape of letters within words [4].

**Fig. 21.15** Several steps in the recognition of a courtesy amount

#### Potential character segmentation



#### Potential character recognition scores

	class=euro score=1.00
	class=2 score=1.00
	class=0 score=1.00
	class=comma score=1.00
	class=0 score=1.00
	class=0 score=1.00

#### Best interpretations

20.00 score=1.00  
2.00 score=0.25

The moderate size of the lexicons also justifies the fact hybrid analytical-holistic methods have been used where analytical features designed at the grapheme or character level are viewed as holistic features.

Word recognition techniques, either analytical or holistic, have to produce a list of candidate words together with their confidence values. These confidence values depend on the particular recognition method used and can be accordingly interpreted as distances, probabilities, and likelihoods. For integration purposes, they are often normalized within a 0–1 scale and all confidence values sum up to 1. As for courtesy amount recognition, perturbation and multi-recognition engine strategies are commonly applied. In that case, confidence values coming from different recognizers are integrated into a single one through general-purpose integrators.

For legal amount recognition, recognition may take place at the word level or at the phrase level. When done at the word level, recognition has to be preceded by a word segmentation step which separates words based on gaps between them.

Since this segmentation may not be 100 % reliable because some writers tend to omit word gaps, words are commonly organized into a lattice of segmentation hypotheses. This lattice is in turn used as the input to a syntactic analysis typically based on a finite-state automaton version of the syntax of legal amounts. This analysis has also to produce the numeric interpretation of the legal amount through standard syntactic analysis techniques.

In many cases, legal amounts include decimal parts expressed in digits rather than in words as they should be. To take into account such cases, a recognizer equivalent to that used for courtesy amounts but with parameters suited to decimal values is applied. Due to its potential cost in terms of computing resources used, it is important to decide when to apply it. A rejection scheme based on a threshold over confidence values at the word recognition level plus additional value like the position in the legal amount image is normally used for that purpose.

The above is valid for legal amount recognition applied in pure recognition or in verification mode. In the particular case of verification mode, the syntactic constraints are modified to only accept the candidate amounts input to the verification.

Figure 21.16 illustrates these principles in the recognition of a legal amount guided by the courtesy amount recognition hypotheses.

### Decision Making

Decision making is the step which integrates candidates coming from both the courtesy and legal amount recognition within the overall recognition strategy described above. The result of the integration may be a single amount whose confidence is high enough to be the only interpretation of the image, or a list of candidate amount together with their confidence value which may then be used in a higher-level decision process like the balancing of a set of checks whose sum of amounts is known from outside, or a rejection decision when the list of candidates is empty.

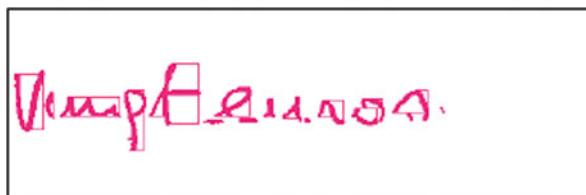
Here again general integration schemes are used to integrate the candidate list coming from the courtesy and legal amounts [19].

## State of the Art in Check Processing Applications

The performance of check recognition applications is highly dependent on the country where it is put into practice because checks forms and the way they are filled up differ strongly from one country to another [5, 8, 10, 15, 20, 23, 40, 43]. Moreover, the range of possible amounts depends on the currency and on local habits about what checks are used to pay for. However, practical systems with read rates as high as 75 % at 0.1 % error rates have been reported. In some applicative contexts, the potential amount for a check is known prior to recognition because it has been read on a remittance slip or has been keyed on an automated teller machine. It may also be constrained rather than precisely known in applications where only a limited number of amounts are possible or when a check is remitted within a batch whose total value is known. In those cases, recognition performances at 0 % reject rate are the figures to be considered. Recognition rates higher than 90 % have been reported in such contexts.

**Fig. 21.16** Legal amount recognition and decision

#### Input line image



#### Potential word segmentations

PW1: vingt

PW2: euros.

PW3: vingt euros.

#### Word re

PW1:	word=VINGT	score = 1.67e-11
	word=DOUZE	score = 5.93e-16

PW2:	word=EUROS	score = 6.45e-12
	word=DEUX	score = 9.40e-17

PW2:	word=SOIXANTE	score = 8.78e-27
	word=QUARANTE	score = 3.45e-29

#### Courtesy amount hypotheses

20.00 score=1.00

2.00 score=0.25

#### Legal amount hypotheses (guided by courtesy amount)

VINGT_EUROS	score = 1.08e-22
-------------	------------------

DEUX_EUROS	score = 2.50e-34
------------	------------------

#### Final decision and score

20.00 score=0.999

## Reference Datasets

Few image databases have been made public for check recognition. Most of the existing databases have been kept secret by banks and vendors due to private confidentiality issues. Among the few available ones, one can cite the database by the University of Bari, Italy [9], and the CENPARMI Arabic checks dataset [1].

**Table 21.3** Indicative list of vendors of check recognition software and systems

Company	Website
A2iA	<p><a href="http://www.a2ia.com">www.a2ia.com</a></p> <p>A2iA is a leading developer of natural handwriting recognition, Intelligent Word Recognition (IWR) and Intelligent Character Recognition (ICR), technologies and products for the payment, mail, document, and forms processing markets. A2iA is a pioneer in check processing. Its software technology is deployed through system integrators</p> <p>A2iA's technology is available in 6 language versions: English, French, German, Italian, Portuguese, and Spanish and 23 country-specific technology versions</p>
Kappa Image	<p><a href="http://www.kappaimage.com">www.kappaimage.com</a></p> <p>Kappa Image delivers fraud detection software and services. Kappa, currently protecting over ten million accounts globally, is a software solution providing automated forgery and counterfeit detection from check or giro images</p>
Mitek Systems	<p><a href="http://www.miteksystems.com">www.miteksystems.com</a></p> <p>Mitek Systems is currently a provider of mobile imaging solutions. Mitek Systems has long been an industry leader and innovator in image analytics and pattern recognition. Mitek Systems delivers a software suite of applications and tool kits that focus on the areas of Payments, Document identification, Signatures, Fraud, and Data Capture. Mitek Systems suite is used in Mitek Systems mobile check deposit product</p>
Orbograph	<p><a href="http://www.orbograph.com/car-lar-check-recognition-software.htm">www.orbograph.com/car-lar-check-recognition-software.htm</a></p> <p>Orbograph is a provider of recognition-centric services and software for check processing in the financial industry and end-to-end electronic solutions in healthcare revenue cycle management. Orbograph associated its own check recognition technology with technologies from other vendors</p>
Parascript	<p><a href="http://www.parascript.com">www.parascript.com</a></p> <p>Parascript is a provider of software and technologies for document recognition and other intelligent pattern recognition applications like medical imaging. Parascript was a pioneer in check processing software. Its technology was first created in the 1980s at the Soviet Academy of Sciences working on a theory that a computer could actually read handwriting.</p> <p>Parascript has extended its check recognition technology to international, non-US markets to read digitized check images specific to a particular country. Parascript offers versions for Argentina, Australia, Brazil, Canada, Chile, France, India, Malaysia, Portugal, Puerto Rico, and the UK</p>
Seres	<p><a href="http://www.seres.fr">www.seres.fr</a></p> <p>Seres is a subsidiary of the French La Poste. Its check recognition technology is used internally in the La Poste Group</p>

## Vendors

Check recognition systems are designed and marketed by a number of specialized companies throughout the world among which the most important are given in Table 21.3.

## Conclusion

Postal and check recognition systems are two of the most successful and significant pattern recognition applications through their economic worth and technical complexity. Millions of mail items and checks are processed every day in the world using such systems and millions of work hours are saved. These applications embed a large array of pattern recognition technologies encompassing most of the research topics in the domain. Through the hard challenges that postal and check recognition systems design have raised, many technologies have evolved and many new ones have emerged. Postal and check recognition systems illustrate how fundamental recognition methods have to be adapted to tackle real-life problems. They also prove the critical importance of the integration of basic techniques in the success of a real-world application.

The success and popularity of postal and check recognition primarily implemented in the industrialized countries has recently triggered the development of new systems designed for more and more countries. Through this, a large number of different languages have become subjects for pattern recognition research.

Although they have reached a high level of performance, postal and check recognition systems are still subject to improvement and will benefit in the next years from new technologies. Among these progresses, the application of address recognition to parcels – the only class of paper documents at large whose volume is increasing due to the Internet-based commerce – which has already started has the largest potential for performance increases [6, 27]. Another critical issue for the improvement of address recognition systems is the development of more complete reference address databases. Systems that will learn new addresses from the images they process are a research topic.

While the attractiveness of paper-based communication is steadily decreasing, vendors and users of postal and check recognition systems have been looking for ways of improving the value of paper documents by endowing them with new high-tech functions. In that aim, pattern recognition technologies have a prominent role to play and have actually been exploited.

In the mail domain, postal operators are attempting to support their development by marketing new services based on the recognition of mentions other than addresses on mail items. These new functions concern the prevention of fraud with the automatic control of meter marks and of stamps, the tracking-tracing of individual items, the determination of individual addressees for mail forwarding, the recognition of logos for data mining purposes, the delivery of mail under electronic form, the recognition of stickers, the recognition of stamps for advertisement, etc.

The contents of mail item themselves are becoming a new target for pattern recognition applications with the arrival of mailroom solutions [13].

In the check domain, many security issues motivate the recognition of other fields that amounts like payee bank account numbers, dates, and payer address. New regulations like Check 21 in the United States have also raised the need for new functions like the remote deposit of checks from mobile devices like smartphones.

Beyond their inherent benefits, all these new services are likely to reactivate the virtuous cycle in which postal and check recognition applications will raise new challenges from which the research community will provide them with new answers.

---

## Cross-References

- ▶ [Analysis of the Logical Layout of Documents](#)
  - ▶ [Continuous Handwritten Script Recognition](#)
  - ▶ [Document Creation, Image Acquisition and Document Quality](#)
  - ▶ [Handprinted Character and Word Recognition](#)
  - ▶ [Image Based Retrieval and Keyword Spotting in Documents](#)
  - ▶ [Imaging Techniques in Document Analysis Processes](#)
  - ▶ [Machine-Printed Character Recognition](#)
  - ▶ [Page Segmentation Techniques in Document Analysis](#)
  - ▶ [Page Similarity and Classification](#)
  - ▶ [Recognition of Tables and Forms](#)
  - ▶ [Tools and Metrics for Document Analysis Systems Evaluation](#)
- 

## References

1. Al-Ohali Y, Cheriet M, Suen C (2000) Database for recognition of handwritten Arabic cheques. In: Proceedings of the 7th international workshop on frontiers in handwriting recognition, Amsterdam, pp 601–606
2. Bayer T (2005) Method and device for sorting parcels. US Patent 6,888,084
3. Casey RG, Lecolinet E (1996) A survey of methods and strategies in character segmentation. *IEEE Trans Pattern Anal Mach Intell* 18(7):690–706
4. Castro M et al (2005) A holistic classification system for check amounts based on neural networks with rejection. *Pattern Recognit Mach Intell Lect Notes Comput Sci* 3776:310–314
5. Cheriet M, Al-Ohali Y, Ayat N, Suen C (2007) Arabic cheque processing system: issues and future trends. In: Chaudhuri BB (ed) Digital document processing advances in pattern recognition. Springer, London, pp 213–234
6. de Rijcke M, Bojovic M, Homan W, Nuijt M (2005) Issues in developing a commercial parcel reading system. In: Proceedings of the eighth international conference on document analysis and recognition, Seoul, pp 1015–1019
7. Desprez O, Caillon C, Miette E (2010) Method of processing postal items including management of digital fingerprints of the postal items. US Patent 7,674,995
8. Dimauro G, Impedovo S, Pirlo G, Salzo A (1997) Automatic bankcheck processing: a new engineered system. *Int J Pattern Recognit Artif Intell* 11(4):467–504
9. Dimauro G et al (2002) A new database for research on bank-check processing. In: Proceedings of the eighth international workshop on frontiers in handwriting recognition 2002, Niagara-on-the-Lake, pp 524–528
10. Dzuba G, Filatov A, Gershuny D, Kil I, Nikitin V (1997) Check amount recognition based on the cross validation of courtesy and legal amount fields. *Int J Pattern Recognit Artif Intell* 11(4):639–655
11. El-Yacoubi M, Gilloux M, Bertille J-M (2002) A statistical approach for phrase location and recognition within a text line: an application to street name recognition. *IEEE Trans Pattern Anal Mach Intell* 24(2):172–188. Table of contents archive

12. Filatov A, Nikitin V, Volgunin A, Zelinsky P (1999) The AddressScript recognition system for handwritten envelopes. In: DAS'98 selected papers from the third IAPR workshop on document analysis systems: theory and practice, Nagano. Springer, pp 157–171
13. Fujisawa H (2008) Forty years of research in character and document recognition—an industrial perspective. *Pattern Recognit* 41(8):2435–2446
14. Gaceb D (2008) Improvement of postal mail sorting system. *Int J Doc Anal Recognit* 11(2): 67–80
15. Gorski N (2001) Industrial bank check processing: the A2iA CheckReaderTM. *Int J Doc Anal Recognit* 3(4):196–206
16. Impedovo D, Greco N, Lucchese MG, Salzo A, Sarcinella L (2003) Bank-check processing system: modification due to the new European currency. In: Proceedings of the 7th international conference on document analysis and recognition – ICDAR'03, Edinburgh, pp 343–347
17. Ishidera E, Nishiwaki D, Yamada K (1997) Unconstrained Japanese address recognition using a combination of spatial information and word knowledge. In: Proceedings of the 4th international conference on document analysis and recognition, Ulm, 18–20, 1997, p 1016
18. Jayadevan R, Kolhe SR, Patil PM (2011) Automatic processing of handwritten bank cheque images: a survey. *Int J Doc Anal Recognit (IJDAR)* 15(4):1–30
19. Kaufmann G, Bunke H (2000) Automated reading of cheque amounts. *Pattern Anal Appl* 3:132–141
20. Kim G, Govindaraju V (1997) Bank check recognition using cross validation between legal and courtesy amounts. *Int J Pattern Recognit Artif Intell* 11(4):657–674
21. Kittler J, Hatef M, Duin RPW, Matas J (1998) On combining classifiers. *IEEE Trans Pattern Anal Mach Intell* 20(3):226–239
22. Lee CK, Leedham CG (2004) A new hybrid approach to handwritten address verification. *Int J Comput Vis* 57(2):107–120
23. Leroux M, Lethelier E, Gilloux M, Lemarie B (1997) Automatic reading of handwritten amounts on French checks. *Int J Pattern Recognit Artif Intell* 11(4):619–638
24. Liu K, Suen CY, Cheriet M, Said JN, Nadal C, Tang YY (1997) Automatic extraction of baselines and data from check images. *Int J Pattern Recognit Artif Intell* 11(4):675–697
25. Marti UV, Bunke H (2001) Text line segmentation and word recognition in a system for general writer independent handwriting recognition. In: 6th international conference on document analysis and recognition, Seattle, pp 159–163
26. Mercier D, Cron G, Deneux T, Masson M-H (2009) Decision fusion for postal address recognition using belief functions. *Expert Syst Appl Int J* 36(3):5643–5653
27. Miletzki U (2008) Genesis of postal address reading, current state and future prospects: thirty years of pattern recognition on duty of postal services. In: Proceedings of the 14th ACM SIGKDD international conference on knowledge discovery and data mining, Las Vegas, pp 5–6
28. Niblack W (1986) An introduction to digital image processing. Prentice-Hall, Englewood Cliffs
29. Okada M, Shridhar M (1997) Extraction of user entered components from a personal bank check using morphological subtraction. *Int J Pattern Recognit Artif Intell* 11(5):699–715
30. Palumbo PW, Srihari SN, Soh J, Sridhar R, Demjanenko V (1992) Postal address block location in real time. *Computer* 25(7):34–42
31. Pitrelli JF, Subrahmonia J, Perrone MP (2006) Confidence modeling for handwriting recognition: algorithms and applications. *Int J Doc Anal Recognit* 8(1):35–46
32. Roy K, Vajda S, Belaid A, Pal U, Chaudhuri BB (2005) A system for Indian postal automation. In: Proceedings of the eighth international conference on document analysis and recognition, Seoul, pp 1060–1064
33. Setlur S, Lawson A, Govindaraju V, Srihari SN (2001) Large scale address recognition systems truthing, testing, tools, and other evaluation issues. *Int J Doc Anal Recognit* 4(3):154–169
34. Seyedin SA, Tabatabaei Mashadi N, Seyedin SMM (2008) Classifying envelopes using machine vision in reading & processing farsi (Persian) handwritten addresses. In: 5th Iranian conference on machine vision and image processing MVIP 2008 in Persian, Tabriz, Iran, pp 1–7

35. Srihari SN (2000) Handwritten address interpretation: a task of many pattern recognition problems. *Int J Pattern Recognit Artif Intell* 14(5):663–674
36. Srihari SN, Kuebert EJ (1997) Integration of hand-written address interpretation technology into the United States postal service remote computer reader system. In: Proceedings of the 4th international conference on document analysis and recognition, Ulm, pp 892–896
37. Vajda S et al (2009) Automation of Indian postal documents written in Bangla and English. *Int J Pattern Recognit Artif Intell* 23(8):1599–1632
38. Vinciarelli A (2002) A survey on off-line cursive script recognition. *Pattern Recognit* 35(7):1433–1446
39. Vinciarelli A, Bengio S, Bunke H (2004) Offline recognition of unconstrained handwritten texts using HMMs and statistical language models. *IEEE Trans Pattern Anal Mach Intell* 26(6): 709–720
40. Wang D (2009) Study on information fusion based check recognition system, cutting-edge research topics on multiple criteria decision making. *Commun Comput Inf Sci* 35:384–391
41. Wang C, Hotta Y, Suwa M, Naoi N (2004) Handwritten Chinese address recognition. In: 9th international workshop on frontiers in handwriting recognition, Kokubunji, pp 539–544
42. Xue G, Lianwen J (2009) SwiftPost: a vision-based fast postal envelope identification system. In: Proceedings of the 2009 IEEE international conference on systems, man and cybernetics, San Antonio, pp 3268–3273
43. Yu ML, Kwok PCK, Leung CH, Tse KW (2001) Segmentation and recognition of Chinese bank check amounts. *Int J Doc Anal Recognit* 3(4):207–217

## Further Reading

In this chapter postal mail and check processing applications have been demonstrated as drawing their success and characteristics primarily from a clever integration of a number of different document processing techniques. Therefore, readers interested by more detailed descriptions of the involved techniques should refer to many other chapters in this handbook and in particular those cited in the Cross-References section below.

Postal mail and check processing systems are today the focus of two niche markets where a small set of commercial companies have entered into a strong competition. As a consequence, few technical details will be found in the documentations made available by system providers which of course aim to keep secret the distinctive features of their proprietary technologies. However, consulting recurrently these documentations may at least give some hints as to the progress of systems performances and as to the evolution of their functional coverage. There also exist cases where disclosure is compatible with ownership protection especially in the case of patent publication. Interested readers will find detailed descriptions of innovative methods and functions in published patents (e.g., [2, 7]). However, due to the characteristics of patent regulations in the concerned countries, one is more likely to find in this type of source descriptions of novel functionalities rather than detailed document processing methods.

In some other cases (languages and countries not yet addressed by commercial system providers or research teams using postal and check applications as an exemplification of their specific research interests), detailed system descriptions or pointers to their component techniques will be found in research papers published in major journals and conferences having document processing or neighboring domains as one of their focus (e.g., [5, 8, 20, 32, 37, 40, 42]).

---

# Analysis and Recognition of Music Scores

# 22

Alicia Fornés and Gemma Sánchez

## Contents

Introduction.....	750
Music Scores: Problem Definition.....	750
History and Importance of OMR.....	752
Summary of the State of the Art.....	755
State of the Art in the Different Stages of OMR.....	755
Introduction.....	755
Preprocessing.....	755
Staff Detection and Removal.....	755
Music Symbol Extraction and Classification.....	760
Syntactical Analysis and Validation.....	764
State of the Art in Different Domains.....	766
Introduction.....	766
Old Handwritten Music Scores.....	766
Modern Handwritten Music Scores.....	768
Conclusion.....	770
Consolidated Software.....	770
Cross-References.....	771
References.....	771
Further Reading.....	774

---

## Abstract

The analysis and recognition of music scores has attracted the interest of researchers for decades. Optical Music Recognition (OMR) is a classical research field of Document Image Analysis and Recognition (DIAR), whose aim is to extract information from music scores. Music scores contain both graphical and textual information, and for this reason, techniques are closely related to

---

A. Fornés (✉) • G. Sánchez

Computer Vision Center & Computer Science Department, Universitat Autònoma de Barcelona,  
Bellaterra, Spain  
e-mail: [aforres@cvc.uab.es](mailto:aforres@cvc.uab.es); [gemma@cvc.uab.es](mailto:gemma@cvc.uab.es)

graphics recognition and text recognition. Since music scores use a particular diagrammatic notation that follow the rules of music theory, many approaches make use of context information to guide the recognition and solve ambiguities. This chapter overviews the main Optical Music Recognition (OMR) approaches. Firstly, the different methods are grouped according to the OMR stages, namely, staff removal, music symbol recognition, and syntactical analysis. Secondly, specific approaches for old and handwritten music scores are reviewed. Finally, online approaches and commercial systems are also commented.

---

#### Keywords

Graphics recognition • Optical music recognition • Staff removal • Symbol recognition

---

## Introduction

Optical Music Recognition (OMR) is a classical area of interest of Document Image Analysis and Recognition (DIAR) that combines textual and graphical information. The recognition of music score images has been a very active research topic. OMR consists in the understanding of information from music scores (see an example in Fig. 22.1) and its conversion into a machine readable format. It allows a wide variety of applications [5] such as the edition of scores never edited, renewal of old scores, conversion of scores into braille, production of audio files and creation of collecting databases to perform musicological analysis.

## Music Scores: Problem Definition

Music scores are a particular kind of graphical document, which include text and graphic elements. Graphical information appearing in music scores follows some 2D structural rules and corresponds to music information such as notes, staves, and rests (silences). Textual information corresponds to lyrics and annotations such as title, author name, dynamic, and tempo markings, although their presence is not mandatory.

OMR has many similarities with Optical Character Recognition (OCR), because whereas OCR recognizes characters in text, OMR recognizes musical symbols in scores. It is nevertheless true that OMR belongs to graphics recognition because it requires the understanding of two-dimensional relationships, and music elements are two-dimensional shapes. The most common music symbols in a music score are notes, rests, accidentals, and clefs (see Fig. 22.2). Some terminologies used in music notation are the following:

- Staff: Five equidistant, horizontal lines where musical symbols are written down. They define the vertical coordinate system for pitches and provide horizontal direction for the temporal coordinate system.
- Attributive symbols at the beginning: Clef, time, and key signature.



**Fig. 22.1** Example of a handwritten music score

notes	note heads	rests	accidentals
whole note	○ note head for a whole/half note	— whole rest	♭ double flat
half note	● filled note head	— half rest	flat
quarter note		♪ quarter rest	natural
eighth note		♩ eighth rest	# sharp
sixteenth note	beam that joins three notes	♫ sixteenth rest	× double sharp

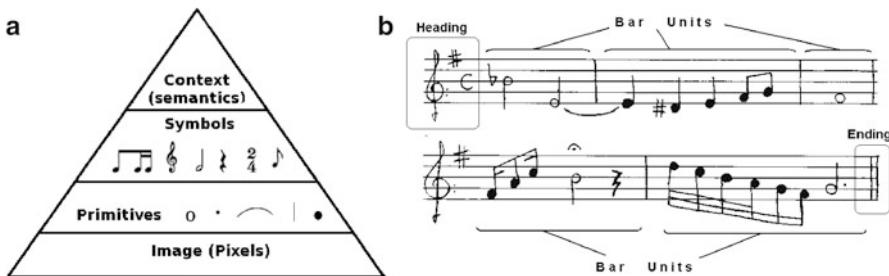
  

flags	clefs
↙ flag for a stem-up eighth note	♪ treble clef
↖ flag for a stem-up sixteenth note	♫ bass clef
↘ flag for a stem-down eighth note	alto clef
↗ flag for a stem-down sixteenth note	

stem                          note head                          flag

**Fig. 22.2** Common elements of music notation

- Bar lines: Vertical lines which separate every bar unit or measure.
- Notes and rests (pauses): Notes are composed of head notes, beams, stems, flags, and accidentals.
- Slurs: Curves that join musical symbols.
- Dynamic and tempo markings indicate how loud/soft the music should be played and the speed of the rhythm of a composition.
- Lyrics: The set of words that will sing the chorus or singers.



**Fig. 22.3** OMR: (a) Levels of an OMR system, (b) structure of a music score

Similarly to OCR systems (which include the pixel, character, word, and semantic level), the levels of the processed information of an OMR system are the image (pixels), graphical primitive, symbol, and context information level (see Fig. 22.3a). Context information helps to correct errors, and whereas dictionaries are commonly used in OCR, the formal music language theory is used in OMR.

A music score mainly consists in three blocks (see Fig. 22.3b): heading, bar units, and ending. Heading consists in the clef (alto, treble, or bass clef), the time signature (usually formed by two numbers that indicate the measure), and the key signature (flats, sharps, or naturals, which indicate the tonality of the music score). Bar units are the containers of music symbols (the amount of music symbols depends on the time signature). Finally, the ending is usually an ending measure bar and sometimes includes repeating marks.

## History and Importance of OMR

From its beginning in 1966, Optical Music Recognition has been an active research field in the domain of document analysis. The first works started in the Massachusetts Institute of Technology by Pruslin in [46], followed by the DO-RE-Mi system proposed by Prerau in [45]. Already in these previous works, the basic stages and challenges for printed music scores were established. First, Pruslin [46] explained what an OMR system should give as output notes, and in what order they are played, their time values or durations, a volume, tempo, and interpretation. Later, Prerau [45] added the need to have a modular system, as the set of music symbols is very large and his work was using only a subset of them. These works started with a first step which scans and binarizes music scores, stressed how staff lines hindered the recognition of symbols by overlapping them, and finished by recognizing music symbols and testing the operation of their system. Just after these works were presented, a musician called Kassler [32] gave his opinion about them. Under his point of view, it was already clear that a computer could convert printed music in a computer readable form after recognizing the music symbols appearing

in it. His worries were that these works only cope with a subset of all possible musical symbols and not the global notation. For that reason he appreciated the modular design of Prerau and its possibility to be increased with more symbols. Both works solved the problem for a subset of music symbols, but only Prerau's system was modular and extensible with more symbols. The problem with Pruslin's work was the preliminary step to remove horizontal and vertical lines in order to obtain isolated symbols. This process distorted so much some symbols that it was not possible to recognize them afterwards. In order to solve this problem, Prerau proposed a new process to detect symbols by using a split and merge process (see section "[Staff Detection and Removal](#)" for more details about these works).

During the decade of 1980s, several approaches appeared, basically focused on staff removal and the recognition of music symbols. Both Mahoney [38] and Roach and Tatem [53] proposed the recognition of music symbols by joining graphical primitives. Contrary, Clarke et al. [9] proposed the recognition of music symbols by analyzing the size and the bounding box of the symbols, very similar to the approach of Prerau. Mahoney also stressed the importance of the distinction between two kinds of music symbols: the ones describing "what" is to be played (e.g., notes, accidentals) and the ones describing "how" music has to be played (e.g., dynamics, tempo markings). Most of previous works were only focused on what is to be played as notes, clefs, and so on. In this decade, it is remarkable the work of Andronico and Ciampa [1], who proposed the use of grammars to guide the recognition. In this way, syntactic information could be used to solve ambiguities. Finally, it must be commented that in 1989, the robot Wabot-2 [39] was created. It was an almost real-time keyboard-playing robot able to read simple music scores and play the organ.

Since first works in 1970s and 1980s, the interest of OMR has grown considerably. Most works through 1990s focused on removing staff lines and recognizing music symbols with the help of grammars and syntactical rules [10, 19, 42], following the idea of Andronico in 1982. Thus, systems could take advantage of music notation theory in order to add syntactical information that guides the recognition and solves ambiguities and inconsistencies. Also during the first decade of 2000, many approaches appeared that dealt with the recognition of polyphonic and complex music scores, such as the work of Kato and Inokuchi [33], who could handle complex music notation such as two voices per staff, chords, shared noteheads, slurs, and pedal markings. Nowadays, the OMR systems of printed scores include effective algorithms (see Gamera and Aruspix systems [49]) to recognize and interpret the resulting 2D arrangement of symbols and precise formalisms for representing the results of interpretation. However, it is interesting to remark that the staff removal problem is still attracting the interest of researchers in OMR, with the proposal of many algorithms for removing staff lines [12, 26], especially for old and handwritten music scores [14, 16, 20, 56].

Contrary to printed scores, the recognition of handwritten music scores is still considered to be an open problem. To the best of our knowledge, the first attempt was performed by Ng in [43], who discussed the adaptation of OMR for printed music scores to handwritten ones. The recognition system for printed music

**Table 22.1** History and evolution of the optical music recognition approaches and the main type of research: *STF* staff removal, *SYM* music symbol recognition, *HW* handwritten music scores, *GR* grammars and rules, *ON* online OMR

Year	Work	Type	Year	Work	Type
1966	Pruslin [46]	STF, SYM	2001	Ng [43]	HW
1970	Prerau [45]	STF, SYM	2003	Pinto et al. [44]	SYM, HW
1982	Mahoney [38]	STF, SYM	2004	Fujinaga [26]	STF
1982	Andronico and Ciampa [1]	GR	2005	Rossant and Bloch [54]	SYM
1988	Clarke et al. [9]	STF, SYM	2005	Macé et al. [37]	ON
1988	Roach and Tatem [53]	STF, SYM	2006	Fornés et al. [20]	STF, HW
1989	Matsushima et al. [39]	GR	2006	Toyama et al. [57]	SYM
1991	Kato and Inokuchi [33]	STF, GR	2006	Homenda and Luckner [31]	SYM
1992	Carter and Bacon [8]	STF, SYM	2006	Pugin [47]	SYM
1993	Randriamahefa et al. [50]	STF, SYM	2006	Miyao and Maruyama [40]	ON
1993	Leplumey et al. [35]	STF	2008	Dalitz et al. [12]	STF
1993	Modayur et al. [42]	GR	2008	Pugin et al. [49]	SYM
1993	Fahmy and Blostein [19]	GR	2009	Cardoso et al. [14]	STF
1995	Baumann [4]	GR	2009	Escalera et al. [17]	SYM
1995	Miyao and Nakano [41]	SYM	2010	Dutta et al. [16]	STF
1995	Couasnon and Rétif [10]	GR	2010	Fornés et al. [21]	SYM, HW
1997	BainBridge and Carter [3]	STF, SYM	2010	Rebelo et al. [51]	SYM, HW
1999	Stückelberg and Doermann [55]	SYM	2011	Escalera et al. [18]	SYM, HW

was based in the detection of graphical primitives in order to regroup them and form music symbols. However, he concluded that this technique was not robust enough for dealing with music symbols because of the high variability in the handwriting styles. Since then, some other approaches for recognizing handwritten music symbols have appeared [21, 44, 51], although most of these systems are only able to recognize a small set of music symbols and not the whole music score.

Some interesting surveys of classical OMR can be found in [5, 30, 52], where several methods to segment and recognize symbols are reviewed. The evolution of OMR is shown in Table 22.1.

## Summary of the State of the Art

The research on Optical Music Recognition could be classified considering the different OMR stages, namely, preprocessing, staff removal, symbol recognition, and validation. Thus, in the next section some of the main works that have been proposed for each one of these stages are reviewed.

However, and since the above approaches must be adapted to the kind of document (e.g., typewritten, handwritten, old, online), section “[State of the Art in Different Domains](#)” is devoted to the specific works for dealing with old, handwritten, and online music documents.

---

## State of the Art in the Different Stages of OMR

### Introduction

The main stages of an Optical Music Recognition (OMR) system are the following: preprocessing, staff removal, symbol recognition, and validation (semantics). It is important to remark that in many cases, the separation between the stages is difficult because some OMR approaches join some of these stages together. For example, the authors of [10] and [33] propose a validation stage that integrates the symbol recognition module. Also, there are few approaches [39] and [47] that do not have any staff removal step.

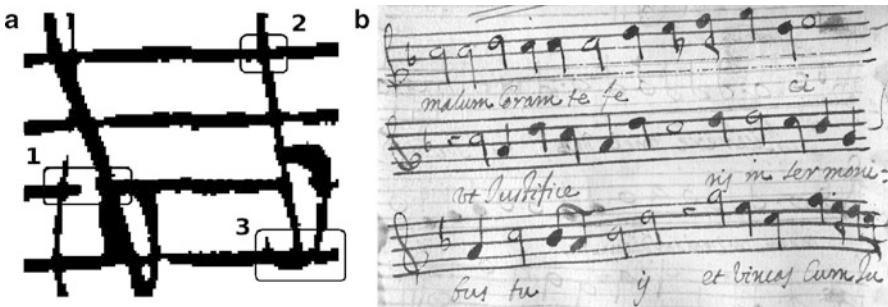
### Preprocessing

The preprocessing step usually consists in binarization, skew correction, and noise removal. Since the preprocessing techniques for music scores are usually generic document image analysis techniques, the reader is referred to [►Chap. 4](#) (Imaging Techniques in Document Analysis Processes) and [52] for more information.

### Staff Detection and Removal

Staff lines play a central role in music notation, because they define the vertical coordinate system for pitches, and provide a horizontal direction for the temporal coordinate system. The staff spacing gives a size normalization that is useful both for symbol recognition and interpretation: the size of musical symbols is linearly related to the staff space.

Most OMR systems detect and remove the staff from the image in order to isolate musical symbols and facilitate the recognition process. For this purpose, staff removal algorithms must cope with gaps, crossing symbols, overlapping symbols (see Fig. 22.4a), and in case of staff lines written by hand (see Fig. 22.4b), also with distorted lines.



**Fig. 22.4** Typical problems related to the removal of staff lines: 1. gap, 2. crossing symbol, 3. overlapping symbol. (b) Old Spanish music score from the seventeenth century, where the staff lines are written by hand. Notice that the staff lines are not equidistant nor parallel and lines have different widths

Typically, staff removal methods are based on projections and run-length analysis, contour-line tracking, or graphs. Next, some of the most common approaches are described.

### Projections and Run Lengths

The first group of methods are based on projections and/or run-length analysis, which are very fast to compute. Methods based on projections require horizontal staff lines. Thus, after deskewing the document, projections (Y projection) are used to recognize staff lines [44]. A defined threshold is usually used to select projections strong enough to be candidate staff lines. These candidates are searched to find groups of five equally spaced lines.

The first staff removal approach, the one proposed by Pruslin [46] in 1966, consisted in eliminating all thin horizontal and vertical lines, including many bare staff-line sections and stems. This results in an image of isolated symbols, such as noteheads and beams, which are then recognized using contour-tracking methods. As expected, this preprocessing step erases or distorts most music symbols.

The method proposed by Fujinaga in [26] detects staves by horizontal projections, but deskews each staff separately. Afterwards, black vertical runs larger than a threshold are removed and consider all remaining connected components with a considering width. Cui et al. [11] propose a method based on the detection and selection of region of interests (ROI). After rotation correction, the staff-line width is estimated using horizontal projection and the staff lines are removed correspondingly. Randriamahefa et al. [50] propose a more complex projections method. It consists in a vertical projection, projection filtering, local minima region finding (the peaks will probably correspond to the staff lines), horizontal projection of each local minima regions (to ensure that those peaks are certainly lines), and linking different peaks between them in order to build up the staff (a staff implies five staff lines). These techniques are usually very robust because even if these lines are bowed, skewed, or fragmented, the staff lines are always found. In most cases,

after the detection of the staff lines, the next step consists in estimating the width of the staff line and then erases the line segments that are smaller than a threshold (proportional to the estimated thickness).

Other methods also perform a run-length analysis. For example, Kato and Inokuchi [33] detect and extract staff lines using histograms, run lengths, and projections. Afterwards, the staff is analyzed (tracking from the left), eliminating short horizontal runs whose width is under a certain threshold. In the staff removal method of Dutta et al. [16], a staff-line segment is considered as a horizontal linkage of vertical black runs with uniform height. Then, the neighboring properties of a staff-line segment are used to validate it as a true segment. Similarly, in the approach of Clarke et al. [9], the staff lines are located by looking for long horizontal runs of black pixels. Then the neighborhood of each staff-line pixel is examined to determine whether a music symbol intersects the staff line at this point. Staves are located by the analysis of a single column of pixels near the left end of the system. Large blank sections indicate gaps between staff lines and are used to divide the image into individual staves. Complete staff separation is not always achievable, because parts of symbols belonging to the staff above or the staff below may be included.

The method proposed by Su et al. [56] first estimates the staff height and space by using the histogram of vertical run length. Afterwards, an initial staff line is modeled, which predicts the line direction and fits an approximate staff-line curve for each image. The fitted staff-line curve can be used to identify the actual location of staff lines on the image, where pixels belonging to staff lines are removed.

### Candidates Assemblage and Contour Tracking

These groups of methods usually detect candidate staff-line segments (usually from the skeleton of the image), which are afterwards analyzed using contour tracking.

Prerau [45] divides the process in fragmentation and assemblage. In the fragmentation step, the system scans along the top and bottom edges of staff lines to identify parts of symbols lying between, above, and below the staff lines (a new symbol fragment has begun whenever a significant change in slope is encountered). In the assemblage step, these symbol fragments are connected if the two symbol fragments (separated by a staff line) have horizontal overlap. One disadvantage of this technique is that symbols which merge with staff lines do not always have horizontal overlap, so with this method, symbols would keep disconnected when they should be connected.

In the work of Roach and Tatem [53], the segmentation stage detects staff lines using measures of line angle and thickness. A window is passed over the image to compute a line angle for every black pixel. The line angle is measured from the center of the window to the furthest black pixel in that window; this furthest black pixel is chosen so that the path from it to the center does not cross any white pixels. To detect staff lines, a large window radius is used. This causes covered staff-line sections to be labelled with a horizontal line angle despite the interference of the superimposed musical symbols. Once a line angle has been determined, a line

thickness can be measured. These two measurements, combined with adjacency information, are used to identify horizontal lines.

The systems proposed by Mahoney [38], Fornés et al. [20] and Dalitz et al. [12] create staff-line candidates from the image. Afterwards, these candidates are analyzed, and some rules are applied (allowable thicknesses, orientations, lengths, and gap lengths) in order to discard or join them. The method removes only those parts of the line that do not overlap other symbols. Good extraction of staff lines is achieved, although more work is needed for dealing with line-region overlap.

In the prototype for handwritten scores presented by Ng in [43], the skeleton of the binary image is obtained in order to transform musical symbols into a set of interconnected curved lines. Then, junction and termination points are extracted from the skeleton representations. In the staff detection phase, all horizontal line segments are parsed to determine if they belong to a part of a staff-line using basic notational syntax and an estimated staff line height.

### **Graph Path Search**

The last group of staff removal methods propose the creation of a graph.

Carter and Bacon (see [8]) propose a system for segmentation that uses processing based on a Line Adjacency Graph (LAG). Because the detection of places where a thin portion of a symbol tangentially intersects a staff line is difficult, most methods create gaps in symbols. Carter proposes a LAG-based analysis that successfully identifies such tangential intersections of symbols with staff lines. In addition, the system locates staff lines despite the image rotation of up to 10° and copes with slight bowing of staff lines and with local variations in staff-line thickness.

Leplumey et al. [35] present a method based on a prediction-and-check technique to extract staves, even detecting lines with some curvature, discontinuities, and inclination. After determining thickness of staff lines and interlines using histograms and run lengths, some hypotheses on the presence of lines are done grouping compatible observations into lines. Afterwards, an interpretation graph is used for joining segments to obtain staff lines. This method process allows little discontinuities thanks to the use of a local predicting function of the staff inclination.

Cardoso et al. [14] propose a graph-theoretic framework where the staff line is the result of a global optimization problem. The staff-line algorithm uses the image as a graph, where the staff lines result as connected paths between the two lateral margins of the image. A staff line can be considered a connected path from the left side to the right side of the music score. The main cycle of the methodology consists in successively finding the stable paths between the left and right margins, adding the paths found to the list of staff lines, and erasing them from the image. To stop the iterative staff-line search, a sequence of rules is used to validate the stable paths found.

### **No Staff Removal**

Finally, there is a small group of approaches that do not remove staff lines, because they need a real-time OMR system [39] or because they would like to avoid the

**Table 22.2** Staff removal comparison of the main group of methods. It shows their performance when dealing with overlapping symbols with staff lines, grouping five staff lines into a staff, and the (estimated) speed of the algorithm. Note that – means poor performance and ++ means very good

Method	Works	Overlapping symbols-staff	Staff-lines grouping	Speed
Projections, histograms, and run lengths	[9, 34, 46], [26, 44, 50], [11, 16, 33, 56]	–	++	++
Candidates assemblage and contour tracking	[20, 38, 53], [12, 43, 45]	+	+	+
Graph path search	[8, 14, 35]	++	+	–

segmentation problems that the staff removal may cause [47]. The Wabot-2 robot of Matsushima et al. [39] performs a template matching without removing staff lines: staff lines are detected and used to normalize the image, to determine the score geometry, and also to restrict the search area for music symbols (then, the recognition of musical symbols must learn symbols which include segments of staves). Staff lines are detected in hardware by a horizontal line filter, tolerating some skew. Where five equally-space lines are found, a staff is deemed to exist. Normalization parameters include staff location, staff inclination, area covered by staff, and notehead size. Afterwards, the image of each staff is normalized according to these parameters.

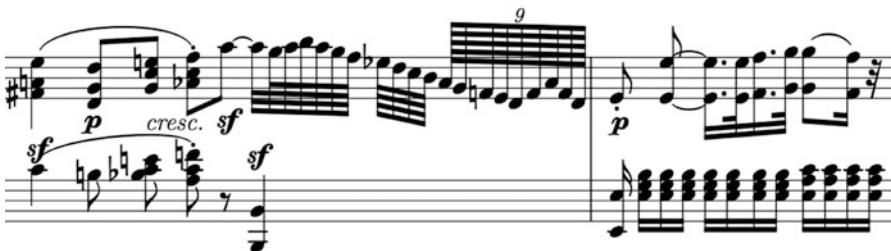
## Summary

A comparison of the different kind of staff removal algorithms is shown in Table 22.2. Systems based on projections and run lengths are fast to compute, but since they usually split symbols that were touching the staff lines, the posterior recognition stage must deal with broken symbols. Contrary the performance of contour-tracking and graph-based techniques is usually better, especially concerning overlapping symbols with staff lines.

Concerning ground truthing for testing the staff removal algorithms, Dalitz et al. [12] have created distorted images from printed music scores. In order to test these algorithms in handwritten music scores, the reader is referred to the CVCMUSCIMA database [24]. Finally, it must be said that a comparative study of four staff removal algorithms can be found in [12]. In addition, two staff removal competitions had been recently organized in ICDAR (International Conference on Document Analysis and Recognition) [22] and GREC (International Workshop on Graphics Recognition) [23]. In both cases, the conclusion is that there is no algorithm that performs best in all cases (depending on the distortions to deal with, one algorithm performs better than others), showing that there is still room for improvements.



**Fig. 22.5** Composition of musical symbols. (a) Two equivalent music symbols. (b) Combinations of 8th and 16th notes



**Fig. 22.6** Compound notes extracted from Sonata “Pathétique” from Beethoven

## Music Symbol Extraction and Classification

The classification of music symbols could be treated as a symbol recognition problem, and therefore, some symbol recognition techniques have been applied. However, simple music symbols are usually combined following certain rules for creating complex compound music symbols (see Fig. 22.5), and for this reason, many techniques based on the analysis of graphical primitives and their combinations have been proposed. Thus, they can cope with the almost infinite variability of compound symbols (see Fig. 22.6).

The main techniques have been classified into different groups, which are described next.

### Template Matching

Template matching means comparing the different region object candidates with the templates that represent the different music symbols. Since it is computationally expensive and sensitive to variations in shape, it has been applied in few cases. For example, Pruslin [46] uses contour tracking to describe connected binary image regions which remain after deleting horizontal and vertical lines. Classification depends both on stroke properties and on inter-stroke measurements (a method for template matching using contour strokes is developed). Toyama et al. [57] present a symbol recognition method for printed piano music scores with touching symbols. The symbol candidates are detected by template matching, and from these candidates, correct symbols are selected by considering their relative positions and mutual connections. Touching primitives are detected using coherence check.

The Wobot-2 robot [39] recognizes the musical symbols (with staff lines) according to a two-level hierarchy: the upper level, corresponding to the recognition

of staff lines, noteheads, and bar lines, is implemented in hardware and the lower level in software. The search is performed using hardware-implemented template matching.

### Simple Operations

This kind of methods bases the recognition of music symbols with simple operations, such as the analysis of bounding boxes, projections, or morphological operations.

Prerau [45] and Clarke et al. [9] use the relative symbol size for an initial classification. In [45], the dimensions of the bounding box are used to look up a list of possible matches (there is a precomputed table containing the standard areas of each symbol in a height/width space). Typically there are three to five possible matches for each symbol, so heuristic tests are used to distinguish symbols that overlap in the height/width space, taking advantage of the syntax, redundancy, position, and feature properties of each symbol type. Contrary, Clarke et al. [9] do not use any lookup table. Instead, pixels in few particular rows and columns of the symbol image are examined.

Lee and Choi [34] use projection methods to recognize staff lines, bar lines, notes, and rests. After detecting the staff lines, projections are used to find bar lines. Notes are recognized using X and Y projections from a small window around the symbol. Characteristic points in the projections are used for classification, where they are compared with the stored projections for known symbols. The main disadvantage of this method is that it is rotation sensitive.

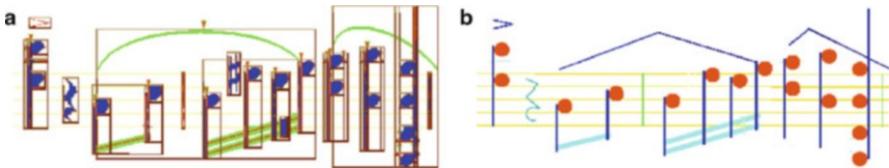
In the works of Bainbridge and Carter [3] and Carter and Bacon [8], objects are classified according to the bounding-box size and according to the number and organization of their constituent sections.

The system proposed by Modayur et al. [42] is composed of two modules: the low-level vision module uses morphological algorithms for symbol detection; the high-level module context information to validate the results. Because morphological operations can be efficiently implemented in machine vision systems, the recognition task can be performed in near real time.

### Joining Graphical Primitives

These systems extract graphical primitives (e.g., noteheads, stems beams, and flags) and combine them in order to form music symbols, such as notes, chords, and beamed note sequences.

Mahoney [38] does not use context information for the recognition of primitives, but it is used to infer musical symbols from the relationships between the different kind of primitives. After extracting line primitives, dot primitives are processed and removed. All measures of distance are normalized on staff-line and staff-space thickness. Sample line parameters are principal direction, angle, thickness, length, and maximum permitted gap. Sample region parameters are mass, width, height, and inclination angle. This process is initially used in an interactive mode to add or modify object descriptions.



**Fig. 22.7** Classification of musical symbols performed in [43]. (a) Oversegmentation of symbols, (b) recognized primitives

In the system proposed by Roach and Tatem [53], knowledge about music notation is represented in a rule-based system, which is applied starting with the earliest steps of symbol segmentation and recognition. The primitives recognized are circular blobs, circles, horizontal lines, non-horizontal line segments, and arcs (clefs are not recognized). Primitive identification is coded as several steps, using context information in the last step. Since notehead detection is extremely difficult in handwritten music scores, a general-purpose blob detector does not work. Thus, in [20] and [53], noteheads are searched in constrained locations: first, vertical lines are located, and then a thickness measure is used to test for wide spots at the ends of each potential stem; if there is a wide spot (whose circularity is under a certain threshold), it is accepted as a notehead. Also, Kato and Inokuchi [33] describe a primitive extraction stage which analyzes the size of lines, blobs, and circles in order to detect noteheads and stems candidates that will be analyzed in the next stages. Similarly, Ng exposes in [43] a prototype for printed scores, where a combination of edges, curvature, and variations in relative thickness and stroke direction is applied to oversegment symbols into lower-level graphical primitives (lines, curves, and ellipses). Afterwards, these primitives will be joined to form music symbols. An example of the detected primitives is shown in Fig. 22.7.

A more complex approach is proposed by Miyao and Nakano [41] for the extraction of heads and stems in printed piano scores. After extracting all region candidates to be stems or heads, a three-layer Neural Network is used to identify heads; the weights for the network are learned by the back propagation method. In the learning, the network learns the spatial constraints between heads and surroundings rather than the shapes of heads. Afterwards, the networks are used to identify the head candidates. Finally, the stem candidates touching the detected heads are extracted as true stems.

### Statistical Approaches

The authors of [51] perform a comparative study of four classification methods for musical primitives: Neural Networks (NN), Support Vector Machines (SVM), k-Nearest Neighbor (k-NN), and Hidden Markov Models (HMM). In the first three methods, the input features are the raw pixels of the image (resized at  $20 \times 20$ ), whereas for HMM, high-level features (see [47]) are extracted from a sliding window. Results show that the best classification method is SVM, closely followed by k-NN.

## Symbol Descriptors

This kind of methods uses symbol descriptors (see ►Chap. 16 (An Overview of Symbol Recognition)) for the recognition of music symbols. For example, Homenda and Luckner [31] present a system for recognizing five different classes of music symbols. They compare methods based on centroids, Zernike moments, and decision trees with split decision. They propose decision trees based on the linear combination of 278 basic features (e.g., histograms, density, symbol direction) and use Principal Component Analysis for improving the final recognition rate. In [17] and [18], Escalera et al. describe two symbol recognition methods that can be applied to hand-drawn music symbols. In both methods, the descriptor defines a probability density function of the shape of the symbol using a rectangular [17] or circular grid [18]; and Adaboost learns the discriminative features that better split symbol classes. Also, Fornés et al. describe a rotation symbol recognition method for recognizing hand-drawn music symbols [21]. The method extracts column sequences of feature vectors from different orientations. Afterwards, the Dynamic Time Warping algorithm is used for finding the best matching between the two symbols to be compared.

## Structural and Syntactical Approaches

The last set of methods (grammars, graphs, fuzzy modeling, etc.) uses context information in order to solve ambiguities. Most of them integrate the recognition of symbols with the validation (syntactic and semantic) stage.

Stückelberg and Doermann [55] propose a probabilistic framework for the recognition of printed scores. The modeling structure is similar to a stochastic attribute grammar, where local parameters are estimated using Hidden Markov Models (HMM). It also uses the Hough Transform and a low-band filter to locate lines and noteheads of note groups.

Randriamahefa et al. [50] propose an attributed graph, which is constructed from the skeleton of the image: the graph nodes correspond to segments and the graph arcs to the links between segments. After constructing the graph, symbols are classified in symbols including notes with black heads (there is at least one segment having a distance to the contour exceeding a certain threshold) and the others. Half notes are detected if there is a stem with a little loop in its extremes.

Rossant and Bloch [54] propose a system based on a fuzzy modeling of symbol classes and music writing rules. First, the individual analysis process (based on pattern matching) performs the segmentation of the objects and the correlation with symbol models stored in a reference base. Then, the fuzzy modeling part provides for each classification hypothesis a possibility degree of membership to the class. It also introduces a fuzzy representation of the common music writing rules by expressing graphical and syntactic compatibility degrees between the symbols. The fuzzy modeling of symbol classes allows to deal with imprecision and variations of symbol shapes.

**Table 22.3** Recognition of music symbols. Comparison of the main kind of methods, showing their robustness to noise, the estimated speed, and the robustness to different types, fonts, or handwriting styles. Note that – means poor performance and ++ means very good

Method	Works	Appearance robustness	Noise robustness	Speed
Template matching	[39, 46, 57]	--	--	–
Bounding box, projections morpholog. operations	[3, 9, 45], [8, 34, 42]	--	--	++
Joining graphical primitives	[38, 41, 53], [20, 33]	+	+	+
Statistical	[47, 51]	+	++	–
Symbol descriptors	[17, 31], [18, 21]	++	++	+
Syntactic and structural	[50, 54, 55]	++	++	–

## Summary

Table 22.3 shows a comparison of the different approaches for the extraction and classification of music symbols. It compares the estimated speed of the system, the robustness to noise, and the sensitivity to the variability of the appearance (e.g., font of the publisher, handwriting variabilities).

Since template matching is computationally very expensive, it can be effectively used when two constraints are satisfied: first, no real-time algorithm is needed (or there is dedicated hardware for this), and second, all the music symbols belonging to the same class look very similar. In contrast, methods based on simple operations are very fast to compute. However, both kinds of methods (template matching and simple operations) are very sensitive to noise. Consequently, the performance depends on the font of the publisher and will not work in handwritten scores.

Methods based on joining graphical primitives are more robust to the variability of symbols' shape, but they cannot deal with symbols that cannot be discomposed into primitives (e.g., music clefs). The recognition of these symbols can be solved with the application of symbol descriptors. However, the best results are usually obtained from methods that make use of context information (rules) such as syntactic and structural approaches (although this usually implies decreasing the speed of the system).

## Syntactical Analysis and Validation

Rules on music notation make the recognition task easier, because the information of two-dimensional relationships between musical symbols can be captured in a syntactic description of music. For that reason, most authors define grammars describing the organization of music notation in terms of music symbols. Some authors [1, 45] use two different grammar levels: lower-level grammars for music symbols (with terminal geometric figures such as dots, circles, and lines and adjacency operations such as above, below, right of, etc.) and high-level grammars for music sentences (larger units with measures containing music symbols).

## Grammars

The first group of methods use a grammar to guide the recognition. In [39] the robot uses a musical grammar to correct errors such as missing beats or contradictory repeat signs. Examples of constraints applied to three-part organ music are the following: a fat double bar appears only at the end of each part, a clef always appears right at the beginning of each staff, the number of beats in each measure should match the time signature, etc.

In [10], a grammar is formalized to work in the image level to produce an accurate segmentation and thus accurate recognition. Whereas most grammars are usually used at a high level to validate the structured document, the system proposed uses context information (syntax) to control the entire recognition process.

## Rules or Graph Grammars

The second group of methods is based on rules or constraints. Modayur et al. [42] propose a high-level reasoning module. The system utilizes prior knowledge of music notation to reason about spatial positions and spatial sequences of recognized symbols. This module is composed of a connected component analysis and a reasoning module (which verifies if every musical symbol accomplishes its own constraints). The high-level module also employs verification procedures to check the veracity of the output of the morphological symbol recognizer.

Kato and Inokuchi [33] describe a sophisticated top-down architecture. It uses a collection of processing modules which represents information about the current bar of music at five levels of abstraction: pixel, primitives, music symbols, meaning (pitch and duration of notes), and context information (interpretations). The four processing modules (primitive extraction, symbol synthesis, symbol recognition, and semantic analysis) have recognition and verification units. The primitive extraction module contains units for recognizing stems, beams, and noteheads. Hypothesized primitives are removed from the pixel image. Unacceptable hypotheses are rejected at higher layers and are sent back to lower layers for further processing. Symbol recognition proceeds one measure at a time and consists on pattern processing and semantic analysis (using context information), required for solving ambiguities of complex notations.

Baumann proposed in [4] the DO-RE-MI-DI++ system, which allows multiple score layouts and polyphonic input. This system allows the definition of syntactic and semantic knowledge by specifying graph-grammar rules. Similarly, Fahmy and Blostein [19] present a graph grammar for recognizing musical notation, where the input graph to the grammar is constructed as a set of isolated attributed nodes representing the musical symbols. The grammar itself forms the edges representing the significant associations between the primitives that are necessary in determining the meaning. Although the proposed approach relies on the ability to control the order of application of the productions, there may be some portions of the grammar in which the order does not need to be specified.

# **State of the Art in Different Domains**

## Introduction

The difficulties found in the recognition of music scores strongly depend on the kind of music scores. For example, the recognition of old documents has to cope with paper degradation, whereas handwritten scores mean dealing with the variability in the handwriting style. In case of old handwritten music scores (see an example in Fig. 22.8), the systems must cope with both problems.

Next, the main approaches for the recognition of old scores, handwritten scores, and also online handwritten music scores are reviewed.

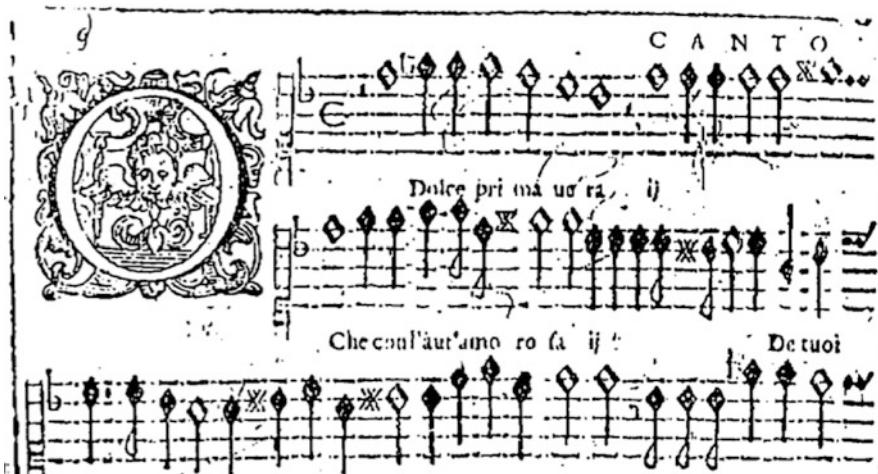
# **Old Handwritten Music Scores**

There are few works concerning the recognition of old music scores. They are briefly described next.

Bainbridge, Carter et al. propose a system for recognizing early music [2, 3, 8]. The system described in [7] is used to recognize old music scores, corresponding to madrigals (see Fig. 22.9) notated in *White Mensual Notation*. Symbols are correctly segmented and an early classification stage has been implemented.



**Fig. 22.8** Old handwritten music score



**Fig. 22.9** Example of an old madrigal, extracted from [7]

The approach first removes the staff lines using Line Adjacency Graph (LAG), which has been explained in the staff removal subsection. Thus, the resulting image contains music symbols or connected components that belong to music symbols. In the classification stage, these elements are classified according to the bounding-box size and the number and organization of their constituent sections. The author states that if there are overlapping or superimposed symbols, another algorithm will be required.

Pinto et al. [44] proposes an OMR method to recognize ancient musical scores. This system copes with specific notation of ancient documents. After the preprocessing stage, the segmentation module divides the music sheet in staff lines, bars, and musical symbols. The staff lines are removed using horizontal projections. Bar lines are located using vertical projections, and objects are segmented using morphological operations and connectivity analysis. The recognition process is based on a graph structure of classifiers, divided into two steps: feature extraction and classification. The method includes the construction of a class hierarchy associated with recognizers that distinguish between clusters of classes based on selected object features. Then, a method for the search of optimal graph hierarchy (manual and automated) and for the classification algorithms themselves is proposed. Finally, the reconstruction stage is needed to relate the recognized symbols with each other and with its staff lines and bars, creating the final description of the music. The system proposed obtains high performance results (97 % of accuracy).

Fornés et al. [20] propose a method for the recognition of graphical primitives in old handwritten music scores. After removing the staff lines using contour tracking, the system detects vertical lines, bar lines, and filled head notes using median filters and morphological operations. The system also recognizes music clefs using Zernike moments. The authors suggest that a grammar should be incorporated in order to cope with the important amount of false positives obtained.

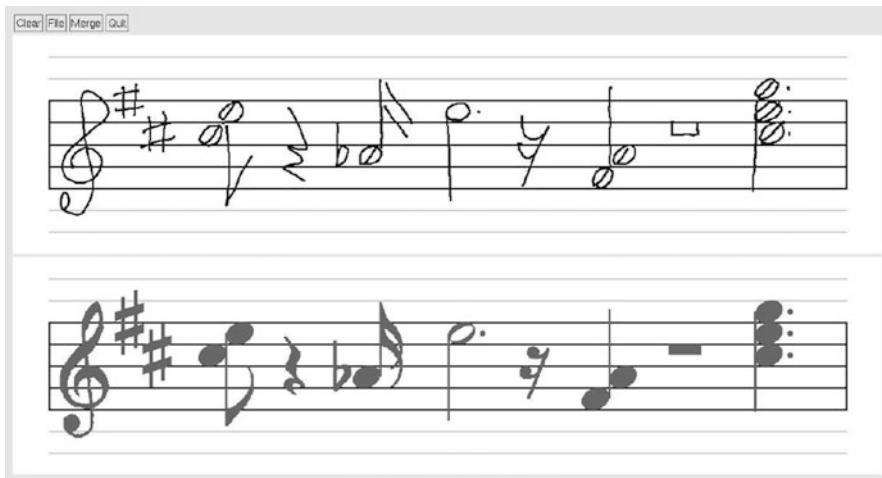
An approach for OMR in printed scores from the 16th to 17th is presented by Pugin [47]. The system consists in a segmentation-free approach based on Hidden Markov Models (HMM). They do not remove the staff lines, and they do not perform any segmentation neither. The goal is to avoid segmentation problems and irregularities. The modeling of symbols on the staff is based on low-level simple features, which include the staff lines. For feature extraction, they use a sliding window as in speech recognition, extracting the following six features for each window: the number of connected black zones, the gravity centers, the area of the largest black element and the smallest white element, and the total area of the black elements in the window. Concerning the HMM, the number of states used matches as closely as possible the width in pixels of the symbol. The training is performed with the embedded version of the Baum-Welch algorithm. For every training iteration, each staff is used once to adapt the models corresponding to the symbols of which the staff is made. The author shows that with the use of these features and HMM, good recognition rates are obtained.

Finally, an interesting comparison of two OMR approaches (Gamut and Aruspix) applied to ancient scores can be found in [49], where the authors demonstrate that paper degradation affects to the final performance. They also perform an evaluation of binarization techniques for OMR in [48].

## Modern Handwritten Music Scores

Ng exposes in [43] a prototype for printed scores, followed by a prototype for handwritten ones, discussing the limitations of the first one for handwritten scores processing. From the skeleton of the image, staff lines are detected and removed. Then, a combination of edges, curvature, and variations in relative thickness and stroke direction is used to perform further subsegmentation and segregate the writings into lower-level graphical primitives (lines, curves, and ellipses). Afterwards, primitives are classified using a k-NN classifier. Each terminal point is parsed to search for any other nearby terminal points which are collinear with the current segment or following a polynomial extrapolation from the terminal points of the current segment. The author comments that a tracing routine using a database of isolated handwritten musical symbols would improve the classification stage. After the classification phase, these subsegmented primitives are regrouped (applying basic syntactic rules) to form musical symbols. Contextual ambiguities are resolved using relative positions of primitives in the staff and between primitives. The reconstruction module offers an intermediate stage where extensive heuristic, musical syntax, and conventions could be introduced to enhance or confirm the primitive recognition and regroupings. Unluckily, no recognition rates are shown in the recognition of handwritten scores.

Rebelo et al. [51] tested the performance of four classification methods: Neural Networks (NN), Support Vector Machines (SVM), k-Nearest Neighbor (k-NN), and Hidden Markov Models (HMM) on 50 handwritten music scores. These methods have been described in the previous section. Results show that SVM and k-NN obtain the best results.



**Fig. 22.10** The online system proposed by Miyao and Maruyama [40]

### OnLine Music Scores

In the last decade, several online music recognition systems have appeared, although the amount of music symbols that can be recognized is still limited. George [27] presents a system based on Neural Networks, which resizes the input symbol, divides the symbol into 25 regions, and computes the density in each one. Afterwards, a multilayer perceptron is used for classification. For evaluating the system, the author has created a database of 4,188 music symbols, containing 20 different music symbols written by 25 people. The system obtains a recognition rate of about 80 %, using the 50 % of the symbols for training and the remaining 50 % for testing.

Macé et al. [37] propose an online system with three main components: a framework pen-based interaction, a formalism to describe the composition of documents, and a handwritten stroke parser. The formalism is based on context-free grammars, with chronological information, spatial structure, and composition rules. The handwritten stroke parser drives the different recognizers (e.g., alteration, stem, head) with empirical heuristics or Neural Networks; however, the authors do not provide detailed information about the recognizers. The framework allows the user to validate or reject an answer from the parser. The system can recognize whole, half, and quarter notes, accidentals, clefs, rests, and bar lines. Unluckily, the authors do not provide any recognition rates.

Miyao and Maruyama [40] present an online symbol recognition system which combines two kind of features for recognizing strokes: time-series data and hand-drawn image features. Then, features are combined to identify the music symbol. An eight-direction Freeman Chain Code is used to represent the time-series data of the stroke, and for matching the codes, string edit distance based on Dynamic Programming is used. For the computation of the image features, the image of the stroke is divided into  $8 \times 8$  regions, and the directional feature of each region is

calculated. Then, a Support Vector Machine is used for the classification. Results of both classifiers are also combined using a Support Vector Machine. Afterwards, the combination of specific strokes for each music symbol is consulted in a predefined table. To allow a stroke distortion, some music symbols have several possibility combinations of strokes. The proposed method reaches a recognition rate of 98 %. An example of the output of their system is shown in Fig. 22.10.

---

## Conclusion

In this chapter, the main works in Optical Music Recognition have been reviewed. Firstly, the music notation and the history of the music recognition research field have been overviewed. Then, the main approaches in each stage of an Optical Music Recognition system have been described. The OMR stages consist of preprocessing, staff removal, symbol recognition, and validation. Finally, the main works concerning the recognition of old music scores as well as handwritten (off-line and online) music scores have been reviewed.

The research could be considered in a mature state concerning the recognition of printed scores, including staff removal, primitives, music symbol recognition, and, also, the validation of the music score.

Open problems are still the recognition of old scores, handwritten music scores, as well as complex music scores (scores with many voices, chords, etc.). In fact, current approaches can effectively deal with simple symbols, but the recognition of complex symbols is still a difficult task. Similarly, online approaches are still not able to effectively recognize complex symbols, and this is probably one of the main reasons why composers prefer to work on paper document yet. In addition, the current performance of the OMR systems is not as good as desired, and therefore they are still not applicable for productive use.

For these reasons, promising approaches are the ones that take into account the context information, the structure of the music score, and the music notation. In this sense, grammars have shown to be a good choice for guiding the recognition, discarding false detections, and helping with ambiguities. In addition, learning-based methods (such as Hidden Markov Models and Neural Networks) are promising research directions for coping with the variability of handwritten music scores.

---

## Consolidated Software

There are several commercial OMR systems, such as:

- SharpEye: <http://www.visiv.co.uk>
- PhotoScore: <http://www.neuratron.com/photoscore.htm>
- SmartScore: <http://www.musitek.com>
- Audiveris (open source): <http://audiveris.kenai.com>

In most systems, the recognition rates significantly decrease when the document is noisy, degraded, or with low resolution. As far as the authors know, PhotoScore is the only one that can deal with handwritten music scores.

Some OMR systems are also available in the academic field. The Gamera framework [15] has been frequently used in OMR applications [12–14, 16], although it has been designed for building document analysis applications. The authors of [49] compare two systems: Gamut (a Gamera application) and Aruspix. Both have been applied to ancient scores. The authors conclude that although Aruspix HMM models outperform the Gamut k-NN classifiers, experiments show that paper degradation affects to the performance of both systems.

---

## Cross-References

- ▶ [An Overview of Symbol Recognition](#)
  - ▶ [Analysis and Interpretation of Graphical Documents](#)
  - ▶ [Analysis of the Logical Layout of Documents](#)
  - ▶ [Continuous Handwritten Script Recognition](#)
  - ▶ [Graphics Recognition Techniques](#)
  - ▶ [Handprinted Character and Word Recognition](#)
  - ▶ [Online Handwriting Recognition](#)
  - ▶ [Text Segmentation for Document Recognition](#)
- 

## References

1. Andronico A, Ciampa A (1982) On automatic pattern recognition and acquisition of printed music. In: Proceedings of the international computer music conference, Venice, pp 245–278
2. Bainbridge D, Bell T (1996) An extensible optical music recognition system. In: Proceedings of the nineteenth Australasian computer science conference, Melbourne, pp 308–317
3. Bainbridge D, Carter N (1997) Automatic reading of music notation. In: Bunke H, Wang PSP (eds) Handbook of character recognition and document image analysis, Chapter 22. World Scientific, Singapore, pp 583–603
4. Baumann S (1995) A simplified attributed graph grammar for high-level music recognition. In: International conference on document analysis and recognition, Montreal, vol 2. IEEE Computer Society, Los Alamitos, pp 1080–1083
5. Blostein D, Baird HS (1992) A critical survey of music image analysis. In: Baird HS, Bunke H, Yamamoto K (eds) Structured document image analysis. Springer, New York, pp 405–434
6. Bruder I, Ignatova T, Milewski L (2004) Integrating knowledge components for writer identification in a digital archive of historical music scores. In: Proceedings of the 4th ACM/IEEE-CS joint conference on digital libraries (JCDL), Tucson, AZ, USA, pp 397–397
7. Carter NP (1992) Segmentation and preliminary recognition of madrigals notated in white mensural notation. *Mach Vis Appl* 5(3):223–229
8. Carter NP, Bacon RA (1992) Automatic recognition of printed music. In: Baird H, Bunke H, Yamamoto K (eds) Structured document image analysis. Springer, Berlin Heidelberg, pp 456–465
9. Clarke A, Brown B, Thorne MP (1988) Inexpensive optical character recognition of music notation: a new alternative for publishers. In: Computers in music research conference, Bailrigg, Lancaster, pp 84–87

10. Coüasnon B, Rétif B (1995) Using a grammar for a reliable full score recognition system. In: International computer music conference, Banff, Canada, pp 187–194
11. Cui J, He H, Wang Y (2010) An adaptive staff line removal in music score images. In: IEEE 10th international conference on signal processing (ICSP), Beijing. IEEE Computer Society, pp 964–967
12. Dalitz C, Drotetboom M, Pranzas B, Fujinaga I (2008) A comparative study of staff removal algorithms. *IEEE Trans Pattern Anal Mach Intell* 30(5):753–766
13. Dalitz C, Michalakis GK, Pranzas C (2008) Optical recognition of psaltic byzantine chant notation. *Int J Doc Anal Recognit* 11(3):143–158
14. dos Santos Cardoso J, Capela A, Rebelo A, Guedes C, Pinto da Costa J (2009) Staff detection with stable paths. *IEEE Trans Pattern Anal Mach Intell* 31(6):1134–1139
15. Drotetboom M, MacMillan K, Fujinaga I (2003) The gamera framework for building custom recognition systems. In: Proceedings of the symposium on document image understanding technologies, Greenbelt, Maryland (USA), pp 7–11. Citeseer
16. Dutta A, Pal U, Fornés A, Lladós J (2010) An efficient staff removal approach from printed musical documents. In: International conference on pattern recognition, Istanbul. IEEE Computer Society, Istanbul, Turkey, pp 1965–1968
17. Escalera S, Fornés A, Pujol O, Radeva P, Sánchez G, Lladós J (2009) Blurred shape model for binary and grey-level symbol recognition. *Pattern Recognit Lett* 30(15):1424–1433
18. Escalera S, Fornés A, Pujol O, Lladós J, Radeva P (2011) Circular blurred shape model for multiclass symbol recognition. *IEEE Trans Syst Man Cybern B Cybern* 41(2):497–506
19. Fahmy H, Blostein D (1993) A graph grammar programming style for recognition of music notation. *Machine Vis Appl* 6:83–99. Springer
20. Fornés A, Lladós J, Sánchez G (2006) Primitive segmentation in old handwritten music scores. In: Liu W, Lladós J (eds) Graphics recognition: ten years review and future perspectives. Volume 3926 of lecture notes in computer science, pp 279–290. Springer, Berlin Heidelberg
21. Fornés A, Lladós J, Sánchez G, Karatzas D (2010) Rotation invariant hand drawn symbol recognition based on a dynamic time warping model. *Int J Doc Anal Recognit* 13(3):229–241
22. Fornés A, Dutta A, Gordo A, Lladós J (2011) The icdar 2011 music scores competition: staff removal and writer identification. In: International conference on document analysis and recognition (ICDAR), Beijin, China. IEEE Computer Society, pp 1511–1515
23. Fornés A, Dutta A, Gordo A, Lladós J (2012) The 2012 music scores competitions: staff removal and writer identification. In: Kwon Y-B, Ogier J-M (eds) Graphics recognition. New trends and challenges. Lecture notes in computer science, vol 7423. Springer, Berlin/Heidelberg
24. Fornés A, Dutta A, Gordo A, Lladós J (2012) Cvc-muscima: a ground truth of handwritten music score images for writer identification and staff removal. *Int J Doc Anal Recognit* 15(3):243–251
25. Fornés A, Lladós J, Sanchez G, Bunke H (2012) Writer identification in old handwritten music scores. In: Papaodysseus C (ed) Pattern recognition and signal processing in archaeometry: mathematical and computational solutions for archaeology. IGI Global, Hershey, Pennsylvania, USA, pp 27–63
26. Fujinaga I (2004) Staff detection and removal. In: George S (ed) Visual perception of music notation. Idea Group, IRM Press, Hershey PA, USA, pp 1–39
27. George S (2003) Online pen-based recognition of music notation with artificial neural networks. *Comput Music J* 27(2):70–79
28. George S (2005) Visual perception of music notation: on-line and off-line recognition. IRM Press, Hershey PA, USA
29. Gordo A, Fornés A, Valveny E (2013) Writer identification in handwritten musical scores with bags of notes. *Pattern Recognit* 46:1337–1345. doi:<http://dx.doi.org/10.1016/j.patcog.2012.10.013>
30. Homenda W (2005) Optical music recognition: the case study of pattern recognition. In: Kurzynski M, Puchala E, Wozniak M, Zolnirek A (eds) CORES, advances in soft computing, Rydzyna Castle, Poland. Editorial: Springer, Berlin Heidelberg, vol 30. Springer, pp 835–842

31. Homenda W, Luckner M (2006) Automatic knowledge acquisition: recognizing music notation with methods of centroids and classifications trees. In: International joint conference on neural networks, Vancouver, Canada. IEEE Computer Society, pp 3382–3388
32. Kassler M (1972) Optical character recognition of printed music: a review of two dissertations. *Perspect New Music* 11(2):250–254
33. Kato H, Inokuchi S (1991) A recognition system for printed piano music using musical knowledge and constraints. In: Baird HS, Bunke H, Yamamoto K (eds) Structured document image analysis. Springer, Berlin Heidelberg, pp 435–455
34. Lee MW, Choi JS (1985) The recognition of printed music score and performance using computer vision system (in Korean and English translation). *J Korea Inst Electron Eng* 22(5):429–435
35. Leplumey I, Camillerapp J, Lorette G (1993) A robust detector for music staves. In: Proceedings of the international conference on document analysis and recognition, Tsukuba Science city, Japan, pp 902–905
36. Luth N (2002) Automatic identification of music notations. In: Proceedings of the second international conference on WEB delivering of music (WEDELMUSIC), Darmstadt, Germany, pp 203–210
37. Macé S, Anquetil É, Coüasnon B (2005) A generic method to design pen-based systems for structured document composition: development of a musical score editor. In: Proceedings of the 1st workshop on improving and assessing pen-based input techniques, Edinburgh, pp 15–22
38. Mahoney J (1982) Automatic analysis of musical score images. B.sc. thesis, Massachussets Institute if technology, Dept of Engineering and Computer Science
39. Matsushima T, Ohteru S, Hashimoto S (1989) An integrated music information processing system: Psb-er. In: Proceedings of the international computer music conference, Columbus pp 191–198
40. Miyao H, Maruyama M (2007) An online handwritten music symbol recognition system. *Int J Doc Anal Recognit* 9(1):49–58
41. Miyao H, Nakano Y (1995) Head and stem extraction from printed music scores using a neural network approach. In: Proceedings of the 3rd international conference on document analysis and recognition, Montreal, Canada, pp 1074–1079
42. Modayur BR, Ramesh V, Haralick RM, Shapiro LG (1993) Muser: a prototype musical score recognition system using mathematical morphology. *Mach Vis Appl* 6(2–3):140–150
43. Ng K (2002) Music manuscript tracing. Blostein D, Kwon Y-B (eds) *Graphics Recognition Algorithms and Applications*, Volume 2390 of lecture notes in computer science, Springer, Berlin Heidelberg, pp 330–342.
44. Pinto J, Vieira P, Sosa J (2003) A new graph-like classification method applied to ancient handwritten musical symbols. *Int J Doc Anal Recognit (IJDAR)* 6(1):10–22
45. Prerau D (1970) Computer pattern recognition of standard engraved music notation. PhD thesis, Massachussets Institute if technology, Dept of Engineering and Computer Science
46. Pruslin D (1966) Automatic recognition of sheet music. PhD thesis, Massachussets Institute if technology
47. Pugin L (2006) Optical music recognition of early typographic prints using hidden markov models. In: International conference on music information retrieval, Victoria, Canada, pp 53–56
48. Pugin L, Burgoyne JA, Fujinaga I (2007) Goal-directed evaluation for the improvement of optical music recognition on early music prints. In: Rasmussen EM, Larson RR, Toms E, Sugimoto S (eds) *Proceedings of the 7th ACM/IEEE-CS joint conference on digital libraries*, Vancouver, Canada. ACM, pp 303–304
49. Pugin L, Hockman J, Burgoyne JA, Fujinaga I (2008) GAMERA versus ARUSPIX. Two optical music recognition approaches. In: *Proceedings of the 9th international conference on music information retrieval*, Philadelphia, USA, pp 419–424
50. Randriamahefa R, Cocquerez J, Fluhr C, Pépin F, Philipp S (1993) Printed music recognition. In: Proceedings of the international conference on document analysis and recognition, ICDAR, Tsukuba Science city, Japan, pp 898–901

51. Rebelo A, Capela G, Cardoso J (2010) Optical recognition of music symbols. *Int J Doc Anal Recognit* 13(1):19–31
52. Rebelo A, Fujinaga I, Paszkiewicz F, Marcal A, Guedes C, Cardoso J (2012) Optical music recognition: state-of-the-art and open issues. *Int J Multimed Inf Retr* 1(3):173–190
53. Roach J, Tatem J (1988) Using domain knowledge in low-level visual processing to interpret handwritten music: an experiment. *Pattern Recognit* 21(1):33–44
54. Rossant F, Bloch I (2005) Optical music recognition based on a fuzzy modeling of symbol classes and music writing rules. In: IEEE international conference on image processing, Genova, Italy, vol 2, pp 538–541
55. Stückelberg MV, Doermann DS (1999) On musical score recognition using probabilistic reasoning. In: Proceedings of the international conference on document analysis and recognition, ICDAR, Bangalore, pp 115–118
56. Su B, Lu S, Pal U, Tan C (2012) An effective staff detection and removal technique for musical documents. In: IAPR international workshop on document analysis systems (DAS), Gold Coast, Queensland, Australia. IEEE Computer Society, pp 160–164
57. Toyama F, Shoji K, Miyamichi J (2006) Symbol recognition of printed piano scores with touching symbols. In: International conference on pattern recognition, Hong Kong, vol 2. IEEE Computer Society, pp 480–483

## Further Reading

For further reading on Optical Music Recognition, the reader is referred to the critical survey by Blostein and Baird [5], as well as the book by Susan George [28]. Recently, Rebelo et al. have published [52] a survey which also includes a description about the available databases and software. Finally, and concerning the recognition of music symbols, Rebelo et al. have also compared the main techniques in [51].

In addition to the recognition of music scores, some researchers have been focused on writer identification. In this case, the goal is to identify the writer of a music score, which is especially useful for scholars when determining the authorship of anonymous historical documents. For further information, the reader is referred to [6, 22, 25, 29, 36].

Jianying Hu and Ying Liu

## Contents

Introduction.....	776
Challenges and Recent Advances.....	777
PDF Document Analysis.....	777
Motivation.....	777
Challenges.....	777
Overview of PDF Models and Objects.....	778
PDF Document Understanding Approaches.....	779
Specific Object Recognition in PDF .....	784
Web Document Analysis.....	786
Web Page Segmentation.....	786
Web Content Extraction.....	790
Analysis of Text in Web Images.....	794
Web Document Modeling.....	798
Available Software.....	799
PDF Analysis Tools.....	799
Web Document Analysis Tools.....	800
Conclusion.....	801
Cross-References.....	801
References.....	801
Further Reading.....	804

---

J. Hu (✉)

IBM T.J. Watson Research Center, Yorktown Heights, NY, USA  
e-mail: [jyhu@us.ibm.com](mailto:jyhu@us.ibm.com)

Y. Liu

Korea Advanced Institute of Science and Technology (KAIST), Yuseong-gu, Daejeon,  
Republic of Korea  
e-mail: [yingliu@kaist.ac.kr](mailto:yingliu@kaist.ac.kr)

**Abstract**

While traditional document analysis has focused on printed media, an increasingly large portion of the documents today are generated in digital form from the start. Such “documents born digital” range from plain text documents such as emails to more sophisticated forms such as PDF documents and Web documents. On the one hand, the existence of the digital encoding of documents eliminates the need for scanning, image processing, and character recognition in most situations (a notable exception being the prevalent use of text embedded in images for Web documents, as elaborated upon in section “Analysis of Text in Web Images”). On the other hand, many higher-level processing tasks remain due to the fact that the design purpose of almost existing digital document encoding systems (i.e., HTML, PDF) is for display or printing for human consumption, not for machine-level information exchange and extraction. As such, significant amount of processing is still required for automatic information extraction, indexing, and content repurposing from such documents, and many challenges exist in this process. This chapter describes in detail the key technologies for processing documents born digital, with a focus on PDF and Web document processing.

**Keywords**

Content extraction • Digital libraries • Digital publishing • Document layout analysis • Document object model • Document understanding • Page segmentation • Portal document • Web documents

---

**Introduction**

The work on analysis of document born digital started in the mid- to late 1990s, on PDF documents, plain text documents such as email, and analysis of Web documents as semi-structured data [33]. The earliest book of collections of work on Web document analysis appeared in 2004 [4]. Because of the degree of complexity involved in PDF and Web documents and the phenomenal growth in prevalence of these two encoding systems as popular digital publishing media, significant amount of work has been carried out in these two areas and tremendous progress has been made in the past 15 years. This chapter thus focuses on these two dominant areas: PDF Document Process and Web Document Process. For each area, detailed explanations are given on the motivation and research challenges, the key approaches, and state of the art for addressing the challenges, as well as promising new directions. Also provided are descriptions of available software and tools and a comprehensive list of references for further reading.

## Challenges and Recent Advances

### PDF Document Analysis

Along with the rapid expansion of digital libraries, PDF documents have become an important data source for many applications, such as information retrieval or data integration. The demand for extracting and analyzing information from PDF documents is increasing. In order to unlock the information in PDF documents, one needs to not only understand the PDF file format and syntax at object, file as well as document level, but also become familiar with representative PDF document analysis tools and methodologies.

### Motivation

After years of massive digitization, multiple applications and file formats are launched as “digital paper” or “portable documents.” The representative examples are Novell’s *Envoy*, Common Ground’s *Common Ground*, and Adobe’s *Acrobat*. All of them attempt to provide a platform- and application-independent way of distributing electronically produced final form documents. *Acrobat* and its underlying Portable Document Format (PDF) are winning the race to becoming a de facto standard. Version 1.0 of PDF was published in 1993, and PDF has been gradually established as the most popular format of digital document distribution. As a result, the analysis and understanding of PDF documents has become a crucial and vital step for all the further applications such as information extraction.

### Challenges

Although PDF has become the defacto standard for the distribution of electronic documents, especially in digital libraries and the World Wide Web, the print-oriented nature of PDF leads to serious drawbacks as follows:

- Little structural information

Since PDF was originally designed for the final presentation of a document, most PDF documents are untagged. They contain very little or no explicit structural information about the contents such as words, text lines, paragraphs, logos, and figure illustrations, which makes content extracting or repurposing a challenging task. Particularly, it is difficult to reuse, edit, or modify the layout or the content of the document.

- Render ordering problem

The main advantage of PDF documents is to preserve the original document look. As a document reviewer, one cares about the final PDF layout and content, instead of the process when the page is produced. Many different PDF programs (e.g., PostScript) can generate the same document appearances and rendered

effect on the screen; however, they may follow different render ordering. A given PDF file might be structured to write out text a word/character at a time, or render a specific type of texts (e.g., titles) first or some other algorithm that tries to optimize on-screen rendering. Very often, the render ordering is totally different from the “reading order.” For multicolumn text, the file is sometimes rendered across the columns by hopping across the inter-column gutter. Although the lack of standard render sequence does not affect the PDF document displaying and reading, it heavily impacts the performance of document structure analysis and understanding.

- Object overlapping problem

If a PDF document has rich graphical contents, there is a high probability to have block overlay phenomena, e.g., a textual company name and a company logo are embedded in an image, which makes the analysis more complicated. Moreover, even if it is possible to identify all the page objects in a PDF document, they are far from reflecting the logical structure or logical components of document. For example, a text object may only have part of the characters of a word. Path objects are often just a fraction of the whole figure, e.g., one line in a line chart. Such overlapped objects introduce additional burdens and chances in PDF understanding and logical structure analysis.

- Diverse document layouts and inconsistent fonts

Because PDF documents are generated by different approaches (see details in section “[Analysis of Text in Web Images](#)”) and widely used in many domains from scientific papers to scanned transcripts and from advertising flyers to statistical reports. The diverse document layouts and complicated typesetting make PDF documents prone to errors, particularly in multicolumn documents with inconsistent text font types or when the original document has been created at 90° rotation or inter-column gutter is very narrow.

- Object extraction

Although extracting and analyzing the whole PDF file is a popular research topic, sometimes one needs to be able to extract parts of the whole file and use it in Web pages, word processing documents, PowerPoint presentations, or desktop publishing software. Such document segments include not only paragraphs or sentences in specific locations but also typical logical components such as tables or images. Fortunately there are many systems/tools that can satisfy this requirement. For example, the full version of Adobe Acrobat can extract individual images or all images from a PDF and export in various formats such as EPS, JPG, and TIFF. In addition, many third-party PDF extraction Software Tools exist. However, these tools often cannot fulfil the task of object extraction fully automatically. Instead, human involvement such as manually selecting the object boundary in document is needed.

## Overview of PDF Models and Objects

### Outside PDF

Huge number of documents are either created or converted into PDF format from other document formats. They can be generated from nearly all standard

**Table 23.1** The representative PDF generation methods

PDF generation methods	Details
Method 1	Create PDF directly by PDFWriter from document authoring systems
Method 2	Create PDF from PostScript files by Adobe Distiller software
Method 3	Create PDF by Adobe Acrobat Capture based on a bitmap image document in TIFF form

applications (Microsoft Word, Microsoft Excel, Microsoft PowerPoint, Adobe Photoshop, Adobe InDesign, Adobe Illustrator, Quark XPress, etc.) by a pseudo printer driver called *PDFWriter* or directly from PostScript using Adobe's *Distiller*, which is an interpreter to convert PostScript files to Adobe PDF. The generation methods of PDF documents can be classified into the following types shown in Table 23.1.

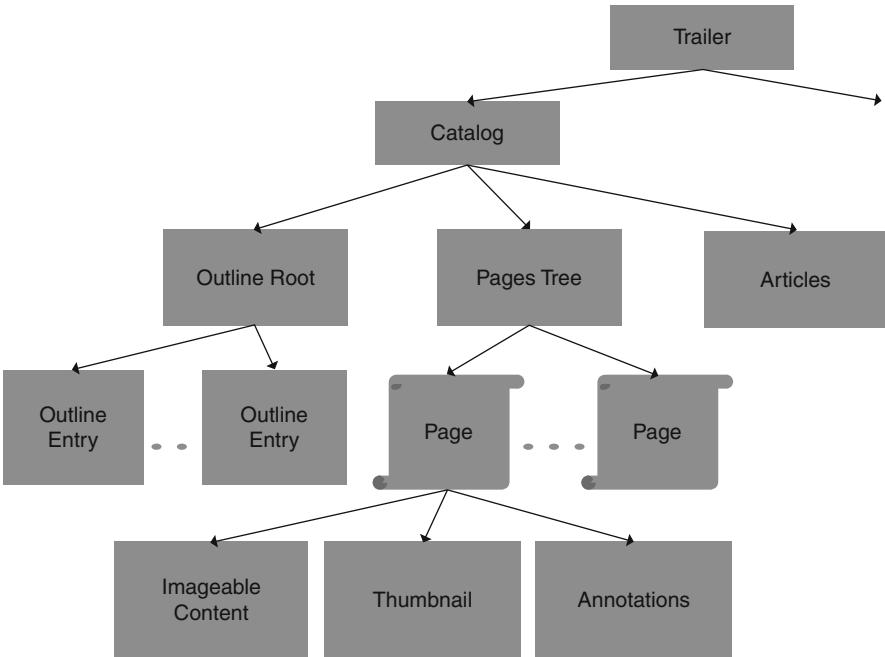
### Inside PDF

PDF files consist of a number of distinct *indirect objects* which may be distributed randomly within the file. These objects include text objects, image objects, and path objects. Every PDF document has a root object (catalog) containing indirect references to various structures. For faster access in viewer software, PDF pages are arranged in a tree structure, instead of linearly. The page tree is a balanced tree of page objects, with page objects at the leaves, and allows faster access to documents containing multiple pages. Please see PDF file structure in Fig. 23.1. Please note that page object does not reflect to the logical structure in a PDF document. Similarly, a text object may only contain part of a word (e.g., several characters) and the path object contains part of graphical illustrations (e.g., a line in a figure). Based on the nature of PDF document content stream and the frequently happened block overlay among different objects, how to analyze and understand the logical component of a PDF document is a challenging problem.

As shown in Fig. 23.1, images are represented as separate child objects in the pages tree model. This makes them easy to identify and no further processing needs to be done to segment or classify these elements. The thumbnail entry in a page object is a tiny JPEG-compressed image of the page, which can be used for navigation in a viewer. The Annotations contains several standard “hyper-features” features, such as link bookmarks and “yellow sticks.” New features can be added to Acrobat software by third-party plug-ins written using the Acrobat SDK.

### PDF Document Understanding Approaches

Because the analysis of document images has a longer history than that of PDF documents, usually there are two ways to analyze PDF documents: either directly analyze the PDF documents or convert the PDF documents into bitmap images and perform existing document image segmentation techniques.



**Fig. 23.1** Part of the main PDF file structure

### Image-Based Approaches

Since the middle of last century, image documents are the main pre-Web documents, which are dominant in the document understanding and analysis field. Image documents are scanned from pseudo binary hardcopy paper manuscripts with a flatbed sheet-fed or mounted imaging devices. Extensive works have been done in detecting logical structure in scanned bitmap image documents for document understanding and information extraction [37]. Although most methods are not originally designed for PDF document analysis, many of them can also be reapplied to PDF documents. They simply make use of a bitmap rendition of each PDF page and apply methods similar to those designed for scanned pages. The typical preprocessing step is to render a PDF document into a bitmap. Such bitmap images will be accepted as inputs, and then text, graphics, and images are identified and extracted. Because of the limited information available in the bitmap images, most studies are limited to business letters, technical journals, and newspapers, where object overlaps are minimal.

In order to identify the textual information as well as understand the document semantically, Optical Character Recognition (OCR) [30] and many other techniques are necessary. This processing also requires the recognition of fonts, as well as words and characters. The resulting PDF is usually much more compact than the TIFF image and can be searched for arbitrary phrases and keywords by all of the standard PDF search engines.

**Table 23.2** Representative literatures on PDF document analysis

Literatures	Methods/tools	Contributions
Smith and Brailsford [40]	Segment a PDF document page into different image and text blocks	Convert a PDF document into a pool of logical, typed blocks
Anjewierde [3]	AIDAS: converted a PDF to a set of text, graphics, and image objects	Incremental Logical Structure Discovery in PDF documents
Futrelle et al. [15]	Obtain the graphics primitives in object form	Recognizing graphics from PDF
Hadjar et al. [18]	Xed: applying TIFF image layout analysis methods on PDF documents	Extracting hidden structures from PDF documents using text and graphic objects
Chao and Fan [8]	Separated each PDF page into three layers: text, image, and vector graphics layer	Identifying logical components in a PDF page
Marinai [35]	Greenstone package	Extract basic metadata from PDF documents
Hassan [19]	PDF parse method based on <i>PDFBox</i>	Object-level document analysis of PDF files

### Direct-Processing Approaches

PDF document content stream lists all the page objects, such as text, image, and path. Thus another way to discover the logical components of a document is to analyze and interpret the layouts and attributes of the page objects so as to correctly break or group them into different logical components.

With the help of many commercial or open-source tools such as XPDF (<http://www.foolabs.com/xpdf>) or PDFBOX (<http://pdfbox.apache.org/>), the page objects can be extracted from PDF sources directly. The major work is to segment the texts in a PDF page into regions containing texts with similar attributes. The two most important attributes are point size and font name, which are stored as part of the page resources in each node of the PDF page tree in Fig. 23.1. These valuable attributes are used as key features with predefined heuristic-based methods for tagging the various segments.

### The State of the Art

Based on the fundamental approaches introduced above, several representative works are selected and listed chronologically in Table 23.2, which not only provides summaries and comparisons of these methods but also reflects the research progress and trend in this area.

Similar to other document media, there are three approaches to analyze the PDF document structure: model-driven (top-down), data-driven (bottom-up), or hybrid approach. Hybrid algorithms can be regarded as a mix of the above two approaches.

- Model-driven approach is also called top-down approach. If PDF document structure and the page model are known in advance, the top-down approach can be used based on the predefined page templates. To apply this method, one needs to define the PDF document model or understand the page template in advance. Top-down approach starts from the whole PDF document page and iteratively split it into smaller geometric elements based on paragraph breaks or intercolumn gutters. The splitting procedure stops when some criterion is met and the ranges obtained at that stage constitute the final segmentation results. Many existing top-down methods in image document analysis field can be reused.
- If the layout information is unknown in advance, a bottom-up method is adopted. Bottom-up algorithms usually start from the lowest-level document objects (e.g., the characters or image pixels) and cluster them into larger connected components such as words, which are then be grouped into lines or zones. Plain text document analysis methods, such as column/layout detection or space analysis, can be reused here.

Although researchers keep proposing new methods/systems to improve the performance of PDF document understanding, most of them obtain the low-level document objects by taking advantage of text extraction tools, such as open-source libraries for processing PDFs: XPDF library, the Apache PDFBox™ library (<http://pdfbox.apache.org>), PDFlib Text Extraction Toolkit (TET <http://www.pdflib.com/products/tet>), PDF2Text ([www.pdf2text.com](http://www.pdf2text.com)), etc. Then diverse techniques are adopted to have a better understanding on the document content and structure. Most literatures in PDF understanding field adopt bottom-up methods that share the following main steps:

- Step 1: Extracting diverse document objects and low-level data in PDF

Usually different PDF extraction tools are used to extract the objects and low-level data. JPedal is a Java package that can extract texts, images, annotations, and forms from PDF. The Apache PDFBox™ library is an open-source Java tool to extract texts from PDF. Smith and Brailsford [40] seem published the first paper that deals with the analysis of PDF files. They introduced a new definition of an Encapsulated PDF (*EPDF*), which segments a PDF document page into different image and text blocks. The Adobe Acrobat SDM is used to obtain textual objects.

In order to minimize the potential logical components overlay and the interference between different logical components in PDF documents, researchers separated a PDF page into three layers: text, image, and vector graphics layer. Anjewierde [3] extracted the logical document structure by converting a PDF document into a set of text, graphics, and image objects using XPDF library. Ten different classes of layout objects (text, line, rectangle, images, curve, etc.) are extracted and each class has ten features (position, font size, color, line width, etc.).

Chao and Fan [8] saved each layer as an individual PDF document. Based on the type of each layer, they decided whether to analyze the PDF document directly or convert it into bitmap images then perform image analysis methods. Text and image objects are obtained directly from the PDF sources. They extracted the

font, size, line spacing, and color as the text style as well as the text strings as the content. Shapes of the images and the masks are extracted for image components. Lines and vector objects are obtained from a bitmap image.

Hadjar et al. [18] also proposed a new tool called Xed for extracting hidden structures from PDF documents using text and graphic objects. Xed extracts all the objects from a PDF document including text, images, and graphics. They merge the typical low-level extraction methods applied on PDF files with the traditional layout analysis performed on TIFF images.

Futrelle et al. [15] adopted the *Etymon PJ Tools* (<http://www.etymon.com/epub.html>) library to obtain the graphics primitives in object form from PDF documents. The extracted information is at a much finer granular level than what is required for document analysis.

Hassan [19] implemented his PDF parse method based on *PDFBox*, an open-source library for processing PDFs. They used *PDFStreamEngine* class to extract and store the objects from a page. Besides, this paper explains how the relevant PDF instructions can be processed to form a set of objects suitable for document analysis.

Marinai [35] designed the Greenstone package, a full integration with multiple popular digital library software, to extract basic metadata from PDF documents. The *metadataPDFPlug* is a plug-in, which imports PDF documents based on PDF reader XPDF and the Ghostscript libraries.

- Step 2: Forming words from characters, lines from words, and layout structures from lines by discriminating line spacing, indentation, and basic geometric relationships

Almost all the related works started with the low-level objects extracted in Step 1 and tried to convert the textual pieces into larger textual elements in a bottom-up way: from characters to words, lines, paragraphs, and even page columns. Usually the word-finding algorithm of the API within the Acrobat Exchange viewer can identify the logical words. Based on the geometric information of each word, line, or paragraph, even larger textual regions can be generated by combining the font name and font size and comparing the start, middle, and end point among adjacent textual elements. The detailed steps can be found in [40]. Besides font name and point size features, they also considered a set of metrics holding the important geometric features of the font: x-height, caps height, serif, sans serif, stem width, and so on. Similarly, Chao and Fan [8] used an attribute vector, which included font name, font size, color, and orientation as well as the four corners of the quad(s) provided by Adobe's Acrobat word finder to represent the style of the words. Grouping adjacent textual pieces step is same as other papers. Marinai [35] adopted *PdfTohtml* to merge text lines into paragraphs by preserving the texts in reading order. Coordinate information is analyzed for each textual piece and merges together textual blocks in horizontal and vertical directions. Hassan [19] proposed a best-first clustering algorithm, a robust bottom-up segmentation algorithm based on visual principles even on complex layouts such as newsprint. There is no doubt that the performance of layout structure analysis is improved, but complex layout structure still needs human intervention.

- Step 3: Identifying document logical components

Usually the logical components are identified based on the structure components extracted in previous steps. Various approaches were proposed and implemented. Top-down combination method is used when the document-style information (e.g., font parameters, size of section titles, line spaces) is known. Another popular method is discovering a hierarchy of layout objects then mapping the layout objects to the logical components. Anjewierde [3] discovered the logical structures based on further grammar processing by processing text, image, and vector graphics layer separately to avoid the effect of the object overlapping. Smith and Brailsford [40] used PDF to represent block-based documents, by carrying each block with the resource requirements as well as geometrical and logical information. With his extension proposal, the authors converted a PDF document into a pool of logical typed blocks. A block formatter called Juggler allows these blocks to be extracted and reused. In order to automatically extract administrative metadata, the authors in [35] developed a novel package *pdf2gsdl* to extract PDF document metadata. They used a Multi-Layer Perceptron (MLP) classifier and the neural classifier to identify regions and label metadata to each of them.

## Specific Object Recognition in PDF

Instead of extracting all the contents in a PDF document, one may be interested in a part of the contents or specific document objects. The lower-level objects include symbols, words, sentences, and paragraphs, which can be easily parsed and identified. The higher-level objects refer to textual granularities that need further analysis. Table is one of representative examples.

In the last 20 years, over 200 papers have been published on table recognition field. Zanibb et al. [47] provides a survey with detailed description of existing approaches and systems. In addition, a detailed bibliography on table segmentation and extraction can be found in <http://www.visionbib.com/bibliography/char969.htm>. To extract data from tables, two separate tasks need to be fulfilled:

- Step 1: Detecting the table boundary from the document
- Step 2: Extracting the table cells as well as other table-related information

Researchers in the automatic table extraction field largely focus on analyzing the table in a specific document media. Most works focus on three main document types: image-oriented, HTML-oriented, and the plain text-oriented. Although many approaches and systems are available for these document types, few methods are designed to deal with PDF tables.

Different media requires different table boundary detection and table decomposition methods. Although they are not new research topics, very few published papers provided detailed algorithms to extract tables from PDF documents. Most PDF table extraction works have similar preprocessing steps:

- Sorting texts by the coordinates to avoid the order uncertainty brought by the document creator (detect the table boundary first and then only sort the small areas)

- Merging text elements into lines, combining lines into blocks
- Analyzing each line/block to decide whether it is belonging to a table boundary

There are several representative works on table detection in PDF documents. Wasserman et al. [44] designed an algorithm to detect potential table regions in a PDF document and to analyze the table structure. First, they converted 100 scientific documents in PDF format into TIFF image format, then treated a set of word boxes as the input of the algorithm, and used the word segmentation technique in [43]. They yielded a set of potential table-region bounding boxes through the processing of a set of modules: word-box consolidation and text line elimination module, vertical-range module, horizontal-range module, and table-region extraction module. The main shortcoming of this paper is the lack of the details. Although they realized the importance of the precise definition of table, there is no definition for all the terms. In addition, there is no experiment and evaluation.

pdf2table [46] is an early-stage tool to extract tables from PDF files. Instead of the typical textual information extraction tools such as PDF2TET or TET, pdf2table uses the pdf2html (<http://pdftohtml.sourceforge.net>) developed by Gueorgui Ovtcharov and Rainer Dorsch to get all text elements (i.e., strings) with the absolute coordinates in the original file. All the returned texts are saved in a structured data format (XML file). Heuristic-based approach is also used to detect and decompose table boundary. However, the performance is far from unsatisfying. To remedy the deficiency of the tool, the authors implemented a graphical user interface, which enables the user to make adjustments on the extracted data.

Tamir Hassan and Robert Baumgratner [20] also used PDFBox to extract texts from PDF files and try to recognize tables. They grouped tables into three categories based on the existing of horizontal and vertical ruling lines. However, ruling lines are not required in real situations. AIDAS system, which extracted the logical document structure from PDF documents [3], used shallow grammars to detect the logical elements as the authors admitted, and no result and evaluation is presented.

Liu et al. designed TableSeer, a table search engine based on extracting table metadata from PDF documents. The early version of TableSeer [31] decided the table boundary within a PDF page by classifying a document page into a set of boxes using a top-down method. Then each box is analyzed and classified into table candidate box or not. The later version of TableSeer adopted a bottom-up method by labeling each document line with the table candidate boxes as table line or non-table line. Both heuristic methods and machine learning-based methods are used in this work.

PDF-TREX [38] recognized each PDF table as a 2-dimensional grid on a Cartesian plane by analyzing a set of basic content elements heuristically in a bottom-up way.

PDF table extraction is a challenging problem because of many reasons. First, double-column even three- or four-column PDF documents introduce more difficulty in data extraction. In addition, many methods are only good for the lucid tables. Second, many heuristics cannot be easily extended and reapplied to other datasets. Each of them will incur a lot of noisy results. Third, widely used thresholds are very important to the final performance and it is impossible to deal with all the cases with a single fixed threshold value.

## Web Document Analysis

Over the last 20 years, the Web has rapidly become a dominant media for information exchange, where users ranging from professional publishing institutions such as New York Times to individual bloggers routinely publish and access documents on demand. The availability and extent of this vast information source coupled with the decentralized nature of the medium pose both increasing demand and significant challenges for automated content analysis. Such content analysis is crucial for applications such as content repurposing, information extraction, Web mining, and Web security. This section presents an overview of the four key challenges in this area: Web Page Segmentation, Web Content Extraction, Analysis of Text in Web Images, and Web Document Modeling.

## Web Page Segmentation

The Web as a publishing medium has become significantly more mature and more sophisticated over the last decade, while at the same time more decentralized. As a result, Web page layout has become complex and diverse. The segmentation of a Web page into visually and semantically coherent components has emerged as one of the most important Web document analysis tasks supporting many important applications and serving as the preprocessing step for comprehensive Web content extraction.

### Motivation

Web page segmentation is the crucial first step in identifying different types of information (i.e., heading, navigation bar, central content), and the informative and non-informative portions of the Web page. Such information is very useful in Web ranking and Web data mining applications, as well as Web mining tasks such as Web document clustering and duplicate detection.

The identification of segments also plays an important role in displaying Web pages on-screen-space-constrained devices such as smartphones and PDAs. While these mobile devices are being increasingly used to access the Web, the small size of the device screens often limits the user experience, as the vast majority of the Web sites are designed for viewing on a standard desktop or laptop screen. Proper page segmentation is vital in providing the ability to automatically adapting such sites for effective viewing on small screens.

### Approaches

Since most Web pages can be parsed into a DOM (Document Object Model) using a standard HTML parser, and by definition the DOM tree contains all element needed to render a Web page, it is a natural place to start with any Web page segmentation task. Given the DOM tree, the problem of Web segmentation becomes

one of the grouping DOM elements into coherent regions. A variety of methods have been developed to address this problem. These methods can be roughly categorized into the following three categories based on the specific grouping mechanism.

### DOM Tree Centric Approach

A large number of Web pages follow a common template at the highest level, which includes header, footer, left side bar, right side bar, and body. These high-level content blocks typically follow certain layout conventions which can be used to derive rules for their classification using information coded in the DOM tree. Since each high-level content block (especially the body block) can still be heterogeneous, more rules based on HTML tags (explicit separators) and layout characteristics (implicit separators) need to be learned for potential further segmentation. The approach involves the following two steps [10]:

- Step 1. High-level content block detection

The DOM tree, which contains the position and dimension information for each node, is traversed to classify each node as belonging to the one of the five high-level blocks. The classification is based on shape and location conventions typically followed by each of these blocks. For example, the header tends to be located at the top portion of the page and also tends to have a “flat” shape, i.e., with low height to width ratio. Instead of devising handcraft rules on spatial location and aspect ratios, a more systematic approach is to extract these spatial features (top, bottom, left, right, and aspect ratios) for each element from the HTML DOM tree and then train a classifier such as nonlinear support vector machine using a set of labeled results.

- Step 2. Further segmentation of content body

To further identify finer grouping within each content body, first a set of explicit separators are identified. Explicit separators are HTML elements that can be determined to be a separator based on its own properties. They include elements with tags, such as `<p>`, `<hr>`, `<table>`, `<td>`, `<ul>`, `<h1>`, and `<div>`, and thin images (which can be readily detected by examining the aspect ratio and can be easily identify with one tree traversal).

While these explicit separators are easy to extract from the DOM tree, they don't necessarily cover all potential separators, since certain separation is visual and not directly coded in the DOM tree. In order to identify these separators, one needs to first project the basic content blocks along the horizontal and vertical axes to generate the project diagrams, and then identify gaps along on 2 each axis as implicit separators. This is very similar to the projection method used in scanned document image segmentation ([► Chap. 5 \(Page Segmentation Techniques in Document Analysis\)](#)).

A natural extension to the projection based approach for page segmentation is the X-Y cut algorithm that has found great success in scanned document image segmentation. Like in the case of scanned documents, repeated cuts can be made along the X and Y direction based on a coherence measure, until desired number of blocks or a predefined coherence threshold is reached.

### Vision-Based Content Structure Approach

The projection-based method and X-Y cut-based algorithms are attempts to combine DOM structure with visual cues for page segmentation. Another approach that takes this one step further and puts more emphasis on visual representation is the algorithm called VIPS (Vision-Based Page Segmentation) [5].

In the VIPS algorithm, instead of operating solely on the DOM tree, a vision-based content structure of a page is deduced by combining the DOM structure and the visual cues. From the root node of the DOM tree, a visual block extraction process is applied to extract visual blocks at each level based on rules designed to capture four visual cues:

- Tag cue: A node is divided if it contains one of the explicit separators as defined in the previous section.
- Color cue: A node is divided if its background color is different from one of its child.
- Text cue: A node is not divided if most of the children of a DOM node are Text nodes.
- A DOM node is divided if the standard deviation of size of its children is larger than a threshold.

All extracted blocks are then placed into a pool for visual separator detection. Visual separators are defined as horizontal or vertical gaps that do not intersect with any block. The algorithm for visual separator detection proceeds in an iterative manner by traversing through all the blocks in the pool. Each separator is then given weights based on the following set of heuristics:

- The larger the distance between blocks on different side of the separator, the higher the weight.
- The weight of a separator is increased if:
  - A visual separator is at the same position as an explicit separator such as <HR>.
  - The differences of font properties such as font size and font weight are more pronounced on two sides of the separator.
  - Background colors are different on two sides of the separator.
- When the structures of the blocks on opposite sides of the separator are similar (e.g., both are text), the weight of the separator is decreased.

Finally, the visual structure construction process starts from the separators with the lowest weight, and the blocks opposite these separators are merged to form new virtual blocks. This process iterates until a predefined level of granularity or degree of coherence is reached.

### Graph Theoretic Approach

The methods described in the previous two sections rely heavily on heuristics applied to tree-based greedy search, which tend to lead to local minima that are not necessarily optimal. To address this problem, Chakrabarti et al. [7] developed an approach which is based on a weighted graph constructed over nodes of the DOM tree, where the cost of assigning pairs of DOM elements to different of identical segments can be explicitly coded. This construct then allows the definition of a

global cost of a segmentation, which can be minimized in a disciplined manner using well-established graph clustering algorithms.

The approach is based on the following formulation. Given a DOM tree, let  $N$  be the set of nodes. A graph is constructed whose node set is  $N$  and whose edges have a weight that represents a cost of placing the adjacent nodes in different segments.  $D_p : L \rightarrow R^{\geq 0}$  represent the cost of assigning a specific label to the node  $p$ . Let  $V_{pq} : L \times L \rightarrow R^{\geq 0}$  represent the cost of assigning two different labels to adjacent nodes of edge  $(p, q)$ . Let  $S : N \rightarrow L$  be the segmentation function that assigns to each node  $p$  the segment label  $S(p)$ . The total cost of a segmentation  $S$  can be represented as

$$\sum_{p \in N} D_p(S(p)) + \sum_{p, q \in N} V_{pq}(S(p), S(q)). \quad (23.1)$$

While conceptually appealing, this objective function is fairly general and doesn't fully capture the specific requirements of the segmentation problem. Particularly, in the segmentation problem the rendering constraint applies, e.g., the area on the screen occupied by a child node is geometrically contained inside that of its parent node in the DOM tree. Therefore, any segmentation should follow the rule that if it places the root of a subtree in a particular segment, then it has to place all the nodes in the entire subtree in the same segment.

One relatively straightforward approach to incorporating the rendering constraint in the above objective function is based on the observation that the actual labels of segments do not really matter in segmentation – one mostly cares about whether adjacent nodes have the same label. Thus the labels can be treated as indistinguishable, and the objective function (23.1) can be changed to

$$\sum_{S(p) \neq S(q)} V_{pq} + \sum_{S(p) = S(q)} (1 - V_{pq}) \quad (23.2)$$

Using objective function (23.2), the rendering constraint can be enforced by considering only the leaf nodes of the DOM tree in constructing the weighted graph, and perform segmentation on these nodes only.

Equation (23.2) encodes the so-called correlation clustering problem, which has known practical combinatorial approximation algorithms, including a recent one called CClus by Ailon, Charikar, and Newman [2] that approximates Eq. (23.2) to within a factor 2.

Two main types of features similar to the ones used in the algorithms described in the previous sections are defined to compute the edge weights  $V_{pq}$ : visual and content based. The visual features include the node's position, shape (aspect ratio), background color, types, sizes, and colors of text. The content-based features are designed to capture the purpose of the content and include features such as average size of sentences, fraction of text within anchors, and tag names. The edge weights are derived from these features using a logistic regression function learned from a set of manually labeled Web pages, where each DOM node is assigned a segment ID. The logistic function takes the feature of two adjacent nodes as input and output

the probability of the two nodes belong to the same segment, which is used as the  $V_{pq}$  score.

While the correlation clustering formulation is intuitive and easy to implement, it has the drawback that it relies solely on the leaf nodes of the DOM tree. By design, the leaf nodes tend to carry limited information, which makes the features sparse and noisy, and thus negatively impact the segmentation quality. To address this problem, Chakrabarti et al. further proposed a different formulation based on energy-minimizing cuts, which is able to take into consideration all DOM nodes while preserving the rendering constraint [7]. While this approach is more general than the correlation clustering approach, it is also much more complex both in terms of graph construction and parameter training, and will not be covered in this chapter. Interested readers are referred to the original publication for more details on this algorithm.

## Web Content Extraction

The information amount in Web steadily grows to be the largest public knowledge repository. As the accumulation of Web pages, extracting Web data into database and spreadsheet instantly and accurately is crucial to diverse applications, such as product catalogue extraction or online price competition. Data in Web documents is usually extracted by wrappers, which are specialized program routines that can make the content of HTML pages directly available. Wrappers extract web contents in three steps: download HTML pages from a Web site, recognize and extract specified data, and save this data in a suitably structured format to enable further manipulation.

### Motivation and Challenges

Web content is the reason that people visit and browse the Web pages. Web pages contain not only the body contents, such as textual information or visual contents, but also many other distracting features including pop-up advertisements, scattered decoration images, or unnecessary hyperlinks. Automatic extraction of useful and relevant contents from Web pages has many applications, ranging from defining “accordion summarization” to Web page classification and labeling, from enabling end users to view Web more freely over small screen devices such as personal digital assistants (PDAs) or cellular phones (e.g., ThunderHawk: <http://www.bitstream.com/wireless/server/workflow.html>), to provide better access to the Web for visually impaired and blinds. Typical tools and systems include Opera (<http://www.openxml.org>), Hal Screen Reader by Dolphin Computer Access (<http://www.dolphinuk.co.uk/products/hal.htm>), and Microsoft’s Narrator ([http://www.microsoft.com/technet/treeview/default.asp?url=/technet/prodtechnol/winxppro/reader\\_overview.asp](http://www.microsoft.com/technet/treeview/default.asp?url=/technet/prodtechnol/winxppro/reader_overview.asp)).

Within the most straightforward Web document, HTML files, web content is displaying in diverse formats: textual, images, sounds, videos, or animations. Essentially, web contents can be classified into two types:

- **Textual contents**

Textual contents include all the written contents on the page, both in text sections and inside figures. The best textual Web content is the text written for the Web directly, rather than simply copy and pasted from another print source.

- **Multimedia contents**

The other type of Web content is multimedia content, which refers to any content that isn't the text. Multimedia contents include several different types of Web contents: animation, images, sound, video, etc.

In order to analyze a Web page for content extraction, the typical method is writing specialized programs called wrappers/parsers to extract target information and contents and mapping them to new formats that support better data exchange and integration. Usually the Web content extraction contains the following steps:

- Retrieving and analyzing a Web page structure for content extraction
- Passing the page to a HTML parser
- Correcting HTML errors if necessary and converting web contents into XML or DOM tree representations for better understanding and organization
- Segmenting Web pages into different zones or categories, extracting the target contents accurately, and mapping them into a predefined data structure for further use.

### **Web Content Extraction Methods**

Since a web page usually contains not only the central content but also many auxiliary elements such as images and extraneous links, automatically identifying and extracting the target content to satisfy different research goals is vital to many applications. The most fundamental and challenging problem of Web content extraction is how to identify and extract the target contents needed.

HTML parser is widely used to extract contents from Web pages in either a linear or a nested fashion. Although different HTML parsers have different working schemes, they share the same input source – HTML pages – and parse Web contents into various formats, such as creating a Document Object Model (DOM) trees as [9] did or XML files, which reflect the HMTL tag hierarchy.

In the last decade, many works have been done on Web content extraction domain. At beginning, researchers started with the manual extraction and wrapper induction. Hammer et al. extracted semi-structured data from HTML pages and converted the information into database objects. Only heuristics based on text patterns is used to identify the boundary of the relevant data. McKeown et al. [36] used semantic boundaries to detect the largest body of the text on a Web page, but this straightforward method only works well for single-body documents with simple structures. Finn et al. [14] is another example that has a strict requirement on the Web page format. This method only extract Web contents from “single-article” sources, where content is presumed to be in a single body. Such manual-based methods can work well for large volume data with preknown document layouts. However, once the HTML page format is changed, the rules should be updated accordingly and are difficult to maintain and extend to new Web pages.

Later methods involving more advanced machine learning techniques were developed in the Web extraction process. A pioneer work for analyzing Web extraction tools and techniques is the RISE Web site (<http://www.isi.edu/integration/RISE/>) while a detailed survey and discussion can be found in [28]. Laender et al. classified the Web content extraction tools into the following categories:

- Languages for wrapper development: emphasizing the prevalent language to construct wrappers, such as Perl, PHP, or JAVA
- HTML-aware tools: taking advantage of Web page hierarchy by turning a Web page in a parsing tree. The representative tools include World Wide Web Wrapper Factory (W4F: <http://db.cis.upenn.edu/DL/WWW8/index.html>), XMRAP, and Lixto ([www.lixto.com](http://www.lixto.com)). Wrapper generation with Lixto is based on examples selected by users. All these tools share similar steps: identifying interesting regions on a HTML page, identifying important semantic tokens, and determining the nesting hierarchy of a page.
- NLP-based tools: building relationship between textual objects using filtering or part-of-speech tagging then deriving extraction rules; RAPIER [6] and WHISK [41] are representative NLP-based Web extraction tools, which apply multiple famous NLP techniques, such as filtering or part-of-speech tagging.
- Wrapper induction tools: generating rules based on training examples; WIEN [27] and SoftMealy are representative approaches that generate wrappers by learning from a set of real examples. Such methods can have good performance without many linguistic constraints. The quality of example pages is crucial to the performance of wrapper generation.
- Modeling-based tools: locating target data based on predefined models. Not surprisingly, researchers also tried to define models for Web pages based on the structure information and modeling primitives. NoDoSE [1] and DEByE [29] are two typical modeling-based Web content extraction tools.
- Ontology-based tools: relying directly on the data instead of structure features; Embley et al. [13] extracted Web objects based on ontology given in a specific domain. Such ontology-based methods can achieve a higher accuracy but the approach incurs high cost.
- Heuristic-based tools [17] is a newly published paper, which present an automatic approach to extract the main content of the Web page using tag tree and heuristics to filter the clutter and display the main content.

Although these methods vary at details, they fulfil a similar task: recognizing and extracting objects from a Web page and populating into a data repository. Based on various methods listed above, many researchers keep extracting Web contents in diverse ways: free-text searching, specific image collection (e.g., business logos or scientific charts), semi-structured document logical components (tabular structured data or scientific references), simplified document summarization, fact information detection supporting the quality of question-answering, document similarity analysis in both structure and semantic levels, etc. The methods to identify and extract a semi-structured document logical component – tables – from Web pages are elaborated in the next section.

## Web Table Extraction

Many powerful systems (e.g., OCTOPUS) use extracted Web tables as fundamental components for various further applications. Web table extraction usually contains two phases:

- Phase 1: Web table boundary detection and table rows/data cells identification
  - Table rows and data cells are delineated by the HTML markups such as `<TABLE></TABLE>` or `<th>/</th>`. Based on the tags, researchers usually convert the HTML table structure into a tree shape, which the children of each node are the next lower level of table elements.
  - Most HTML table detection relies on knowledge engineering techniques like heuristic rules based on the formatting cues and machine learning techniques (e.g., decision trees, Support Vector Machine, and wrapper learning, conditional random fields). In addition, they are highly domain specific.
    - Chen and Tseng [9] tested their algorithm based on heuristic rules and cell similarities on 918 HTML tables from airline information Web pages. Their table detection algorithm uses string, named entity and number category similarity to decide if it is a real or non-real table.
    - Hurst talked about a Naive Bayes classifier algorithm but there is no detail about the algorithm and the experiment [21]. They firstly mentioned the idea of rendering HTML code in order to use the results for detecting relational information in Web tables. Although they do not actually render documents in a browser, they can infer relative positional information of table nodes in an abstract table model.
  - Different from the traditional methods, visual rendition and topographical analysis is another representative method.
    - Kuprl et al. [26] used the visual rendition from the open-source Mozilla browser rather than the HTML tags to detect the Web tables. Their bottom-up clustering algorithm grouped visualized words into large groups and used a set of heuristics to detect tables unsupervised. The reported performance is about 70 %. In addition, there are several limitations: only those table types that are known in advance are supported by their algorithm. If there is a vertical gap in table cells, the algorithm currently does not deliver correct results. Also, the process is relatively slow, because it has to wait for Mozilla to render a page.
    - Gatterbauer and Bohunsky [16] provided an innovative approach to deal with the table extraction problem from Web pages by analyzing CSS2 visual box model. CSS represented textual HTML document elements by rectangular boxes, and a double topographical grid is adopted to manage all these visual boxes. A recursive spatial reasoning algorithm named VENTrec is proposed. VENTrec used keywords as input and tried to expand from any possible HyperBox on the grid into all directions until no possible expansion. The authors relied upon the positional and metadata information of visualized DOM element nodes in a browser. They believed

that the individual steps of table understanding become easier by taking advantage of visual cues of the Web.

- Phase 2: Identifying true Web tables by filtering out false Web tables
  - Some researchers work on the classification of HTML tables because HTML tables are often used to format documents instead of presenting data. To decide whether a table is worthy of further process, we should combine language and layout information for a better content judgement. The drawback of the inconsistent HTML markups tags incurs the difficulty of the further identification on many table components, e.g., table header lines.
  - Wang and Hu concentrated on the HTML table detection in [42]. They classified HTML tables into two types: genuine tables and non-genuine tables, according to whether a two-dimensional grid is semantically significant in conveying the logical relations among the cells. They introduced 16 features that reflect the layout as well as content characteristics of tables. These features are trained on thousands of examples, which are used to build a large table ground truth database.
  - WebTable (<http://windowsutilities.com/webtable.html>) is a recent work with the largest scalability. It extracted 14.1 billion raw HTML tables from the English documents in Google's main index. In order to filter out poor-quality non-relational tables, a series of techniques are adopted to recover those real tables. Only 1.1 % of raw HTML tables are confirmed as relational tables (genuine tables). The resulting tables can be imported into a spreadsheet or database program.

Based on all the above table extraction works, we can observe and summarize the following trends: (1) the extensibility and scalability are two important evaluation parameters that obtain more and more attention. (2) As the prevalent of machine learning methods, they are also widely used to extract Web tables and the performance beats that of rule-based methods in most cases (►Chap. 19 (Recognition of Tables and Forms)).

## Analysis of Text in Web Images

As the use of the Internet continues to grow and Web publishing technology advances, multimedia contents such as images contribute an increasingly more prominent proportion of the Web content. A survey by Petrie et al. [39] shows that among 100 home pages from 10 Web sites, there are on average 63 images per homepage. A significant portion of these images contains text and as such important semantic information about the documents in which the images are embedded.

### Motivation and Challenges

The need to embed text in images typically arises from the desire of the Web document author to present the text with an effect that is not possible using standard markup language features. This text is usually created with a graphics software tool, where the desired effects are applied to both the text and the background, and the

result is saved in image format (e.g., JPEG, GIF). This means the embedded text is usually text of high importance such as titles and headings, because the authors have devoted significant effort to make them “stand out” from the surrounding text. Despite the existence of means for alternative textual descriptions of images (e.g., the ALT tag), Web authors typically do not ensure the existence or consistency of these descriptions. Thus, being able to identify and recognize text embedded in image form using document image analysis tools is crucial for unlocking the semantic content and making it available for important downstream analyses.

The recognition of text in image form on the Web presents unique challenges. On the one hand, such text is free from variations and noises introduced in the scanning process. On the other hand, it has other properties that make it more difficult to process than scanned text encountered in traditional OCR. First, text images from the Web tend to be color images of very low resolution (e.g. 72 dpi) and the font size used for text is often very small (e.g. 5–7 pt). Such conditions pose a challenge to traditional OCR, which typically works with 300 dpi images that are mostly bi-level and character sizes of usually 10 pt or larger. Furthermore, images on Web documents tend to have various artifacts introduced by the authoring software. Finally, these images are often highly complex, with color variation not just in the background but in the foreground characters as well.

While some of these issues are shared by the related problem of recovering text from video sequences or natural scenes (►Chap. 25 (Text Localization and Recognition in Images and Video)), text recovery from Web images faces unique challenges. The recovery of text from video can leverage the temporal continuity of video frames. For example, text in adjacent frames tends to be in identical or similar color. In text extraction from natural scenes, a common assumption that the text as well as the immediate background of uniform color. This assumption does not apply for most Web text images.

## Approaches

The task of recognizing text in images can be decomposed into two main steps: (1) text isolation (also called localization), where the image is segmented such that regions corresponding to potential text components are separated from the background, and (2) text recognition, where strings of character-like components are recognized.

It should be clear that the unique challenges facing text recognition from Web images as identified above mostly affect the first stage of this process. Once this first stage is carried out, techniques from traditional OCR can be adapted with relative ease to carry out the rest of the task. Thus, the focus here will be on methodologies for text isolation from Web images.

### Methods Based on Traditional Color Space

Early approaches to text isolation relied largely on traditional clustering and quantization techniques. Lopresti and Zhou [32] proposed methods for text segmentation, extraction, and recognition. Their method for text segmentation and extraction is

based on clustering in the RGB color space and then for each cluster assessing connected components whose color belongs to that cluster. The approach works well with GIF images (only 256 colors) and when characters are of almost uniform color. With similar assumptions, the segmentation approach of Antonacopoulos and Hu [4] uses two alternative clustering approaches in the RGB space but works on (bit-reduced) full-color images (JPEG) as well as GIFs. Jain and Yu [22] proposed segmentation method based on decomposing an original image into a number of foreground images and a background one. The original number of colors (8- or 24-bit images) is reduced a to a much smaller number (between 4 and 8) of distinct colors by bit dropping and color quantization in the RGB space. Although this method produces good results for relatively simple images, it shares the same problems with the other methods for more complex images.

### Methods Based on the Perceptual Color Space

A drastically different approach based on perceptual color decomposition was proposed by Karatzas and Anotnacopoulos in [24]. This approach adopts a segmentation method based on analyzing differences in chromaticity and lightness that are closer to how humans perceive distinct objects, and has been demonstrated to work effectively under conditions of varying foreground and complex background.

The approach comprises four main steps. Note that the HLS (hue, Lightness, and saturation) color system is used throughout this approach. HLS is a perceptually oriented system that is more suitable for the analysis for images created to be viewed by humans than the traditional RGB system. The four steps are summarized below. Interested readers are referred to the original publication [24] for more details.

- Step 1. Chromatic/achromatic layer splitting

The first step is a preprocessing step based on information provided in published studies on color discrimination by humans in terms of expressions of minimum discrimination ability as a function of each of the physical properties of color (wavelength, color purity, and luminance) [45]. This step separates the chromatic from the achromatic content of the image for independent further processing. Chromatic color is any color for which a dominant wavelength can be identified (e.g., red, green, blue, yellow). A color is said to be achromatic if no dominant wavelength can be identified, i.e., shades of grey including black and white. Separating chromatic from achromatic pixels at this stage is important because any process that examines hue values cannot be successfully applied to achromatic pixels.

- Step 2. Further splitting

In this step subsequent splitting process is applied to identify and describe areas of perceptually similar color in the image. The chromatic and achromatic layers are both split into a number of layers with more refined color uniformity using global information derived from the hue or lightness histogram of each layer.

For the pixels of the achromatic layer, the histogram of lightness is computed and peaks are identified. If two or more peaks are present, consecutive peaks are analyzed by examining the horizontal distance (lightness value difference) between them and their corresponding height difference (ratio of peak maxima),

using thresholds determined in [45] and [23]. At the end of this peak analysis and combination process, the pixels in the layer that have lightness values under each final peak group are exported to a separate sub-layer.

For chromatic colors, first the histogram of the hue values is computed for the pixels of the chromatic layer and peaks are identified (in the same way as peaks of the lightness histogram in the achromatic layer). The horizontal distance (hue difference) between consecutive peaks and their corresponding difference in height is calculated and analyzed for potential combining using “similarity” criteria defined in [45] and [23]. The chromatic layer is thus split into sub-layers of different hue (each layer containing the range of hues under each of the final peaks). For each of the sub-layers produced, the lightness histogram is then computed, peaks are identified, and the peak analysis and grouping process is performed (as in the achromatic layer). New image sub-layers are created for pixels with lightness values in the ranges under each of the final peak groups.

A tree representation is produced after the splitting of the image into achromatic and chromatic layers and their corresponding sub-layers. In this tree, the root represents the original image and the nodes correspond to each of the layers produced by the splitting process. Each layer contains regions with a perceptually distinct hue or lightness.

- Step 3. Color-connected component analysis

To prepare for merging (Step 4), connected components are identified on each leaf layer, using a one-pass labeling algorithm [4] and the average color of each component is computed and stored. Each connected component corresponds to a region in the image that has an effectively distinct color. To identify these components, first the component *extensibility* is examined based on color similarity. All pixels that satisfy the relevant color similarity criterion are noted as the *vexed* area of that component. The topological relationship between any two components is then examined by checking the overlap between the two components when the corresponding vexed areas are taken into consideration. This lead to a measurement of overlap called *overlapping degree*.

- Step 4. Merging

In this final step, every possible pair of components in the leaf layer is examined, and if their overlapping degree is above a predefined threshold (experimentally derived 0.56), a possible merger is identified. All identified possible mergers in each layer are kept in a sorted list and merging starts with the pair of components with the highest overlapping degree. When two components are merged, the two core regions are combined to form the core region of the new component.

After each merger is completed, other potential mergers involving one of the two no-longer-existing components with an existing component are reassessed to take into account the resulting newly created component. After all possible mergers have taken place within each of the leaf layers, merging between the components of all sibling layers (across the same level) is performed. The merging process is repeated level by level, moving up in the tree structure until the root of the tree (the original image) is reached.

At the end of this four-step approach, the original image has been segmented into a number of connected components based on perceptual similarity.

These connected components can then be input to various traditional OCR engines for text line extraction and ultimately text recognition.

While this perceptual-based approach shows great promise, the evaluation of this method remains limited. Downstream analysis methods are needed to fully connect this approach, the state of the art in text line extraction and recognition, so that a complete system of text extraction and recognition applied to general images can be tested and eventually deployed for real applications.

## Web Document Modeling

An area closely related to Web content extraction is Web document modeling, e.g., the modeling of the geometric relationships among different components of various Web documents. While a significant amount of work has been dedicated to representing the geometric structure of a scanned document (►Chap. 6 (Analysis of the Logical Layout of Documents)), relatively little has been done to address the specific characteristics of Web documents. These special characteristics include:

1. The layout of a Web document is specified in the Cascading Style Sheet (CSS). However the CSS model is designed to be used by the Web browser's layout engine for the sole purpose of generating the visual representation of a Web page. It cannot be used to describe spatial relationships and configurations of the various visual elements.
2. A Web document often includes a much larger variety of objects than a scanned document. These objects may include navigation menu, Web form elements, and posts in a Web forum. As such, a Web document could be not only a resource for textual or media information but also an application.

Since work in this area is still in its very early stages and the definitive technique has yet to emerge, a survey of the existing work and pointers to a promising new direction are provided here.

Most of the work on representing geometric structure of a Web document has been carried out in a largely ad hoc manner, driven by the specific applications including predominantly Web content extraction and Web content classification. The most widespread representation is based on X-Y cut type of document segmentation as described in Section “[Web Page Segmentation](#)”, an approach borrowed from scanned documents. This leads to a layout model based on inclusion relations, where the whole document can be represented as a tree, and has been used primarily for content extraction and retrieval applications [12, 16]. Another type of representation is based on adjacent graphs such as 2D conditional random fields used for content extraction [48]. Finally, in some methods the DOM tree is augmented with visual characteristics to indirectly represent the geometric relationships for content extraction [34].

The first attempt at constructing a comprehensive geometric model of Web documents has just been reported recently. The authors propose a very general modeling framework called Web Page Geometric Model (WPGM). The model takes CSS boxes and their features as computed by the browser's layout engine as the

main building blocks, called Geometric Objects (GO), and provides mechanisms for both *quantitative modeling* (i.e., spatial position attribute) and *qualitative modeling* (i.e., spatial relationship among objects). The spatial relations supported include topological relations [11], distance between objects, directions, alignment [25], and 2D interval relations. Originally designed for Web accessibility, this approach has the potential of providing for the first time a unifying model that can be used for various applications including content extraction, Web page classification, and customized rendering. However its effectiveness and utility remains to be tested in these broad applications.

---

## Available Software

In this section, we introduce multiple representative open-source or business software and tools, which can analyze digital documents in PDF or HTML format. All these tools provide high-quality data sources and information foundation for further document analysis and understanding.

### PDF Analysis Tools

Since most PDF documents describes the low-level structural objects such as a group of characters, lines, curves, and images as well as associated style attributes such as font, color, stroke, fill, and shapes, accurately extracting all the objects with minimal potential component overlay and the interference between different logical components is a challenging problem. Both open-source and commercial converters are available. These converters usually only use PDF's low-level-based features due to PDF's complex generation process. The major state-of-the-art tools for PDF analysis and starting with those professional packages and libraries are analyzed in this section. All the listed tools are mature tools that are freely accessible and can be used directly.

- Adobe Systems Inc. provides a software developers kit (SDK) to extract logical words from a PDF file. This SDK gives access via the API of Acrobat viewers to the underlying portable document model. Within the viewers, there is a built-in algorithm that finds all the logical words within a PDF document for string-search purposes. This algorithm is sound for all single-column documents but can get confused when presented with multicolumn texts.
- JPedal (<http://www.jpedal.org/>), a library for Java development of PDF applications, has both commercial and open-source versions. End users submit PDF documents in the JPedal Web site and JPedal extracts text, images, annotations, and forms and notifies users the result of the extraction in XML format via emails. JPedal made good attempt on reading order resorting task.
- MatterCast (<http://www.mattercast.com/default.aspx>) has a commercial product called SVG Imprint. MatterCast can convert PDF to SVG in a single or batch way. The output in SVG can retain the original document layout and integrates all the

fonts embedded in PDF document. The raster images inside the PDF documents are extracted as separate file.

- BCL (<http://www.bcltechnologies.com/document/index.asp>) provides three products BCL Drake, BCL Freebird, and BCL Jade. All of them have either in input or output PDF documents. BCL Drake converts PDF documents into RTF for viewing and editing in Microsoft Word. BCL Freebird, as a plug-in for Adobe Acrobat, can convert PDF documents into graphics (Tiff, Jpeg, and Bmp). As another Acrobat plug-in, BCL Jade can extract text, tabular data, and graphics from PDF documents. However, this work can only be finished semi-automatically, with the help of mouse selection. Instead of being able to extract the images embedded inside a PDF page, BCL can only extract the image of the whole page.
- Glance (<http://www.pdf-tools.com/en/home.asp>) is another commercial PDF analysis product with a set of products and API. All of them have either in input or output PDF documents. The products include pdimgcomp (a PDF image compression, conversion, and extraction tool), Img2pdf and text2pdf (two products that convert images and text into PDF documents, respectively), pdcat useful for concatenation, pdsplit for splitting, pdsel for page selection, pdw for text extraction in ASCII or Unicode, and pdwebl for adding link annotations.
- Apache PDFBox™ (<http://pdfbox.apache.org/>) is an open-source Java PDF library for working with PDF documents. This project allows creation of new PDF documents, manipulation of existing documents, and the ability to extract content from documents.

## Web Document Analysis Tools

HTML parsers are implemented in different languages such as Java, Perl, or Python. The typical tools in Java include HTMP parser in <http://htmlparser.sourceforge.net/> and TagSoup in <http://java-source.net/open-source/html-parsers/tagsoup>.

The former introduces a fast JAVA library of real-time parser for real-world HTML pages, which supports two main functionalities: Web content extraction and transformation. Extraction encompasses all the information retrieval programs such as text extraction, link extraction, screen scraping, image/sound extraction, and other affiliated functions (link check as well as site monitoring). HTML parser produces a stream of tag objects, which can be further parsed into a searchable tree structure.

HtmlCleaner (<http://htmlcleaner.sourceforge.net/>) is another open-source HTML parser written in Java. It is designed to clean up the mess introduced by the original HTML page authors and reorganize the texts, attributes, and tags nicely. Similar to other HTML parsers, HtmlCleaner accepts HTML documents as input files. After processing and dirty cleaning, a well-formed XML file is generated.

NekoHTML (<http://nekohtml.sourceforge.net/>) is a similar HTML parser with cleaning and rendering functions. It scans and parses HTML files, fixes many common mistakes or unsuitable contents, then displays useful information using

standard XML interfaces. In addition, NekoHTML can fix the tag unbalance problem by providing custom tag and adding missing parent elements.

Other similar tools include HotSAX (<http://hotsax.sourceforge.net/>), Cobra 0.95.1 (<http://html.xamjwg.org/cobra.jsp>), JTidy (<http://jtidy.sourceforge.net/>), and VietSpider HTMLParser (<https://sourceforge.net/projects/binhgiang/>).

---

## Conclusion

This chapter provides a detailed account of the challenges, approaches, and the state of the art for the analysis of documents that originate from a digital form. The focus is on PDF and Web documents, two of the most dominate forms of digital document encoding. Challenges for both types of documents are centered around extraction of basic content elements, understanding of the layout and relationship among these elements, and organization of these elements into higher-level, semantically meaningful segments suitable for downstream applications such as information extraction and archiving, Web mining, and Web content repurposing. Addressing these challenges require a combination of advanced techniques drawn from the diverse fields of machine learning, computational linguistics, information retrieval, and image processing. Significant progress had been made in the last 15 years and many tools are now available that provide basic functionalities. However, challenges remain, particularly around the areas of extraction of complex basic elements (e.g., tables, text embedded in images), the understanding of pages with complex layout structures, and system extensibility and scalability for the efficient and effective handling of large number of documents. Some new directions have emerged, including perceptually based methods, graph theoretic approaches, and generative models. The full development and evaluation of these approaches provide significant and promising scope for future research.

---

## Cross-References

- ▶ [Analysis of the Logical Layout of Documents](#)
- ▶ [Page Segmentation Techniques in Document Analysis](#)
- ▶ [Recognition of Tables and Forms](#)
- ▶ [Text Localization and Recognition in Images and Video](#)

---

## References

1. Adelberg B (1998) NoDoSE – a tool for semi-automatically extracting structured and Semi-structured data from text documents. In: ACM SIGMOD international conference on management of data (SIGMOD'98), Seattle, pp 283–294
2. Ailon N, Charikar M, Newman A (2005) Aggregating inconsistent information: ranking and clustering. In: 37th STOC, Baltimore, pp 684–693

3. Anjewierden A (2001) AIDAS: incremental logical structure discovery in PDF documents. In: 6th international conference on document analysis and recognition (ICDAR), Seattle, Sept 2001, pp 374–378
4. Antonacopoulos A, Hu J (ed) (2004) Web document analysis: challenges and opportunities. World Scientific, Singapore
5. Cai D, Yu S, Wen J-R, Ma W-Y (2003) Extracting content structure for web pages based on visual representation. In: 5th Asia Pacific Web Conference, pp 406–415
6. Califf ME, Mooney RJ (1999) Relational learning of pattern-match rules for information extraction. In: Proceedings of the sixteenth national conference on artificial intelligence and the eleventh innovative applications of artificial intelligence conference, AAAI'99/IAAI'99, Orlando. Menlo Park, pp 6–11
7. Chakrabarti D, Kumar R, Punera K (2008) A graph-theoretic approach to webpage segmentation. In: WWW 2008, Beijing, pp 377–386
8. Chao H, Fan J (2004) Layout and content extraction for pdf documents. In: Marinai S, Dengel A (eds) Document analysis systems VI. Lecture notes in computer science, vol 3163. Springer, New York/Berlin, pp 13–224
9. Chen JS, Tseng DC (1996) Overlapped-character separation and construction for table-form documents. In: IEEE international conference on image processing (ICIP), Lausanne, pp 233–236
10. Chen Y, Xie X, Ma W-Y, Zhang H-J (2005) Adapting web pages for small-screen devices. Internet Computing, 9(1):50–56
11. Cohn AG (1997) Qualitative spatial representation and reasoning techniques, vol 1303. Springer, Berlin, pp 1–30
12. Duygulu P, Atalay V (2002) A hierarchical representation of form documents for identification and retrieval. IJDAR 5(1):17–27
13. Embly DW, Campbell DM, Jiang YS, Liddle SW, Lonsdale DW, Ng Y-K, Smith RD (1999) Conceptual-model-based data extraction from multiple-record web pages. Data Knowl Eng 31(3):227–251
14. Finn A, Kushmerick N, Smyth B (2001) Fact or fiction: content classification for digital libraries. In: Joint DELOS-NSF workshop on personalisation and recommender systems in digital libraries, Dublin, p 1
15. Futrelle RP, Shao M, Cieslik C, Grimes AE (2003) Extraction, layout analysis and classification of diagrams in PDF documents. In: International conference on document analysis and recognition (ICDAR) 2003, proceedings, Edinburgh, vol 2, p 1007
16. Gatterbauer W, Bohunsky P (2006) Table extraction using spatial reasoning on the CSS2 visual box model. In: Proceedings of the 21st national conference on artificial intelligence (AAAI), Boston, vol 2, pp 1313–1318
17. Gupta N, Hilal S Dr (2011) A heuristic approach for web content extraction. Int J Comput Appl 15(5):20–24
18. Hadjar K, Rigamonti M, Lalanne D, Ingold R (2004) Xed: a new tool for extracting hidden structures from electronic documents. In: Document image analysis for libraries, Palo Alto, pp 212–224
19. Hassan T (2009) Object-level document analysis of PDF files. In: Proceedings of the 9th ACM symposium on document engineering (DocEng'09), Munich. ACM, New York, pp 47–55
20. Hassan T (2009) User-guided wrapping of PDF documents using graph matching techniques. In: International conference on document analysis and recognition – ICDAR, Barcelona, pp 631–635
21. Hurst M (2001) Layout and language: challenges for table understanding on the web. In: Proceedings of the 1st international workshop on web document analysis, Seattle
22. Jain AK, Yu B (1998) Automatic text location in images and video frames. Pattern Recognit 31(12):2055–2076
23. Karatzas D (2002) Text segmentation in web images using colour perception and topological features. PhD Thesis, University of Liverpool

24. Karatzas D, Anotnacopoulos A (2007) Colour text segmentation in web images based on human perception. *Image Vis Comput* 25(5):564–577
25. Kong J, Zhang K, Zeng X (2006) Spatial graph grammars for graphical user interfaces. *CHI* 13:268–307
26. Krupl B, Herzog M, Gatterbauer W (2005) Using visual cues for extraction of tabular data from arbitrary HTML documents. In: Proceedings of the 14th international conference on world wide web (WWW), Chiba
27. Kushmerick N (2000) Wrapper induction: efficiency and expressiveness. *Artif Intell Spec Issue Intell Internet Syst* 118(1–2):15–68
28. Laender AHF, Ribeiro-Neto BA, da Silva AS, Teixeira JS (2002) A brief survey of web data extraction tools. *ACM SIGMOD Rec Homepage Arch* 31(2):84–93
29. Laender AHF, Ribeiro-Neto B, da Silva AS (2002) DEByE – date extraction by example. *Data Knowl Eng* 40(2):121–154
30. Lien Y-LL (1989) Apparatus and method for vectorization of incoming scanned image data. United States Patent US4,817,187, assigned to GTX Corporation, Phoenix, Arizona, 28 Mar 1989
31. Liu Y, Bai K, Mitra P, Lee Giles C (2007) TableSeer: automatic table metadata extraction and searching in digital libraries. In: ACM/IEEE joint conference on digital libraries, Vancouver, pp 91–100
32. Lopresti D, Zhou J (2000) Locating and recognizing text in WWW images. *Inf Retr* 2(2/3):177–206
33. Lovegrove W, Brailsford D (1995) Document analysis of PDF files: methods, results and implications. *Electron Publ Orig Dissem Des* 8(3):207–220
34. Luo P, Fan J, Liu S, Lin F, Xiong Y, Liu J (2009) Web article extraction for web printing: a DOM+visual based approach. In: Proceedings of the DocEng, Munich. ACM, pp 66–69
35. Marinai S (2009) Metadata extraction from PDF papers for digital library ingest. In: Proceedings of the 10th international conference on document analysis and recognition (ICDAR), Barcelona, pp 251–255
36. McKeown KR, Barzilay R, Evans D, Hatzivassiloglou V, Kan MY, Schiffman B, Teufel S (2001) Columbia multi-document summarization: approach and evaluation. In: Document understanding conference, New Orleans
37. Okun O, Doermann D, Pietikainen M (1999) Page segmentation and zone classification: the state of the art. Technical report: LAMP-TR-036/CAR-TR-927/CS-TR-4079, University of Maryland, College Park, Nov 1999
38. Oro E, Ruffolo M (2009) PDF-TREX: an approach for recognizing and extracting tables from PDF documents. In: ICDAR'09 proceedings of the 2009 10th international conference on document analysis and recognition, Barcelona, pp 906–910
39. Petrie H, Harrison C, Dev S (2005) Describing images on the web: a survey of current practice and prospects for the future. In: Proceedings of human computer interaction international (HCI), Las Vegas, July 2005
40. Smith PN, Brailsford DF (1995) Towards structured, block-based PDF. *Electron Publ Orig Dissem Des* 8(2–3):153–165
41. Soderland S, Cardie C, Mooney R (1999) Learning information extraction rules for semi-structured and free text. *Mach Learn Spec Issue Nat Lang Learn* 34(1–3):233–272
42. Wang Y, Hu J (2002) Detecting tables in HTML documents. In: Fifth IAPR international workshop on document analysis systems, Princeton, Aug 2002. Lecture notes in computer science, vol 2423, pp 249–260
43. Wang Y, Phillips IT, Haralick RM (2000) Statistical-based approach to word segmentation. In: 15th international conference on pattern recognition, ICPR2000, vol 4. Barcelona, Spain, pp 555–558
44. Wasserman HC, Yukawa K, Sy BK, Kwok K-L, Phillips IT (2002) A theoretical foundation and a method for document table structure extraction and decomposition. In: Lopresti DP, Hu J, Kashi R (eds) Document analysis systems. Lecture notes in computer science, vol 2423. Springer, Berlin/New York, pp 29–294

45. Wyszecki G, Stiles W (1982) Color science: concepts and methods, quantitative data and formulae, 2nd edn. Wiley, New York
46. Yildiz B, Kaiser K, Miksch S (2005) pdf2table: a method to extract table information from PDF files. In: Proceedings of the 2nd Indian international conference on artificial intelligence (IICAI05), Pune, pp 1773–1785
47. Zanibbi R, Blostein D, Cordy JR (2004) A survey of table recognition: models, observations, transformations, and inferences. *Int J Doc Anal Recognit* 7(1):1–16
48. Zhu J, Nie Z, Wen J-R, Zhang B, Ma W-Y (2005) 2D conditional random fields for web information extraction. In: Proceedings of the ICML'05, Bonn. ACM, pp 1044–1051

## Further Reading

Web Document Analytics, Apostolos Antonacopoulos and Jianying Hu (Editors), World Scientific, 2004.

PDF Explained, John Whitington, O'Reilly Media, 2011.

Web Data Mining: Exploring Hyperlinks, Contents, and Usage Data, Bing Liu, Springer, 2011.

---

# Image Based Retrieval and Keyword Spotting in Documents

24

Cheow Lim Tan, Xi Zhang, and Linlin Li

## Contents

Introduction.....	806
Background and History.....	806
Need for Document Image Retrieval.....	808
Chapter Overview.....	809
Word Image Representation.....	810
Character Shape Coding.....	810
Holistic Word Representation.....	811
Handwritten Word Representation.....	812
Keyword Spotting.....	817
Character Shape Coding-Based Keyword Spotting.....	817
Holistic Word Representation-Based Keyword Spotting.....	818
Keyword Spotting for Handwritten Documents.....	820
Summary of Keyword Spotting Methods.....	823
Document Image Retrieval.....	823
Retrieval Models.....	823
Character Shape Coding-Based Document Image Retrieval.....	826
Holistic Word Representation-Based Document Image Retrieval.....	828
Handwritten Document Image Retrieval.....	829
Summary of Document Image Retrieval Methods.....	832
Conclusion.....	834
Current State of the Art with Applications.....	836
Future Outlook.....	838
Cross-References.....	838
References.....	839
Further Reading.....	842

---

C.L. Tan (✉)

Department of Computer Science, National University of Singapore, Singapore, Singapore  
e-mail: [tancl@comp.nus.edu.sg](mailto:tancl@comp.nus.edu.sg)

X. Zhang • L. Li

National University of Singapore, Singapore, Singapore  
e-mail: [xizhang@comp.nus.edu.sg](mailto:xizhang@comp.nus.edu.sg)

---

**Abstract**

The attempt to move towards paperless offices has led to the digitization of large quantities of printed documents for storage in image databases. Thanks to advances in computer and network technology, it is possible to generate and transmit huge amount of document images efficiently. An ensuing and pressing issue is then to find ways and means to provide highly reliable and efficient retrieval functionality over these document images from a vast variety of information sources. Optical Character Recognition (OCR) is one powerful tool to achieve retrieval tasks, but nowadays there is a debate over the trade-off between OCR-based and OCR-free retrieval, because of OCR errors and wastage of time to OCR the entire collection into text format. Instead, image-based retrieval using document image similarity measure is a much more economical alternative. Till now, many methods have been proposed to achieve different sub-tasks, all of which contribute to the final retrieval performance. This chapter will present different methods for presenting word images and preprocessing steps before similarity measure or training and testing and discuss different algorithms or models for achieving keyword spotting and document image retrieval.

---

**Keywords**

Character shape coding • Document image retrieval • Holistic word representation • Keyword spotting

---

## Introduction

### Background and History

Over the years, various ways have been studied to query on document images using physical layout, logical structure information, and other extracted contents such as image features using different levels of abstraction. However, for document images containing predominantly text, the traditional information retrieval (IR) approach using keywords is still often used. For these document images, conventional document image processing techniques can be easily utilized for this purpose. For instance, many document image retrieval systems first convert the document images into their text format using Optical Character Recognition (OCR) techniques (for details about OCR techniques, please see ►Chap. 10 (Machine-Printed Character Recognition)) and then apply text IR strategies over the converted text documents. Several commercial systems have been developed based on this model, using page segmentation and layout analysis techniques and then applying OCR. These systems include Heinz Electronic Library Interactive Online System (HELIOS) developed by Carnegie Mellon University [1], Excalibur EFS, and PageKeeper from Caere. All these systems convert the document images into their electronic representations to facilitate text retrieval.

Generally the recognition accuracy requirements for document image retrieval are considerably lower than that for document image processing tasks [2].

A document image processing system will analyze different text regions, understand the relationships among them, and then convert them to machine-readable text using OCR. On the other hand, a document image retrieval system asks whether an imaged document contains particular words which are of interest to the user, ignoring other unrelated words. Thus, essentially a document image retrieval system answers “yes” or “no” to the user’s query, rather than exact recognition of characters and words as in the case of document information processing. This is sometimes known as keyword spotting with no need for correct and complete character recognition but by directly characterizing image document features at character, word, or even document level. Motivated by this observation, some efforts are concentrating on tolerating OCR recognition errors in document image retrieval.

However, high costs and poor quality of document images often prohibit complete conversion using OCR. Moreover, non-text components in a document image cannot be easily represented in a converted form with sufficient accuracy. Under such circumstances, it may be advantageous to explore techniques for direct characterization and manipulation of image features in order to retrieve document images containing text and other non-text components. So far, efforts made in this area include applications to word spotting, document similarity measurement, document indexing, and summarization. Among all these, one approach is to use particular codes, known as character shape codes [3], to represent characters in a document image instead of a full conversion using OCR. It is virtually a trade-off between computational complexity and recognition accuracy. This method is simple and efficient, but has the problem of ambiguity. Moreover, in order to obtain character codes, correct character segmentation is important which may not be possible in some cases, such as when characters are interconnected or overlapping resulting in segmentation errors. To address this issue, word-level methods are proposed which treat a single word as a basic unit for recognition or keyword spotting, or even segmentation free methods are proposed to avoid any segmentation errors.

In addition to segmentation, there are other important issues in document image retrieval research. Font shape is one such issue. Different fonts have their distinguished, intrinsic patterns. Thus features which are perfect for one kind of font may not be proper for another. Marinai et al. [4] proposes font adaptive word indexing method. However, font is one of the many characteristics the imaged document has, and many other features should be chosen carefully to adapt to different situations, such as different languages. Besides, inflectional languages also present partial word recognition problem. For example, “develop” is the partial word for “development,” “developed,” “develops,” etc. Complete matching will miss out other variants of the query word which should be retrieved in keyword spotting.

While imaged document retrieval has been mainly on printed document images in the past, there have been rising research interests on handwritten document image retrieval particularly in view of the increasing number of handwritten materials being scanned in digital format for safe keeping or for public access. For instance, valuable historical documents are digitized for permanent storage. Bank checks and handwritten receipts are scanned or photographed as records for

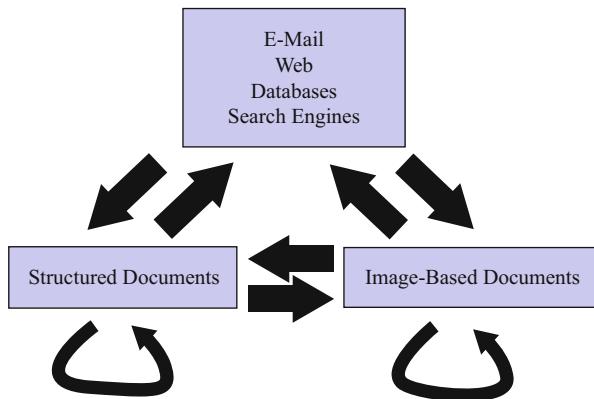
accounting purposes. Handwritten medical records are kept in image format for future reference. Handwriting recognition presents even much greater challenges to OCR which often fails miserably due to high variability of writing styles, severe noisy condition, and poor image quality (please see ►Chap. 12 (Continuous Handwritten Script Recognition) for challenges in handwritten text recognition). Various approaches based on raw and uncorrected OCR'ed text or direct image features have been studied [5] for handwritten document retrieval.

## Need for Document Image Retrieval

There is a perception in the industry and even among IT professionals that document image retrieval is only a temporary solution which will be phased out eventually due to technological progresses into the paperless age with online applications and publishing on the internet [6]. Paper-based documents are viewed as legacy documents which will disappear in the foreseeable future. The truth of the matter is that the rapid advancement in printing technology has in fact resulted in rapid proliferation of hard copy documents resulting in many “save trees” slogans by cutting down on printing. The call for the paperless office in the 1980s has in fact been met with even more papers [7]. On the other hand, affordable scanning and digital photography devices enable people to capture document contents as images with ease. Many paper-based documents ended up in image format rather than in their electronic equivalents. From a different perspective, the conversion of paper documents into image format appears to be a never-ending process for many years to come. This is because of the fact that there are still an enormous collection of documents and books in printed form that need to be converted to images. Huge collections of historical documents in libraries and national archives are waiting to be digitized before they deteriorate with time. Massive collection of out-of-print or rare books can only be preserved by turning them into images. Google's effort in digitizing books on a grand scale [8] is only touching the tip of an iceberg.

Yet from another perspective, there are various reasons that many documents will continue to appear in paper form. From a forensic point of view, paper-based documents are still much preferred over their electronic versions. Willful forgeries on papers, however sophisticated they may be, can be easily detected with modern forensic technology. Many legal documents, payment bills, and receipts must be presented in paper format for reasons such as the need for tangible proofs and personal or corporate privacy. For some legal requirements, paper-based documents should be scanned and OCRed, but they should be kept online in image format. The OCR results should be viewed as an “annotation” of the document image and not as the definitive representation. Publishers will continue to print books rather than making them available in electronic format for obvious reasons to prevent unauthorized copying and distribution. Today, in many parts of the world for some cultures and languages, paper-based documents are still prevalent in their societies. While advances in the internet world have led to many standards for semantics of electronic documents such as XML to facilitate search and retrieval,

**Fig. 24.1** Major role of image-based documents alongside with structured electronic documents in future document retrieval systems (From [6])

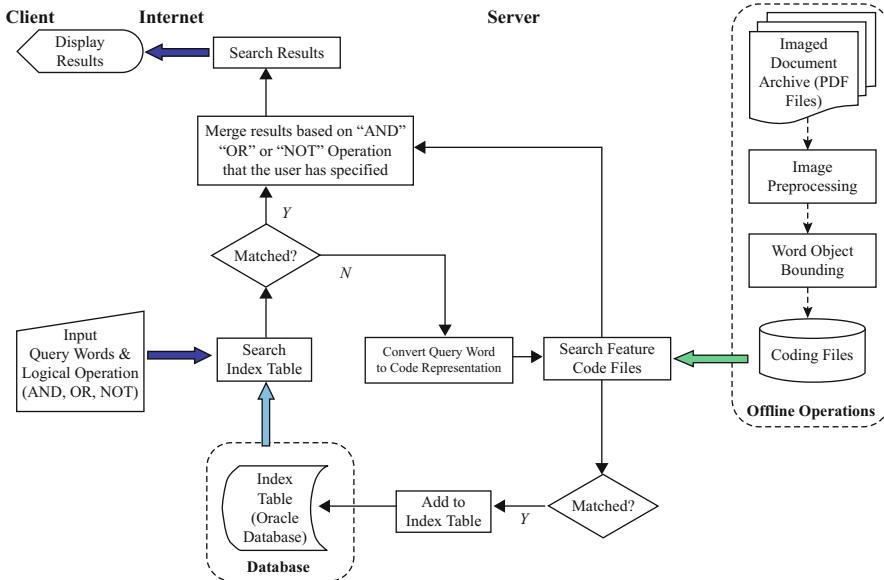


they are unable to handle document images. Document image retrieval definitely has a place in the modern world of information retrieval [6]. It is foreseen that in future document retrieval systems, image-based documents will play a major role alongside structured electronic documents as depicted in Fig. 24.1.

A scenario for a particular need for document image retrieval can be seen when there is a huge collection of legacy or historical document images scanned or captured with digital cameras under some very poor conditions. Doing OCR for the entire collection is an overkill as most of them may not be read after all, not to mention the likely poor quality output hampering readability. Instead, document image retrieval using word features to spot keywords and hence to facilitate rapid search for candidate documents in their original image rendition for easy reading will be a more economical and practical solution. A web application is presented in [9] for retrieving scanned legacy documents from a digital library. As shown in Fig. 24.2, all imaged documents are preprocessed to convert them into some code representation. When a user inputs a set of query words, the input words are converted into image features using the same coding scheme to allow speedy matching for document retrieval.

## Chapter Overview

In the following sections, methods for document image retrieval and keyword spotting will be introduced in detail. In section “[Word Image Representation](#),” different methods for presenting word images in Character Shape Coding, holistic word representation, and handwritten word representation will be first presented. Section “[Keyword Spotting](#)” will next describe how these different word representation schemes are used to achieve keyword spotting in printed and handwritten documents. Section “[Document Image Retrieval](#)” will then describe how the methods are used in retrieval of printed and handwritten document images. Finally, section “[Conclusion](#)” will conclude this chapter with an outlook for the future.



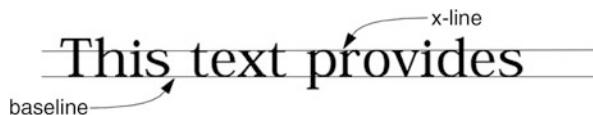
**Fig. 24.2** A web-based document image retrieval system (From [9])

## Word Image Representation

### Character Shape Coding

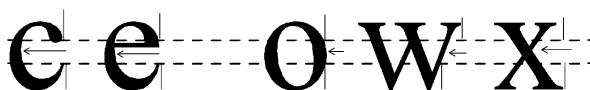
Due to the language dependency and the execution time of OCR, Character Shape Coding technique (referred to as word shape coding in some literature [10]) has been proposed. Character Shape Coding methods take an individual character as input and map character objects to a smaller symbol set. For example, English characters can be mapped to six different shape codes, by locating the baseline and x-line from the given text line images [3], namely, classifying characters based on ascender, descender, and dot. Figure 24.3 shows the positions of the baseline and x-line of an example text line, and Table 24.1 presents the mappings between shape codes and character images based on the positions of connected components in each character image. For example, the word shape coding representation for the word “left” is “AxAA,” which is named as a token, and a sequence of shape codes represents a word image. However, different word images may be mapped to the same token because of the confusions between several characters represented by the same sequence of shape codes. In order to reduce the confusions, characters represented by x shape code are further partitioned into two groups, based on the fact that confusions between e and s,r,a,o,n happen most frequently. Shown as Fig. 24.4, considering the middle part of the character image, it can be seen that “c” and “e” have a deep east concavity, but the others have little or none. Based on this

**Fig. 24.3** Positions of the baseline and x-line (From [3])



**Table 24.1** Mappings of character images to a set of shape codes

Shape code	Character images	Number of components	Component position	Component bottom position
A	A-Zbdfhklt	1	Above x-line	Baseline
x	acemnorsuvwxz	1	x-line	Baseline
g	gpqy	1	x-line	Below baseline
i	i	2	x-line	Baseline
j	j	2	x-line	Below baseline



**Fig. 24.4** “c” and “e” have a deep east concavity in their middle parts, but others also represented by x shape code have little or none (From [3])

observation, “c” and “e” are represented by a new shape code **e** and the rest are represented by **x** as before. Therefore, for a given text file, all generated sequences of shape codes are used for retrieval tasks, with fewer confusions.

Because the encoding (mapping) process is based on a set of simple and universal image features, Character Shape Coding methods are fast computable and language independent to a certain level. Therefore, they are widely employed in document image retrieval applications with speed constraints and language identification for Indo-European languages.

## Holistic Word Representation

Holistic word representation treats each word as a whole entity and thus avoids the difficulty of character segmentation. This is different from the character-level processing strategy taken by OCR and Character Shape Coding and is exactly why holistic word representation is robust to degraded image quality. In particular, broken and touching characters are one of the major document image degradation factors, and holistic word representation is naturally immune to them. It even works when some letters are ambiguous. Another important reason why holistic word representation presents an attractive alternative lies in its apparent similarity in the approach to how humans read text. Hull [11] points out the fact that according to psychological experiments, humans do not read text character by character, but, rather, they recognize words or even small groups of nearby words while they are reading. In the same way, features can be extracted at the word level instead of at the character level. Lu and Tan [12] proposed a word image coding technique,

**Table 24.2** Values of ADA and the corresponding positions of primitives

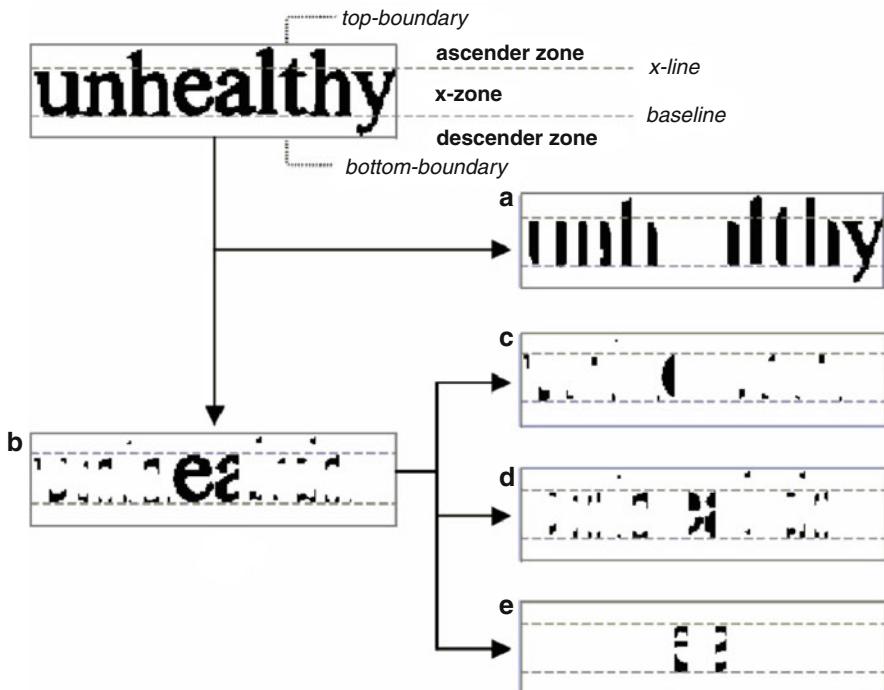
Character value of $\omega$	The corresponding position of the primitive
“x”	The primitive is between the x-line and the baseline
“a”	The primitive is between the top boundary and the x-line
“A”	The primitive is between the top boundary and the baseline
“D”	The primitive is between the x-line and the bottom boundary
“Q”	The primitive is between the top-boundary and the bottom boundary

which is a little more complicated compared with Spitz's method introduced in section “[Character Shape Coding](#),” but with an impressive advantage of much less ambiguity. For a given word image, it is segmented into several discrete entities from left to right, each of which is named as a primitive. A word image is represented by a sequence of features extracted from all primitives. There are two attributes used to describe a primitive: one is called Line-or-Traversal Attributes (LTA) denoted by  $\sigma$  and the other one is called Ascender-and-Descender Attribute (ADA) denoted by  $\omega$ . Therefore, the word image with  $n$  primitives are represented by  $<(\sigma_1, \omega_1)(\sigma_2, \omega_2) \dots (\sigma_n, \omega_n)>$ . For ADA,  $\omega$  has five character values, i.e.,  $\omega \in \Omega = \{x', a', A', D', Q'\}$ . Each character value represents a characteristic of one primitive, shown in Table 24.2. Different lines in Table 24.2 are illustrated in Fig. 24.5. For LTA, first, the straight stroke lines (SSL) are extracted from the whole word image, and only the vertical and diagonal strokes are under consideration. An example of extracted SSLs are shown in Fig. 24.5a. There are three types of SSLs: vertical stroke, left-down diagonal stroke, and right-down diagonal stroke, evaluated as “l,” “w,” and “v,” respectively. Furthermore, if a primitive has “x” or “D” ADA value, it is necessary to check whether there is a dot on top of the vertical stroke line, and if so, reassign “i” or “j” to the LTA value of the primitive. Next, traversal features are extracted from the remaining image, shown in Fig. 24.5b, by scanning each column.  $T_N$  is the traversal number which indicates the number of black-to-white transitions or vice versa in one column. Table 24.3 shows the feature codes for different values of  $T_N$ . The combination of the values of SSL features and traversal features are treated as the LTA value of the word image. Consequently, a word image is represented by a series of ADA and LTA values for all primitives, which can be further used to achieve information retrieval or keyword spotting tasks.

Holistic word representation approaches are widely employed in keyword spotting and document image retrieval for degraded document images when traditional OCR or Handwritten Word Recognition (HWR) fails, such as historical handwritten documents, scanned images of envelopes, and ancient (medieval) manuscript archives presenting a major challenge for OCR or HWR.

## Handwritten Word Representation

Scanned handwritten documents in image format are becoming common in digital libraries and other image databases. Google and Yahoo have now made handwritten



**Fig. 24.5** Features of primitives (From [12]). (a) Straight stroke line features, (b) remaining strokes after removing strokes in (a) from the original image, (c) traversal  $T_N = 2$ , (d) traversal  $T_N = 4$ , (e) traversal  $T_N = 6$

**Table 24.3** Feature codes for different values of  $T_N$

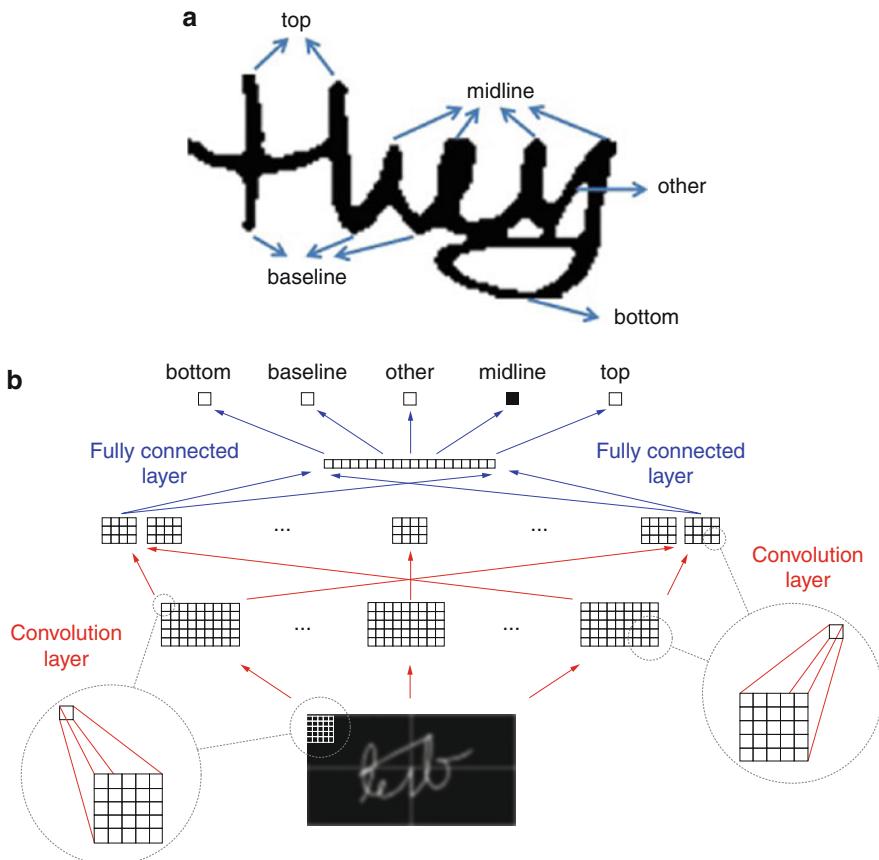
The value of $T_N$	Feature code
0	$\theta$ , indicating the space between two consecutive characters
2	$\kappa$ : the ratio of the number of black pixels to the x-height; $\xi = D_m/D_b$ : $D_m$ is the distance from the top-most pixel to the x-line, and $D_b$ is the distance from the bottom-most pixel to the baseline
4,6,8	Similar to $T_N = 2$

documents available for their search engines [13]. Due to severe degradation of such images because of noise contents and imperfect structure, OCR results are usually very poor with handwritten documents. It is therefore impossible to convert handwritten documents in full-text format and store it as such. Research on how to efficiently and correctly index and retrieve these handwritten imaged documents are hence imperative. Tomai et al. [14] discuss how difficult recognizing degraded

historical manuscripts is and that OCR is not a proper tool for historical manuscript recognition. Recent methods are based on keyword spotting for handwritten documents as an alternative and effective way for handwritten document retrieval. While many keyword spotting methods have been developed for printed document images with excellent performances, handwritten imaged documents are quite different from their printed counterparts due to their own particular characteristics. Hence, methods performing well for printed documents do not work well for handwritten documents. For handwritten documents, the following aspects require attention. Firstly, unlike printed documents which generally have predefined structures for different document types, the layout of handwritten documents is unconstrained. Many handwritten texts contain scribbling of characters in disarray and in any direction. Secondly, there is a high variability of writing styles among different writers. Even for the same writer, the writing style may change under different writing conditions. Thirdly, handwritten texts present an even much greater challenge for character or word segmentation as writing tends to exhibit a high degree of continuity between strokes. Antonacopoulos and Downton [15] discuss special problems in dealing with handwritten documents. See also ►Chap. 12 (Continuous Handwritten Script Recognition) on such problems.

In order to recognize handwritten documents, various methods for preprocessing steps have been proposed, such as text detection [16], line extraction [17], binarization, and noise reduction or removal [18]. In particular, baseline detection and slant removal are even more important which are used to reduce variations of writing styles. The simple method for baseline detection is based on the horizontal projection histogram, and a selected threshold is used to separate the main body part and the ascender or descender. Another supervised machine learning algorithm [19] trains a convolution classifier to classify each local extrema of a word image to locate the baselines. Figure 24.6 shows the definition of labels for each local extrema and how the convolution classifier works. This method can handle more complicated and cursive handwritten words. One of the global methods for slant correction is based on the vertical projection histograms of the shear transformed word images by all possible slant angles [20]. This kind of method tries to correct the slant by removing the slant angles for long vertical strokes in the word image. However, such global slant correction methods suffer from the assumption that all characters or long strokes are slanted by the same angle. To address this problem, local slant correction methods are proposed. After calculating the slant angles for each position of the word image, dynamic programming can be used to decide the optimal slant angle for each position, by propagating the slant angles of long verticals to their neighbors [21]. Figure 24.7 presents the main idea of local slant correction based on dynamic programming.

Furthermore, methods of representation of handwritten texts have been investigated, such as word segmentation using each word as a single unit for recognition [22] and text line segmentation to avoid word segmentation errors [23]. A sequence of feature vectors extracted from each column in the normalized word or text line images are used to represent handwritten documents. One feature vector is extracted

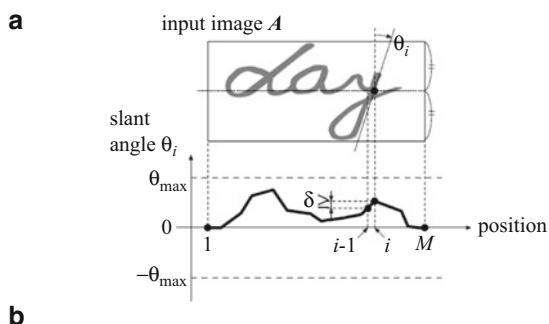


**Fig. 24.6** Baseline detection based on supervised machine learning. (a) Labels of local extrema. (b) The convolution classifier (From [19])

in a sliding column window with width of 1 pixel and moves from left to right along the whole word image. A set of usually used features are listed below:

1. The number of black pixels
  2. The center of gravity of the pixels
  3. The second order moment of the sliding window
  4. The location of the uppermost pixel
  5. The location of the lowermost pixel
  6. The orientation of the uppermost pixel
  7. The orientation of the lowermost pixel
  8. The number of black-white transitions
  9. The number of black pixels divided by the number of all pixels between the upper- and lowermost pixels

**Fig. 24.7** Local slant correction method based on dynamic programming (From [21]). (a) Slant angles for each position of the word image. (b) Dynamic programming for local slant correction



**Fig. 24.8** The first image shows the best links between disconnected CCs, namely, *lines* in between “W,” “in,” “ches,” and “ter.” The second image shows the corresponding binary mask, and the last image represents the final contour of the word image (From [24])

Besides, a single closed contour of a word image can also be used to represent handwritten words, even without the need for skew and slant correction [24]. Connected components (CCs) are first extracted from the binary word image, and small groups of pixels are discarded. CCs are ordered based on the horizontal positions of their gravity centers. Next, add the best connecting link between a pair of successive CCs. Only the links pairing the contour points of two CCs, the ends of which are both inside, or within a close distance to the main body part, or far from the main body part, are treated as the valid link. The shortest valid link is the best connecting link between two successive CCs. After all the best links are generated, the final binary mask are created and an ordered contour tracing procedure is applied to the mask to get the closed contour which is used to represent the word image. Figure 24.8 shows the intermediate results of the contour extraction.

## Keyword Spotting

### Character Shape Coding-Based Keyword Spotting

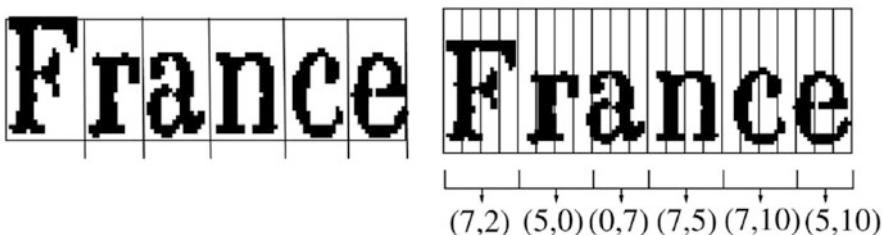
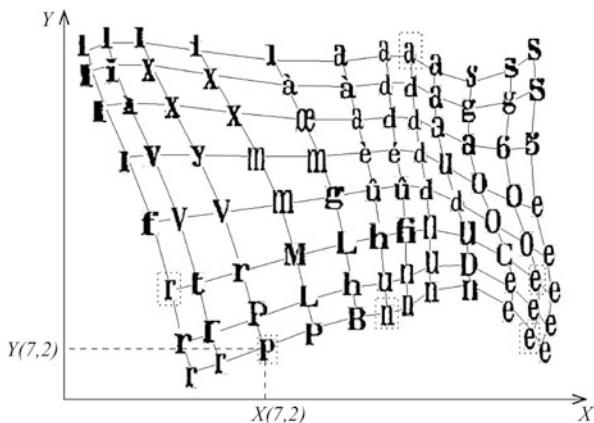
Character Shape Coding approaches still suffer from broken and touching characters in keyword spotting and thus is difficult to be applied to degraded document images. Marinai et al. [4] proposed a coding scheme, in which character shape codes are not explicitly mapped to their ASCII equivalent; instead, connected components, namely, the character objects (CO), are clustered by an unsupervised approach self-organizing map (SOM). Each character prototype CO used for CO labeling is scaled to a predefined size of grid, and the feature vector of the CO image is represented by density values of each grid item. For example, if the grid has the size of  $8 \times 10$ , the dimension of the feature vector is 80. The SOM is arranged into an  $N \times M$  feature map, and each neuron of the SOM,  $\text{SOM}(i, j)$ , indicates the center of one cluster.  $\text{SOM}(i, j)$  accepts the average of feature vectors of all COs assigned to the corresponding cluster, denoted by  $\text{Prot}(i, j)$  and other neurons from the map as input, so that  $\text{Prot}(i, j)$  can be treated as the pattern for a particular cluster. Then, the high-dimension feature vector  $\text{Prot}(i, j)$  is mapped to 2D space at the end of SOM training, which is refined by the Sammon mapping to denote the position of each neuron in the SOM map, namely,  $S(i, j) = (X(i, j), Y(i, j))$ . Figure 24.9 shows a trained SOM, each neuron of which represents the CO in the cluster closest to the prototype.

Based on the trained SOM, a query word can be represented as follows:

1. COs,  $\{\text{CO}_h(0), \text{CO}_h(1) \dots \text{CO}_h(n-1)\}$  are extracted from the query word image,  $W_h$ , each of which is then labeled with a SOM neuron,  $\text{SOM}(i_k, j_k)$ , whose  $\text{Prot}(i, j)$  has the smallest Euclidean distance to  $\text{CO}_h(k)$ .
2. Each CO is divided into a predefined number of vertical slices, and each slice is labeled by the SOM neuron of the largest CO overlapping it.
3. The word image is finally represented by the SOM neurons assigned to groups of slices, called the zoning vector,  $\vec{Z}_h$ . For example, as shown in Fig. 24.10, the word “France” is first separated into six COs, labeled as  $(7, 2), (5, 0), (0, 7), (7, 5), (7, 10), (5, 10)$ , respectively, and then each CO is further divided into three or four slices. The final representation of the word image “France” is  $\{(7,2), (7,2), (7,2), (7,2), (5,0), (5,0), (5,0), (5,0), (0,7), (0,7), (0,7), (7,5), (7,5), (7,5), (7,10), (7,10), (7,10), (7,10), (5,10), (5,10), (5,10)\}$ .

Thus the similarity between two word images is evaluated as the Euclidean distance between their zoning vectors. Since there is no direct mapping between ASCII characters and shape code strings, a word image has to be generated for any text query to calculate its Character Shape Coding string. This word shape code scheme is suitable for indexing printed documents for which current OCR engines are less accurate. This language independent and unsupervised system beat OCR in Boolean keyword retrieval application in six datasets of different fonts and languages.

**Fig. 24.9** The graph presentation of the trained SOM (From [4])



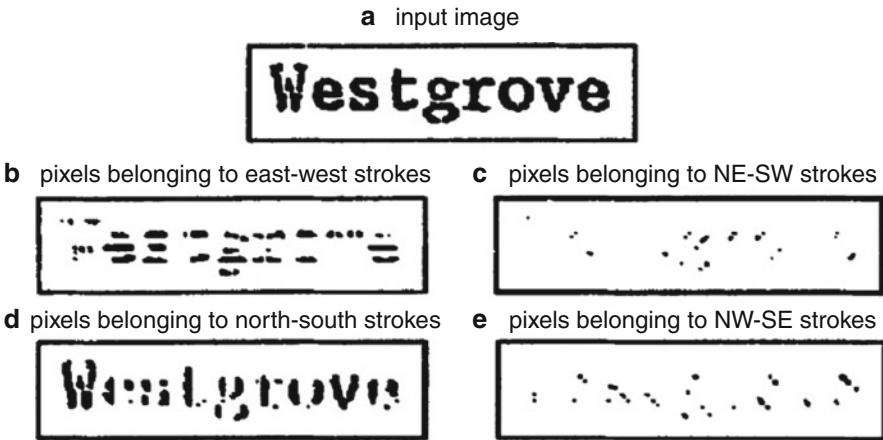
**Fig. 24.10** An example of the representation of a word image based on SOM (From [4])

### Holistic Word Representation-Based Keyword Spotting

Keyword spotting is to locate the occurrences of keywords in a document image. It is the most fundamental search type and the building block of Boolean document image search which treats any document with the presence of the keyword as relevant.

With holistic word representation-based keyword spotting, document images are firstly segmented into word images; features of word images are extracted and usually stored as a sequence of feature vectors. A word image synthesized from a set of character samples of various fonts is generated for the query. Feature vectors of the query are then computed. A distance  $d$  between the query word and a candidate word is computed, and an occurrence is found when  $d$  is below a certain threshold.

Ho et al. [25] proposed a method for word recognition in degraded images of machine-printed postal addresses on envelopes based on word shape analysis. Word images are first partitioned into a  $4 \times 10$  grid resulting in 40 cells. Local direction contribution is extracted from each cell in the grid and recorded in a feature vector, which stores the normalized number of pixels belonging to four categories of strokes (east-west, northeast-southwest, north-south, and northwest-south) as shown



**Fig. 24.11** An example of the four categories of strokes. The current run length of each black pixel is calculated in each of the four directions, and the pixel is labeled as one direction in which the run length of the pixel is maximum (From [25])

in Fig. 24.11. By concatenating all feature vectors, each image is represented by a 160-dimensional feature vector. Given an input image, the computed feature vector is compared to a given lexicon. Every word in the lexicon is synthesized into feature vectors for different font samples. Based on the distances calculated by summing over the absolute differences between corresponding feature components, a ranking list is produced. The top number of words in the ranking list are deemed to be most similar to the input image.

For document images where even word-level segmentation is not feasible, Leydier et al. [26] propose a method to detect zones of interest within an image. They explore several word image descriptors and point out that the gradient orientation is the best descriptor. Together with cohesive elastic matching algorithm, this method compares only informative parts of the query with only parts of the documents that may contain information, totally avoiding segmentation. Query images are manually obtained from the archive. In this approach identical words with different forms like singular and plural, “ing” and “ed,” might be classified as different classes. In their later work [27], a method to generate query image for text query is proposed by concatenating the glyphs for Alphabets, which are extracted from the target collection. With this method, grammatical variations of the keywords can also be searched at the same time.

Machine-printed words have basically similar forms even with degradation. The handwritten words, however, have much more variance in terms of visual appearance. Therefore, the generation of query image is very essential for the overall performance, especially when the word classes have severe variance, such as a handwritten image collection of multiple words, which will be discussed later.

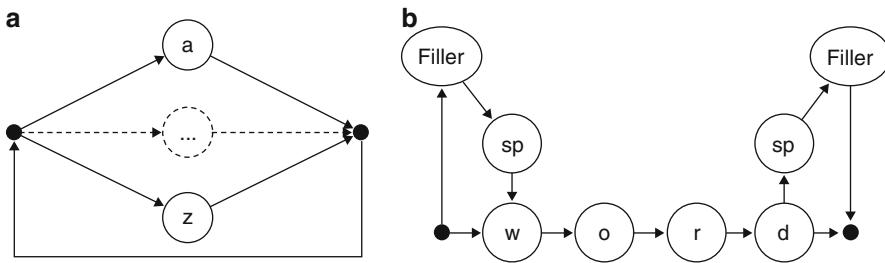
The keyword spotting paradigm, which is based on matching algorithms, may not be easily scalable to large document collections. It is very time-consuming to calculate the mutual distances between the query template and all word instances. Hence pruning, a fast rejection step in order to eliminate impossible word hypothesis, is often employed using simple word image features, such as the ratio of width to height, before the matching step.

## Keyword Spotting for Handwritten Documents

In [28], the authors present a holistic word approach to HWR. The approach treats each word as a single, individual entity and recognizes separated words based on their overall shape characteristics, rather than recognizing individual components in a word, such as using OCR. There are two kinds of HWR, off-line HWR which deals with the recognition tasks after the words have been written down already and online HWR which deals with the recognition tasks while the words are being written and temporal information such as position and speed of writing are collected for later recognition. Off-line HWR will be the focus in the following discussion. Before recognition, individual words are separated, and in order to achieve the segmentation and recognition, different features are extracted directly from the original images. The author presents a host of methods that perform the recognition of separated words or phrases with proper lexicon. The off-line HWR is limited, because it fully depends on the lexicon provided. Features which represent word image characteristics and are used for segmentation and recognition have important influence on performance and should be treated carefully.

A Hidden Markov Model (HMM)-based method which trains subword models can also be used to achieve keyword spotting for handwritten documents [29]. Especially, word segmentation is not needed and the keyword can be arbitrary which is not necessary in the training set. One HMM model is used to represent a character, so that a word or text line can be presented by concatenating a sequence of character HMMs together. Figure 24.12a shows a filter model which can generate any sequence of characters. Denoted by the filter model, it is possible to get a sequence of characters with a probability for a word or text line image. In order to achieve keyword spotting, a keyword model, shown in Fig. 24.12b, is used for decoding and a probability can be computed indicating how likely a text line image contains the keyword. In other words, if the probability is higher than a threshold, it is likely that the text line contains the keyword, otherwise, the keyword probably does not appear in the text line. The proposed method performs better than standard template matching algorithm based on Dynamic Time Warping (DTW) [30] and can adopt different writing styles because binary text line images are normalized prior to feature extraction in order to cope with different writing styles. This method is based on good separation and normalization of each text line.

In order to extract handwritten words, which is essential for final performance, Rodríguez-Serrano et al. [31] developed a supervised system to locate keywords in handwritten document images of multiple writers, with trained word class models,



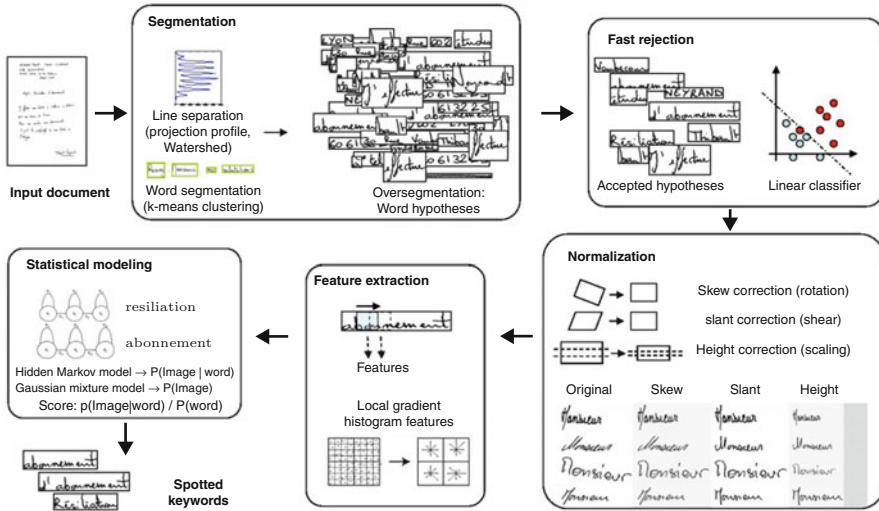
**Fig. 24.12** HMM model for keyword spotting (From [29]). (a) Filter model. (b) Keyword model

instead of character models. For a word image  $X$ , the probability to be word  $W_i$  is  $P(W_i|X)$ . From Bayes's rule, the probability will be

$$\log p(W_i|X) = \log p(X|W_i) + \log P(W) - \log p(X_i)$$

With a labeled training set,  $p(X|W_i)$  is modeled by HMM and  $p(X)$  is modeled using Gaussian Mixture Model (GMM). Note that  $\log p(W)$  is a constant for all  $W_i$ . In this way, the system need not implicitly generate the query image. Many features are tried to represent the word images, but, the best performance is obtained using local gradient histogram (LGH) features. They also found out that semicontinuous HMMs are superior when labeled training data is scarce as low as one sample per keyword, compared with continuous HMMs. A disadvantage of general HMM-based keyword spotting is that a sufficient large of training data is required. However, this spotting system, shown in Fig. 24.13, can easily be extended to word recognition given appropriate training data.

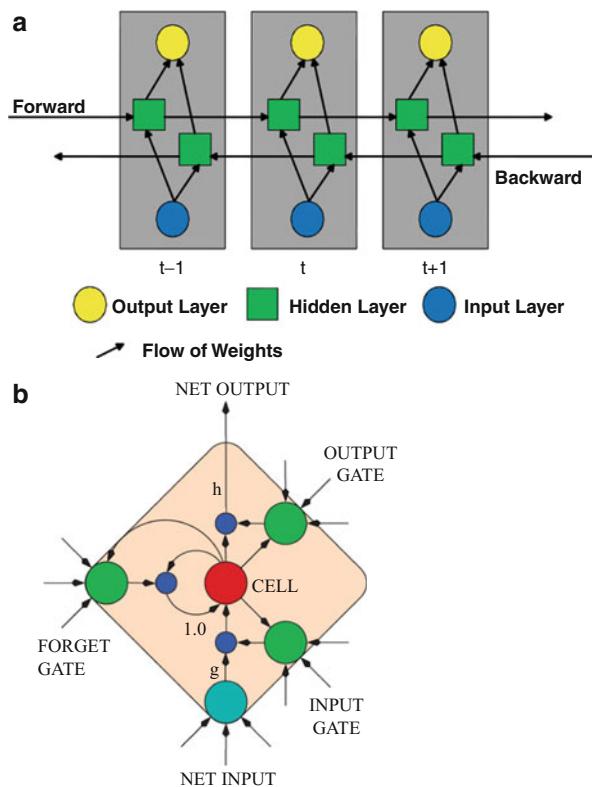
HMMs have several general drawbacks. First, the probability of the observation is only dependent on the current state, so that useful context information cannot be modeled easily. Secondly, HMMs are always generative and need to model the distribution of the observed variables, but sequence labeling tasks are discriminative. Finally, the number of hidden states for HMM models and the number of GMM models for each state are always decided on experimental efforts, and the estimation of these two parameters is prone to be task dependent. Based on these observations, Artificial Neural Network (ANN) is recently used in handwritten retrieval tasks. Particularly, Recurrent Neural Network (RNN), one kind of ANN, accepts one feature vector at one time and makes full use of all useful context information to estimate the output label. Thus, RNN can be an alternative to HMMs and also overcomes the drawbacks of HMMs. But, traditional RNN suffers one limitation in that it requires pre-segmented data, because a separate output label is required for each position in the input sequence. In order to overcome this limitation, a specially designed output layer Connectionist Temporal Classification (CTC) for RNN is used to directly map the input sequence to the label sequence with a probability, without the need of pre-segmented data.



**Fig. 24.13** Handwritten keyword spotting system (From [31])

Keyword spotting using BLSTM (Bidirectional Long Short-Term Memory) with CTC for off-line handwritten documents is introduced in [32]. Figure 24.14a shows a traditional RNN architecture, and Fig. 24.14b presents a special hidden block LSTM designed to address one limitation of traditional RNN, i.e., the range of context information RNN can accept is very limited in practice, which is called the vanishing gradient problem. The gates in the LSTM block are used to let RNN store and access information over a long period of time. During preprocessing, the document is segmented into individual text lines automatically, and feature vectors are extracted for each text line, which is then sent to the input layer of the BLSTM. There are as many nodes as the number of labels in the CTC output layer, plus one extra “blank” or “no-label” node.  $y_k^t$  denotes the probability of label  $k$  appearing at time  $t$  in the input sequence based on the output of the BLSTM, and  $p(\pi|x)$  denotes the probability of observing a path  $\pi$  given the input sequence  $x$  with length  $T$ , where  $\pi$  is a sequence of labels with the same length  $T$  and  $p(\pi|x) = \prod_{t=1}^T y_{\pi_t}^t$ ,  $\pi_t$  is the label at time  $t$  along  $\pi$ . There are many paths which may be mapped to the same label sequence by the function  $B$ , i.e.,  $B(\pi)$  removes the blanks and repeated labels in  $\pi$ , so that the probability of a label sequence  $l$  given  $x$  is the sum of probabilities of all possible paths mapped to  $l$ :  $p(l|x) = \sum_{\pi \in B^{-1}(l)} p(\pi|x)$ . The above equation allows the training of the BLSTM with CTC without pre-segmented data. For decoding, it is to find an  $l$  with the highest probability given the input  $x$ , combining with a dictionary and a language model  $G$ , which is formulated as  $l^* = \arg \max_l p(l|x, G)$ . Associated with a modified CTC Token Passing algorithm in [33] in order to get an approximate solution for  $l^*$ , unconstrained handwriting recognition and template-free keyword spotting are achieved. It is not necessary for

**Fig. 24.14** The structure of BLSTM Neural Network. (a) Bidirectional RNN with three time frames unfolded. (b) An LSTM hidden block with one cell for RNN (From [33])



the query keyword to appear in the training set. Furthermore, the method does not require word segmentation which makes it a very flexible matching paradigm.

## Summary of Keyword Spotting Methods

The keyword spotting methods presented in this section are summarized in Table 24.4 as a comparison in terms of their word representation, output, and their pros and cons.

---

## Document Image Retrieval

### Retrieval Models

Two statistical information retrieval (IR) models are widely used in relation to imaged document retrieval, namely, vector space (VS) model [34] and probability model [35].

**Table 24.4** Keyword spotting methods

Method	Word representation	Output	Pros and cons
Character shape Coding-based method [4]	A word image is divided into character objects each assigned an SOM neuron. Further subdivision of each character object into vertical slices allows representation of a word image as a neuron sequence forming a zoning vector	Similarity between two word images is based on the distance between their zoning vectors	This method is language independent and is able to tolerate degraded documents with broken or touching characters. But results are not satisfactory when documents contain words with large variant and unseen font style
Word shape-based approach [25]	Local stroke direction distribution extracted from each cell of the grid is used to present a word image	A ranking list of a given lexicon, with the top being the most likely word of the input word image	This method treats a word image as a whole and avoids error of character segmentation. It can deal with degraded documents with variable quality but may suffer from word segmentation errors
Zones of interest-based segmentation-free method [26]	The keyword image and the documents to be searched are divided into zones of interest(ZOIs) to allow mapping between ZOIs based on features of gradient orientation	Occurrences of the keyword in documents, ranked by score	This method avoids word segmentation errors. But some words with different forms, such as variant thickness of strokes or character height, may be classified into different classes. The method is also sensitive to fonts
HMM-based method [29]	Word or text line images are presented as a sequence of feature vectors, each extracted from an image column	Probability of a word or text line image containing the keyword and the starting and ending position for each character of the keyword	HMM models can be trained on word or text line images to avoid segmentation errors. However, transcripts of training data should be available and a large amount of training data is needed. HMM models may not make full use of context
RNN-based method [32]	Feature vectors extracted from each column of the input images	Likelihood of a word or text line image containing the keyword and the keyword position in the input image	RNN can incorporate more context information. It can deal with discriminative tasks, i.e., sequence labeling, better than HMM models which are always generative. RNN can also accept word or text line images as input to avoid segmentation errors. RNN however may incur more processing time

The basic assumption of VS model is that a fixed sized term set (lexicon) is used to characterize both documents and queries, and the terms are un-related (orthogonal) to each other. In particular, a lexicon  $T = \{t_1, t_2, \dots, t_\tau\}$  is predefined, and a document  $D$  is represented as  $\{w_{D,1}, w_{D,2}, \dots, w_{D,\tau}\}$  where  $w_{D,i}$  is the weight of term  $t_i$  in this document. The query  $Q$  is represented in the same manner. The similarity between  $Q$  and  $D$  is usually estimated by cosine distance:

$$\text{sim}(Q, D) = \frac{\sum w_{D,i} \times w_{Q,i}}{\sqrt{\sum w_{D,i}^2 \times w_{Q,i}^2}} \quad (24.1)$$

where  $w_{D,i}$  is a statistical measure used to evaluate how important a word is to a document in a collection.  $w_{D,i}$  could simply be the frequency of  $t_i$  in  $D$ . However, the most widely used term weighting scheme is  $tf.idf$  (term frequency and inverse document frequency).  $tf$  for term  $t_i$  is the number of occurrences of  $t_i$  in document  $D$  over the sum of number of occurrences of all terms in document  $D$ , while  $idf$  for term  $t_i$  is obtained by dividing the total number of documents by the number of documents containing the term and then taking the logarithm of that quotient. Other weighting schemes such as [36] are proven to be more efficient in text classification task.

Probability model ranks documents according to their relevance to the query. The ranking function is

$$f(D) = \log \frac{P(r|D)}{P(nr|D)} \quad (24.2)$$

where  $P(r|D)$  means the probability of  $D$  being relevant to the query, while  $P(nr|D)$  means the probability of being not relevant. Based on Bayes' Rules,

$$P(r|D) = \frac{P(D|r)P(r)}{P(D)} \quad (24.3)$$

and

$$P(nr|D) = \frac{P(D|nr)P(nr)}{P(D)} \quad (24.4)$$

Thus

$$f(D) = \log \frac{P(D|r)P(r)}{P(D|nr)P(nr)} \quad (24.5)$$

where these four probabilities could be estimated by a training step: given a training set,  $P(r)$  and  $P(nr)$  are the probability of relevant and irrelevant documents. With the term irrelevant assumption taken by the VS model,  $P(D|r)$  and  $P(D|nr)$  can be evaluated based on the presence (or absence) of individual terms  $t_i$  in the relevant set or irrelevant set. After simplification,

$$f(D) = \sum_{i=1}^n x_i \times \log \frac{P(x_i = 1|r)(1 - P(x_i = 1|r))}{P(x_i = 1|nr)(1 - P(x_i = 1|nr))}$$

where  $x_i$  is a binary representation of term  $t_i$ . If  $t_i$  is present in the collection  $x_i = 1$ , otherwise  $x_i = 0$ .  $P(x_i = 1|r)$  is the probability of  $t_i$  present in relevant documents, while  $P(x_i = 1|nr)$  is the probability of  $t_i$  present in the irrelevant documents. There are complicated formulae based on non-binary independence assumption, but they are not well tested.

Different ways are employed to evaluate these four probabilities. The most widely used one is proposed by Robertson and Sparck Jones [35]:

$$f(D) = \sum_{i=1}^{\tau} x_i \times \log \frac{r(i)(N - n(i) - R + r(i))}{(n - r(i))(R - r(i))} \quad (24.6)$$

where  $r(i)$  is the number of relevant documents containing a query term  $t_i$ ,  $N$  is the number of documents in the collection,  $n(i)$  is the number of documents containing  $t_i$ , and  $R$  is the number of relevant documents.

Normally a threshold is predefined, if the distance between the query and a document is within the threshold, the document is considered as “relevant” by the IR system. Alternatively, all top  $k$ -ranked documents are considered as “relevant.”

Two well-accepted measures for the evaluation of retrieval effectiveness are recall and precision. Recall is the ratio of the number of relevant documents returned to the total number of relevant documents in the collection, and precision is the ratio of relevant documents returned to the total number of documents returned.

Relevance feedback uses users’ judgments about the relevance of documents to select appropriate terms to expand the query.

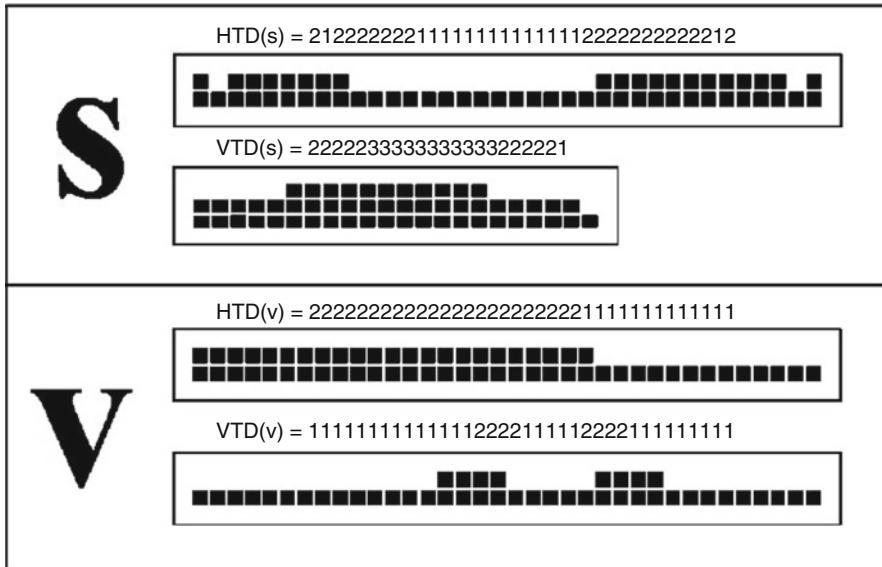
Typical query expansion process is as follows extract terms from relevant documents; weight and rank them; choose top  $k$  terms, either add them automatically or display them for user to select; add these new terms to the original query; and perform another search. A term re-ranking function proposed by Robertson and Sparck Jones [35] is as follows:

$$rw_i = \log \frac{(r(i) + 0.5)(N - n(i) - R + r(i) + 0.5)}{(n(i) - r(i) + 0.5)(R - r(i) + 0.5)} \quad (24.7)$$

## Character Shape Coding-Based Document Image Retrieval

For the Character Shape Coding-based document image retrieval, document images are firstly converted and stored as symbol strings. A query is translated into the same symbol representation by means of a table lookup. The search can be done at the text level.

In Character Shape Coding methods, each word is mapped uniquely to a corresponding symbol string, but one symbol string may be mapped to several real words because of the reduced symbol set, leading to ambiguity. For example, in Spitz’s coding scheme, both “left” and “loft” have identical shape code string



**Fig. 24.15** Two examples of character coding (From [37])

“AxAA.” The ambiguity level is different from method to method. It is an important retrieval performance indicator.

Tan et al. [37] propose a Character Shape Coding scheme for imaged document retrieval by text. The encoding was based on the vertical traverse density (VTD) and horizontal traverse density (HTD) of the character object, shown in Fig. 24.15. HTD is a vector indicating the numbers of line segments as scanning the character horizontally line by line from top to bottom. VTD is another vector obtained from vertical scanning from left to right. After characters are extracted from documents, each character object  $i$  can be represented by  $(HTD_i, VTD_i)$ . For two character objects,  $i$  and  $j$ , their distance  $D_{ij}$  is defined as

$$D_{ij} = \text{diff}(HTD_i, HTD_j) + \text{diff}(VTD_i, VTD_j) \quad (24.8)$$

where  $\text{diff}(V_i, V_j)$  is used to calculate the distance between two vectors. Based on the distance between any two character objects, all character objects in one document image can be grouped into a set of classes, each of which is presented by the mean of all objects in the corresponding class, namely, the centroid of the class. Because different document images can have different number of classes, and the centroids of classes may have different dimensions, normalization of the centroid vector is needed. If the original vector is  $V = v_1 v_2 \cdots v_n$ , the normalized vector is  $V' = v'_1 v'_2 \cdots v'_{n_c}$ , where the constant  $n_c$  is the dimension of the normalized vector, then  $v'_i = v_{(i \times n/n_c)}$ ,  $\forall i \in [1, n_c]$ . After all centroids of classes in all documents are normalized to the same dimension  $n_c$ , the distance between two classes are

computed, and if two classes have a very close distance, they are merged to the same class. Consequently, equivalent character objects from different documents can be denoted by the same unified class. A sequence of  $n$ -grams is obtained by moving a sliding window with  $n$  items along the text, one character forward at one time. In order to assign a unique number to every different  $n$ -gram and keep track of the frequencies of all distinctive  $n$ -grams, a hash table is created and used as a vector to represent a document image. So, the similarity between two document images is calculated as

$$\text{Similarity}(X_m, X_n) = \frac{\sum_{j=1}^J x_{mj} x_{nj}}{\sqrt{\sum_{j=1}^J x_{mj}^2 \sum_{j=1}^J x_{nj}^2}} \quad (24.9)$$

where  $X_i$  is the document vector of image  $i$  with  $J$  dimension, and  $X_i = x_{i1}x_{i2}\cdots x_{iJ}$ . This retrieval method is language independent and very useful for documents with similar font and resolution, without the use of OCR.

## Holistic Word Representation-Based Document Image Retrieval

Touching adjacent characters are difficult to be separated and present challenges to character coding methods. In view of this, some retrieval systems try to apply segmentation free methods to represent the entire word directly. In [12], the word image is expressed as  $P = < p_1 p_2 \cdots p_n > = < (\sigma_1, \omega_1)(\sigma_2, \omega_2) \cdots (\sigma_n, \omega_n) >$ , which has been introduced in Section “[Holistic Word Representation](#).” Due to the low resolution and poor quality, adjacent characters may be connected together, such that some features may be lost between two adjacent characters. Furthermore, noise can add or substitute features in the word image. An inexact feature matching is proposed to address such problems.  $V(i, j)$  is the similarity between two prefixes  $[a_1, a_2 \cdots a_i]$  and  $[b_1, b_2 \cdots b_j]$ , so that  $V(n, m)$  is defined as the similarity between two primitive strings  $A$  with length  $n$  and  $B$  with length  $m$ . Using dynamic programming,  $V(i, j)$  is formulated as

$$V(i, j) = \max \begin{cases} 0 \\ V(i - 1, j - 1) + \epsilon(a_i, b_j) \\ V(i - 1, j) + \mu(a_i, -) \\ V(i, j - 1) + v(-, b_j) \end{cases} \quad (24.10)$$

where  $1 \leq i \leq n, 1 \leq j \leq m$ , and  $\forall i, j, V(i, 0) = V(0, j) = 0$ . Matching a primitive with a spacing primitive “-” is defined as  $\mu(a_k, -) = v(-, b_k) = -1$  for  $a_k \neq -, b_k \neq -$ , and matching a primitive with a spacing primitive “&” is defined as  $\mu(a_k, \&) = v(\&, b_k) = -1$  for  $a_k \neq \&, b_k \neq \&$ . The score of matching two primitives is defined as

$$\epsilon(a_i, b_j) = \epsilon((\sigma_i^a, \omega_i^a), (\sigma_j^b, \omega_j^b)) = \varepsilon_1(\sigma_i^a, \sigma_j^b) + \varepsilon_2(\omega_i^a, \omega_j^b) \quad (24.11)$$

where  $\epsilon(\&, \&) = 2$ , and if the two elements are the same as  $\varepsilon_1$  or  $\varepsilon_2$ , its value is 1, otherwise,  $-1$ . Finally, the matching score of two images is formulated as

$$S_1 = \max_{i,j} V(i, j) / V_A^*(n, n) \quad (24.12)$$

where  $V_A^*(n, n)$  is the score of matching A with itself. This feature matching algorithm has the ability to match partial word. Given a document image, word objects are first extracted. If the ratio  $w/h$ , where  $w, h$  is the width and height of the word object, is smaller than a threshold  $\lambda$ , the word image is denoted as a stop word which provides little contribution to the similarity between document images and discarded. All the remaining word images are represented by the feature sequence of primitives. Based on the feature matching method introduced above, words in one document are grouped into different classes, in each of which the similarity score between any two word images is greater than a predefined threshold,  $\delta$ . The frequencies of all classes are used as the document vector, normalized by dividing the total number of word images in the document image. Consequently, the similarity between document  $\overline{Q}$  and  $\overline{Y}$  is

$$S(\overline{Q}, \overline{Y}) = \frac{\sum_{k=1}^K q_k \times y_k}{\sqrt{\sum_{k=1}^K q_k^2 \sum_{k=1}^K y_k^2}} \quad (24.13)$$

where  $q_i$  and  $y_i$  are the document vector of  $\overline{Q}$  and  $\overline{Y}$  respectively, and K is the dimension of the document vector, which is equal to the number of different classes in the document. Because of the inexact feature matching, the method can tolerate noise and low resolution and overcome the difficulties caused by touching characters. Shown as the experiment results, the proposed information retrieval method is feasible, reliable, and efficient.

## Handwritten Document Image Retrieval

Retrieval of handwritten document images remains a challenging problem in view of the poor recognition results from OCR, high variability in writing style, and low quality of many written documents such as historical manuscripts. Retrieval based on OCRed text is understandably very difficult though attempts have been made to clean up the OCR output. Other than that, retrievals are usually based on matching with word features in handwritten documents with query words which are converted using the same feature set. The query may be entered into the retrieval system by the user. It may also come in the form of printed or handwritten image as query. A recent approach for retrieval to historical documents makes use of the corresponding transcript for each document as a means for query.

Rath et al. [38] take advantages of the availability of transcribed pages for historical manuscript images, and the web-based retrieval system shown in Fig. 24.16.



**Fig. 24.16** The web-based retrieval system (From [38])

Word images in documents are extracted and holistic shape features are used to represent these word images, in order to avoid character segmentation errors caused by degradation. In [38], each word image is represented in terms of a discrete feature vocabulary (denoted as  $H$ ) as a kind of “image language.” A representation scheme is proposed consisting of 52 feature terms per word image, out of a feature vocabulary of size 494. A learning model is trained to map any given word image to a word from an English vocabulary  $V$  with a probability. If the training data is  $T$ , containing a set of word images with their transcripts, and each word  $W_i$ , the  $i$ th word in  $T$ , is represented by  $(h_i, \omega_i)$ , where  $\omega_i$  is the corresponding transcript from  $V$ , and  $h_i$  is represented by a set of features  $(f_{i,1} \dots f_{i,k})$  from  $H$ . So,  $W_i$  is denoted by  $(\omega_i, f_{i,1} \dots f_{i,k})$ . The probability of the distribution over  $W_i$  is formulated:

$$P(W_i = \omega, f_1 \dots f_k | I_i) = P(\omega | I_i) \prod_{j=1}^k P(f_j | I_i) \quad (24.14)$$

$$P(\omega, f_1 \cdots f_k) = E_i[P(W_i = \omega, f_1 \cdots f_k | I_i)] = \frac{1}{|T|} \sum_{i=1}^{|T|} P(\omega | I_i) \prod_{j=1}^k P(F_j | I_i) \quad (24.15)$$

with the assumption that elements in  $H$  and  $V$  are underlying multinomial probability distribution and observed values are i.i.d random samples. So that, based on the trained model, the probability of  $\omega$  is the transcription of the given testing word image, with feature vector  $(f_1 \cdots f_k)$ , is

$$P(\omega | f_1 \cdots f_k) = \frac{P(\omega, f_1 \cdots f_k)}{\sum_{v \in V} P(v, f_1 \cdots f_k)} \quad (24.16)$$

Consequently, given a query  $Q = q_1 \cdots q_m$  and a document page  $Pg$ , the probability of  $Pg$  containing the query is  $P(Q|Pg) = \prod_{j=1}^m P(q_j | Pg)$ , where  $P(q_j | Pg) = \frac{1}{|Pg|} \sum_{o=1}^{|Pg|} P(q_j | f_{o,1} \cdots f_{o,k})$ ,  $f_{o,1} \cdots f_{o,k}$  are the features for the word image at position  $o$  in  $Pg$ . This method for handwriting retrieval is called Probabilistic Annotation Model. Direct Retrieval is also applied by  $P(f, Q) = \sum_{W \in T} P(W)P(f|W)P(Q|W)$ ,  $f \in H$ . These two models can both retrieve unlabeled images with a text query without recognition, because the recognition always provides poor results for historical manuscripts.

A modified Vector Model named Multiple-candidate Vector Model is used to index and retrieve degraded medical forms with very low recognition accuracy [39]. The performance of information retrieval may decrease because of the incorrect recognition of the query word. But if the  $n$  top candidates given by the word recognizer are used for a query, the performance can be improved obviously. The count of the occurrences of the query word in one document is very essential for document retrieval, and poor recognition cannot provide valid and reliable counts. However, the count calculated based on the  $n$  candidates is more reliable, due to the fact that the true transcript of the query word is very likely in these  $n$  candidates. So, given a word recognizer  $WR$ , the output of which is denoted as  $O$ , and a collection of documents  $I$ , the frequency of a term  $t_i$  in a document  $d_j$ , is formulated as

$$tf_{i,j}^{ocr} = E\{tf_{i,j}|O\} = E\left\{\frac{\text{freq}_{i,j}}{\sum_l \text{freq}_{l,j}}|O\right\} = \frac{E\{\text{freq}_{i,j}|O\}}{\sum_l E\{\text{freq}_{l,j}|O\}} \quad (24.17)$$

$$E\{\text{freq}_{i,j}|O\} = \sum_{\omega \in \omega(d_j)} P(\omega = t_i) \quad (24.18)$$

where  $\text{freq}_{i,j}$  is the raw frequency of  $t_i$  in  $d_j$ ,  $l$  is any term in  $d_j$ , and  $\omega(d_j)$  is the set of words in  $d_j$ . For a word image  $\eta$  of word  $\omega$ ,  $WR$  gives a rank list of  $\eta$ ,  $(\omega_1, \omega_2 \cdots \omega_n)$ , so the rank of  $\omega_i$  is  $i$ , and the probability of  $\omega$  equal to  $\omega_i$  is denoted

by  $P^{(i)}$ . When considering the IDF of a term, documents which contain the term with  $P^{(i)}$  smaller than a threshold  $c_t \times P^{(1)}$  are discarded, so that

$$idf_i^{\text{ocr}} = \log \frac{|\{d_j\}|}{|\{d_j | \exists \eta \in d_j \text{ s.t. } P^{(\text{rank}_{\eta}^{(t_j)})} \geq c_t \times P^{(1)}\}|} \quad (24.19)$$

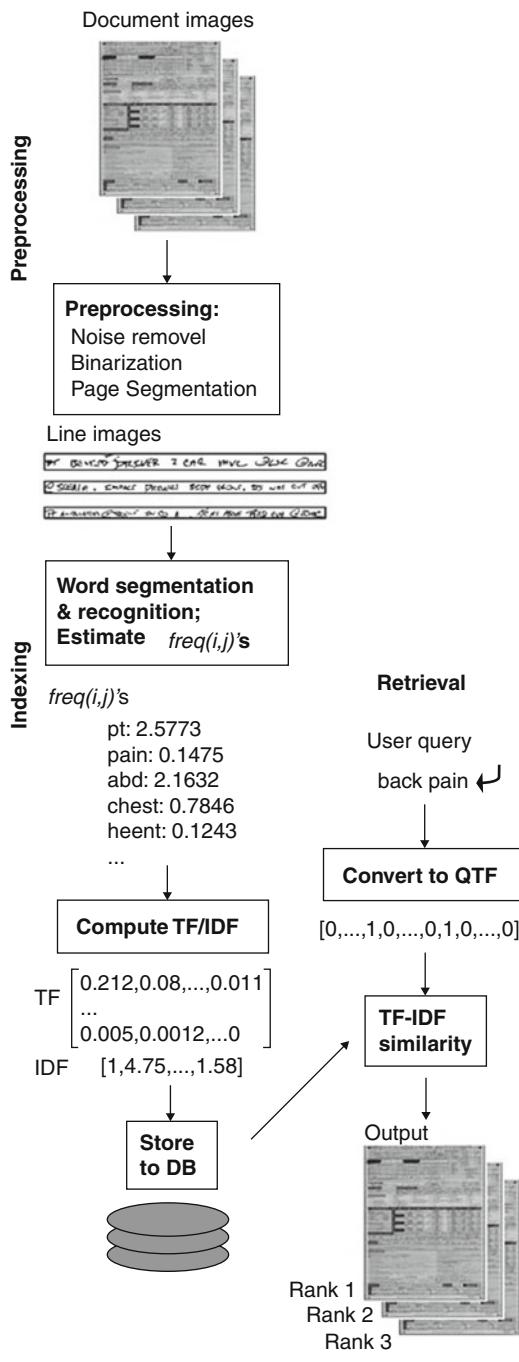
Based on the modified definition of  $tf$  and  $idf$  values, the modified vector model can retrieve and tolerate OCR errors of degraded medical forms very well.

Another different approach can be found in the work of [5] which aims to deal with imperfect recognition results of OCR due to illegible writing such as those in medical documents which often pose great challenges for retrieval tasks [40]. Figure 24.17 shows the flowchart of the search engine in [5]. Due to large variation in writing styles, low image quality, and large amount of medical words, the authors present three methodologies to solve the retrieval problem based on poor OCR recognition. The first one attempts to correct errors in OCR'ed text by a strategy based on lexicon reduction [41]. All the forms are separated into different categories according to their topics and then generate a lexicon of words which are used to extract phrases from the original recognized text by OCR. The extracted phrases are then used to obtain topic or category distribution for every document in the test set. For later recognition, the reduced lexicon and the corresponding document categories are used. In [42], the information about the document topic is also applied to compute the posterior probability of every term appearing in the document. The second method is based on modified IR model to tolerate the uncorrected OCR'ed text. Instead of computing the frequency of the terms directly from the document in the classic vector model, the modified model calculates the frequency based on the estimation of term count, word segmentation probability, and word recognition likelihood. After noise removal, normalization, and segmentation in preprocessing step, the modified vector model is used for the search engine. The query provided by the user is converted into the same vector format as input into the model and the test documents are ranked based on similarity with the input vector. The third method does not depend on the OCR'ed text, and instead deals with the retrieval tasks by image features, named keyword spotting, locating all the occurrences of the query word. Keyword spotting-based retrieval can be achieved by probabilistic word spotting model or feature-based word spotting model.

## Summary of Document Image Retrieval Methods

The document image retrieval methods covered in this section are summarized in the following two tables with comparison of their word representations, means of documents retrieval, and their pros and cons. Table 24.5 summarizes the methods for printed document image retrieval. Table 24.6 summarizes those for handwritten documents.

**Fig. 24.17** The flowchart of the proposed search engine  
(From [5])



**Table 24.5** Coding methods for printed document image retrieval

Coding method	Word representation	Document retrieval	Pros and cons
Symbol strings [3]	Each word is represented as a symbol string, where each symbol represents the shape of a component character	Documents are represented and indexed by a set of sequences of symbol strings for all their occurring words	This method reduces the size of symbol set, but at the expense of ambiguity due to symbol collision. Document degradation also leads for wrong symbol representation
VTD and HTD features [37]	Each character is represented by two vectors VTD and HTD through vertical and horizontal scanning	Documents are segmented into connected components to identify characters. Each document is then represented as a feature vector constructed from the occurrences of character classes based on VTD and HTD. Similarity between two document images based on feature vectors forms the basis for retrieval	This method is applicable to multilingual, different fonts documents and robust to degraded, low resolution documents
Holistic word coding [12]	Each word is treated as a whole object and represented by a sequence of features extracted directly from the whole word image	Word objects extracted from documents are segmented into primitives from left to right to form primitive strings. Documents are then represented as vectors based on classes of primitive strings of word objects to facilitate retrieval by means of similarity between document vectors	This method treats the word image as a whole and avoids character segmentation problems due to touching of characters. An inexact feature matching scheme is proposed to allow partial word matching

## Conclusion

Keeping documents as image format is a more economical and flexible alternative than converting imaged documents into text format by OCR. Furthermore, it is more robust for different variations and degradation. OCR always suffers from poor quality, low resolution, large distortion, nonuniform lighting, complicated layout, and deformations of imaged documents, such as historical documents with much noise and missing strokes, camera-based documents with larger distortions, whiteboard notes with complicated layout and signatures which have very large variations and cannot even be recognized by a normal recognizer. In recent decades, imaged document processing methods are proposed to deal with specific tasks, such

**Table 24.6** Handwritten document image retrieval methods

Method	Word representation	Document retrieval	Pros and cons
Probabilistic annotation model [38]	Words are represented by shape features and Fourier coefficients of profile features. To achieve cross-language task, the range of values of each feature is divided into several bins	Given transcripts and feature vectors of all words in the training set, a probabilistic model for retrieval can be trained to estimate the probability of a word transcript and a sequence of feature vector occurring together	This method is text-based retrieval method and can achieve multiple-words query tasks. But it suffers from queries which do not appear in the training set
Statistical classifier [39]	Each word is represented by a set of scores, each associated with a possible text transcript	The query word and all words in documents are converted into $N$ -best lists. Document relevance can be obtained by measuring the metric scores between $N$ -best lists	This method is text-based retrieval and is not dependent on word redundancy to overcome transcription errors. It can also query multiple words. However, transcriptions for the query word and documents must be available
Adapted vector model [5]	Documents and queries are represented by a vector space of term frequency (TF) and inverse document frequency (IDF) for each term in the vocabulary. TFs and IDFs are estimated and refined by use of word segmentation and recognition likelihood	Retrieval is achieved by measuring the similarity between vectors of query and data documents with a rank list	This method can overcome poor results of OCR and achieve queries on both printed or handwritten documents. But, it is still susceptible to OCR errors due to large variations in writing style or low image quality

as identifying different writers based on writing style, which cannot be achieved by OCR, and others try to solve general problems, i.e., keyword spotting and document retrieval, for different languages.

For research in keyword spotting in degraded image collections, holistic word representation is a popular approach. The key point of holistic word representation technique is to find good representation and distance measure that are sensitive to difference among word classes while tolerant to the variation within a word class. This is particularly so for handwritten document retrieval. From the literature, DTW-based comparison is appealing in terms of resilience to variances in handwritten words. HMM and ANN with appropriate training show even more powerful generalization ability to greater variances. Another issue in keyword spotting in

document images is the grammatical variation of the query word. Using multiple queries may be a solution, while interaction with the user for variant forms is another.

## Current State of the Art with Applications

With the development of computer technology, huge quantities of imaged documents whose text format versions are not available are stored in digital libraries for public access and permanent preservation. Document image retrieval technique is widely used in digital libraries [43]. For retrieval purpose, three steps are needed: document storage (or indexing), query formulation, and similarity computation with subsequent ranking of the indexed documents with respect to the query. There are two kinds of retrieval techniques available, recognition based and retrieval without recognition, such as keyword spotting. The former approach is based on the conversion into electronic text with the advantages of easy integration into a standard information retrieval framework and lower computational cost of similarity computation and results ranking. But this recognition-based method cannot deal with documents with high-level noise, multilingual content, nonstandard font, or a free-style variable layout. So, the latter kind of methods, which is recognition free, has been proposed to address the above problems. Because of the large scale of document categories, a variety of applications are possible. Based on recognition, it is possible to deal with retrieval from OCR'ed documents, citation analysis, handwriting recognition, layout analysis, and born-digital documents. On the other hand, without recognition, keyword spotting for image retrieval, recognition and retrieval of graphical items, dealing with handwritten documents, and documenting image retrieval based on layout analysis have led to some dedicated uses. Furthermore, even though recognition-free methods are promising, how to integrate them into a standard digital library which mostly adopts to OCR-based techniques is another important issue because of the difficulty in using sophisticated recognition-free matching algorithms [44]. One possible approach is to consider combining recognition-based and recognition-free methods together to improve the performance. Imaged document retrieval, whether recognition-based or recognition-free, is an answer to the current large scale document imaging projects such as the Google Book Search [8] and the million-page complex document image tested [45].

Many forms or document files, such as business contracts and legal agreements, contain signatures. Serving as individual identification and document authentication, signatures present convincing evidence and provide an important form of indexing for effective document image retrieval in a broad range of applications [46]. The authors address two fundamental problems for automatic document image search and retrieval using signatures. The first one is detection and segmentation including effectively segmenting the signatures from the background and choosing an appropriate meaningful presentations of signatures for later analysis. Detecting and segmenting signatures which present themselves as free-form objects is a

challenging task [47]. The authors use 2D contour fragments for each detected signature motivated by structural variations of off-line handwriting signatures. This kind of 2D point features provide several advantages, because the preserved topology and temporal order under structural variations or clustered background are not needed compared to other compact geometrical entities used in shape representation. The second problem pertains to the use of matching for determining whether two given objects belong to the same class. Proper representations, matching algorithm, and similarity measure method should be investigated so that retrieval results can be invariant to large intra-class variability and robust under interclass similarity. The authors use two kinds of nonrigid shape matching algorithms: one is based on shape context representation [48], and the other formulates shape matching as an optimization problem that preserves local neighborhood structure [49]. Like the problem that [46] aims to solve, with large repository of business documents, retrieving documents signed by the person provided by the user, in other words, using a query signature image to retrieval from databases, is an interesting task [50]. In a similar vein, indexing and retrieval of Logo [51] or architectural symbols [52] are also state of the art applications of document image retrieval technology.

A new emerging application, though not really on paper-based media, has found practical use of document image retrieval techniques. This involves image capture of whiteboard writing to serve as records of meetings and notes taking for classroom lessons. However, whiteboard writing recognition presents far greater challenges than paper-based handwriting recognition. Writing on the whiteboard tends to be highly disorganized and haphazard. The writing style is free and extremely variable due to the writing posture while standing and the use of different types of marker pens. The writing contains not only text but also a host of different kinds of drawing comprising lines, shapes, and symbols. This requires research into separating different parts of the whiteboard contents into individual functional units, such as texts and graphs, without any prior knowledge of text lines, fonts, character sizes, etc. [53] recognizes handwritten sketches and [54] deals with mind maps [55]. Liwicki and Bunke [56] propose a method for online whiteboard note processing using an off-line HMM-based recognizer. To do so, it uses a special pen-tracking hardware, known as eBeam system which is restricted and suffers from obvious flaw of preventing natural interaction. The data is collected online to get idealized text lines. In the preprocessing step, the authors apply online and off-line preprocessing operations to solve two problems of the recorded online data: noisy points and gaps within strokes. And then, online data is transformed into off-line format as input for the basic off-line recognizer which is an HMM-based cursive handwriting recognizer described in [57]. Most whiteboard image processing works rely simply on camera capture to provide only off-line data. For instance, [58] works on camera-based whiteboard images under real-world conditions obtaining only low quality images with distortion. In this work, text lines are detected and ink contrast is enhanced. Using a sliding window, the normalized extracted text lines are subdivided into a sequence of overlapping stripes which will be presented as semicontinuous HMMs. The HMM-based writing model is then integrated with a

bigram language model to improve the performance. In another application, [59] uses two cameras and a pen capture tool on the whiteboard to recognize Japanese characters based on some character matching.

While whiteboard images have been mostly captured with video or digital cameras mounted in front or above the whiteboard, a new generation of electronic whiteboards (also known as interactive whiteboard) has made it possible to directly capture whiteboard contents using built-in imaging device. This facilitates instant generation of whiteboard images which are free of occlusion by the writer and with considerably better image quality. Thus electronic whiteboard opens exciting applications of document image retrieval to index and recall past imaged records and notes of meetings.

## Future Outlook

Looking ahead, efficiently and correctly storing, indexing and retrieving a large amount of imaged documents is an imperative task. This is especially so considering Google and Yahoo are working on making millions of imaged documents accessible through the internet with their search engines. In such situations, rapid and satisfactory responses are quite important. So, large collection recognition with high computation speed, recall, and precision rate is an important issue. And, querying on multiple words, even very long words, effectively should also be researched into. Until now, many methods have been tried to solve recognition problems for languages such as English and Chinese with reasonably good performance, but there are many other languages in which a number of valuable documents are written and which are not yet widely explored. While other language character or word features should be studied, language and script independent models should be researched into also. Such models will be needed for multilingual documents involving different languages or scripts on one single document page.

Today, the pervasive use of digital cameras and mobile phone cameras in the society has led to the creation of many more document images or scene images with text captured under different conditions. While image resolution is often not an issue with advances in camera technology, many such images suffer from unfavorable effects for imaged document retrieval and keyword spotting. These include uneven lighting, curved, or warped document surfaces such as those taken from open pages of thick books, perspective distortion, and complex background in scene text capture. Archiving and storing of such document images on public web sites or other repositories entail similar tasks like indexing, text content-based retrieval, and searching for query words. Exciting challenges are lying ahead for which new paradigms and models will continue to emerge in the years to come.

---

## Cross-References

- ▶ [Continuous Handwritten Script Recognition](#)
- ▶ [Handprinted Character and Word Recognition](#)

- ▶ [Language, Script, and Font Recognition](#)
- ▶ [Machine-Printed Character Recognition](#)
- ▶ [Page Similarity and Classification](#)
- ▶ [Text Segmentation for Document Recognition](#)

---

## References

1. Galloway EA, Gabrielle VM (1998) The heinz electronic library interactive on-line system: an update. *Public-Access Comput Syst Rev* 9(1):1–12
2. Taghva K, Borsack J, Condit A, Erva S (1994) The effects of noisy data on text retrieval. *J Am Soc Inf Sci* 45(1):50–58
3. Spitz AL (1995) Using character shape codes for word spotting in document images. In: Dori D, Bruckstein A (eds) *Shape, structure and pattern recognition*. World Scientific, Singapore, pp 382–389
4. Marinai S, Marino E, Soda G (2006) Font adaptive word indexing of modern printed documents. *IEEE Trans Pattern Anal Mach Intell* 28(8):1187–1199
5. Cao H, Govindaraju V, Bhardwaj A (2011) Unconstrained handwritten document retrieval. *Int J Doc Anal Recognit* 14:145–157
6. Breuel TM (2005) The future of document imaging in the era of electronic documents. In: Proceedings of the international workshop on document analysis, IWDA'05, Kolkata. Allied Publishers, pp 275–296
7. Sellen AJ, Harper RHR (2003) The myth of the paperless office. MIT, Cambridge/London
8. Vincent L (2007) Google book search: document understanding on a massive scale. In: Proceedings of the international conference on document analysis and recognition, Curitiba, vol 2. IEEE, pp 819–823
9. Zhang L, Tan CL (2005) A word image coding technique and its applications in information retrieval from imaged documents. In: Proceedings of the international workshop on document analysis, IWDA'05, Kolkata. Allied Publishers, pp 69–92
10. Lu S, Li L, Tan CL (2008) Document image retrieval through word shape coding. *IEEE Trans Pattern Anal Mach Intell* 130(11):1913–1918
11. Hull JJ (1986) Hypothesis generation in a computational model for visual word recognition. *IEEE Expert* 1(3):63–70
12. Lu Y, Tan CL (2004) Information retrieval in document image databases. *IEEE Trans Knowl Data Eng* 16(11):1398–1410
13. Levy S (2004) Google's two revolutions. *Newsweek*, December 27:2004
14. Tomai CI, Zhang B, Govindaraju V (2002) Transcript mapping for historic handwritten document images. In: Proceedings of the eighth international workshop on frontiers in handwriting recognition, 2002, Niagara-on-the-Lake. IEEE, pp 413–418
15. Antonacopoulos A, Downton AC (2007) Special issue on the analysis of historical documents. *Int J Doc Anal Recognit* 9(2):75–77
16. Indermuhle E, Bunke H, Shafait F, Breuel T (2010) Text versus non-text distinction in online handwritten documents. In: Proceedings of the 2010 ACM symposium on applied computing, Sierre. ACM, pp 3–7
17. Liwicki M, Indermuhle E, Bunke H (2007) On-line handwritten text line detection using dynamic programming. In: Ninth international conference on document analysis and recognition, ICDAR 2007, Curitiba, vol 1. IEEE, pp 447–451
18. Zimmermann M, Bunke H (2002) Automatic segmentation of the iam off-line handwritten english text database. In: 16th international conference on pattern recognition, Quebec, vol 4, pp 35–39
19. Simard PY, Steinkraus D, Agrawala M (2005) Ink normalization and beautification. In: Proceedings of the eighth international conference on document analysis and recognition 2005, Seoul. IEEE, pp 1182–1187

20. Vinciarelli A, Luettin J (2001) A new normalization technique for cursive handwritten words. *Pattern Recognit Lett* 22(9):1043–1050
21. Uchida S, Taira E, Sakoe H (2001) Nonuniform slant correction using dynamic programming. In: Proceedings of the sixth international conference on document analysis and recognition, 2001, Seattle. IEEE, pp 434–438
22. Manmatha R, Han C, EM Riseman, Croft WB (1996) Indexing handwriting using word matching. In: Proceedings of the first ACM international conference on digital libraries, Bethesda. ACM, pp 151–159
23. Likforman-Sulem L, Zahour A, Taconet B (2007) Text line segmentation of historical documents: a survey. *Int J Doc Anal Recognit* 9(2):123–138
24. Adamek T, O'Connor NE, Smeaton AF (2007) Word matching using single closed contours for indexing handwritten historical documents. *Int J Doc Anal Recognit* 9(2):153–165
25. Ho TK, Hull JJ, Srihari SN (1992) A word shape analysis approach to lexicon based word recognition. *Pattern Recognit Lett* 13(11):821–826
26. Leydier Y, Lebourgeois F, Emptoz H (2007) Text search for medieval manuscript images. *Pattern Recognit* 40(12):3552–3567
27. Leydier Y, Ouji A, Lebourgeois F, Emptoz H (2009) Towards an omnilingual word retrieval system for ancient manuscripts. *Pattern Recognit* 42(9):2089–2105
28. Madhvanath S, Govindaraju V (2001) The role of holistic paradigms in handwritten word recognition. *IEEE Trans Pattern Anal Mach Intell* 23(2):149–164
29. Fischer A, Keller A, Frinken V, Bunke H (2010) Hmm-based word spotting in handwritten documents using subword models. In: 2010 international conference on pattern recognition, Istanbul. IEEE, pp 3416–3419
30. Myers CS, Habiner LR (1981) A comparative study of several dynamic time-warping algorithms for connected-word. *Bell Syst Tech J* 60(7):1389–1409
31. Rodríguez-Serrano JA, Perronnin F (2009) Handwritten word-spotting using hidden markov models and universal vocabularies. *Pattern Recognit* 42(9):2106–2116
32. Frinken V, Fischer A, Bunke H (2010) A novel word spotting algorithm using bidirectional long short-term memory neural networks. In: Schwenker F, El Gayar N (eds) Artificial neural networks in pattern recognition. Springer, Berlin/Heidelberg, pp 185–196
33. Graves A, Liwicki M, Fernández S, Bertolami R, Bunke H, Schmidhuber J (2009) A novel connectionist system for unconstrained handwriting recognition. *IEEE Trans Pattern Anal Mach Intell* 31:855–868
34. Salton G, Wong A, Yang CS (1975) A vector space model for automatic indexing. *Commun ACM* 18(11):613–620
35. Robertson SE, Sparck Jones K (1976) Relevance weighting of search terms. *J Am Soc Inf Sci* 27(3):129–146
36. Lan M, Tan CL, Low HB (2006) Proposing a new term weighting scheme for text categorization. In: Proceedings of the 21st national conference on artificial intelligence, Boston
37. Tan CL, Huang W, Yu Z, Xu Y (2002) Imaged document text retrieval without OCR. *IEEE Trans Pattern Anal Mach Intell* 24:838–844
38. Rath TM, Mammatha R, Lavrenko V (2004) A search engine for historical manuscript images. In: Proceedings of the 27th annual international ACM SIGIR conference on research and development in information retrieval, Sheffield. ACM, pp 369–376
39. Cao H, Farooq F, Govindaraju V (2007) Indexing and retrieval of degraded handwritten medical forms. In: Proceedings of the workshop on multimodal information retrieval at IJCAI-2007, Hyderabad
40. Cao H, Bhardwaj A, Govindaraju V (2009) A probabilistic method for keyword retrieval in handwritten document images. *Pattern Recognit* 42(12):3374–3382
41. Milewski RJ, Govindaraju V, Bhardwaj A (2009) Automatic recognition of handwritten medical forms for search engines. *Int J Doc Anal Recognit* 11(4):203–218
42. Bhardwaj A, Farooq F, Cao H, Govindaraju V (2008) Topic based language models for ocr correction. In: Proceedings of the second workshop on analytics for noisy unstructured text data, Singapore. ACM, pp 107–112

43. Marinai S (2006) A survey of document image retrieval in digital libraries. In: 9th colloque international Francophone Sur l'Ecrit et le document (CIFED), Fribourg, pp 193–198.
44. Aschenbrenner S (2005) Jstor: adapting lucene for new search engine and interface. *DLib Mag* Vol. 11, no. 6
45. Agam G, Argamon S, Frieder O, Grossman D, Lewis D (2007) Content-based document image retrieval in complex document collections. In: Proceedings of the SPIE, vol 6500. Document Recognition & Retrieval XIV, San Jose.
46. Zhu G, Zheng Y, Doermann D (2008) Signature-based document image retrieval. In: Computer vision–ECCV 2008, Marseille, pp 752–765
47. Zhu G, Zheng Y, Doermann D, Jaeger S (2007) Multi-scale structural saliency for signature detection. In: 2007 IEEE conference on computer vision and pattern recognition, Minneapolis. IEEE, pp 1–8
48. Belongie S, Malik J, Puzicha J (2002) Shape matching and object recognition using shape contexts. *IEEE Trans Pattern Anal Mach Intell* 24:509–522
49. Zheng Y, Doermann D (2006) Robust point matching for nonrigid shapes by preserving local neighborhood structures. *IEEE Trans Pattern Anal Mach Intell* 28:643–649
50. Srihari SN, Shetty S, Chen S, Srinivasan H, Huang C, Agam G, Frieder O (2006) Document image retrieval using signatures as queries. In: Second international conference on document image analysis for libraries 2006, DIAL'06, Lyon. IEEE, p 6
51. Jain AK, Vailaya A (1998) Shape-based retrieval: a case study with trademark image databases. *Pattern Recognit* 31(9):1369–1390
52. Terrades OR, Valveny E (2003) Radon transform for lineal symbol representation. *Doc Anal Recognit* 1:195
53. Weber M, Liwicki M, Dengel A (2010) a.Scatch-a sketch-based retrieval for architectural floor plans. In: 2010 12th international conference on frontiers in handwriting recognition, Kolkata. IEEE, pp 289–294
54. Vajda S, Plotz T, Fink GA (2009) Layout analysis for camera-based whiteboard notes. *J Univers Comput Sci* 15(18):3307–3324
55. Burzan T, Burzan B (2003) The mind map book. BBC Worldwide, London
56. Liwicki M, Bunke H (2005) Handwriting recognition of whiteboard notes. In: Proceedings of the 12th conference of the international graphonics society, Salerno, pp 118–122.
57. Marti UV, Bunke H (2001) Using a statistical language model to improve the performance of an hmm-based cursive handwriting recognition system. *IJPRAI* 15(1):65–90
58. Plotz T, Thurau C, Fink GA (2008) Camera-based whiteboard reading: new approaches to a challenging task. In: Proceedings of the 11th international conference on frontiers in handwriting recognition, Montreal, pp 385–390
59. Yoshida D, Tsuruoka S, Kawanaka H, Shinogi T (2006) Keywords recognition of handwritten character string on whiteboard using word dictionary for e-learning, International Conference on Hybrid Information Technology, Cheju Island, Vol. 1, pp 140–145
60. Konidaris T, Gatos B, Ntzios K, Pratikakis I, Theodoridis S (2007) Keyword-guided word spotting in historical printed documents using synthetic data and user feedback. *Int J Doc Anal Recognit* 9(2):167–177
61. Lu Y, Tan CL (2004) Chinese word searching in imaged documents. *Int J Pattern Recognit Artif Intell* 18(2):229–246
62. Zhang H, Wang DH, Liu CL (2010) Keyword spotting from online chinese handwritten documents using one-vs-all trained character classifier. In: 2010 12th international conference on frontiers in handwriting recognition, Kolkata. IEEE, pp 271–276
63. Senda S, Minoh M, Ikeda K (1993) Document image retrieval system using character candidates generated by character recognition process. In: Proceedings of the second international conference on document analysis and recognition, 1993, Tsukuba. IEEE, pp 541–546
64. Sagheer MW, Nobile N, He CL, Suen CY (2010) A novel handwritten Urdu word spotting based on connected components analysis. In: 2010 international conference on pattern recognition, Istanbul. IEEE, pp 2013–2016

65. Moghaddam RF, Cheriet M (2009) Application of multi-level classifiers and clustering for automatic word spotting in historical document images. In: 2009 10th international conference on document analysis and recognition, Barcelona. IEEE, pp 511–515
66. Leydier Y, Le Bourgeois F, Emptoz H (2005) Omnilingual segmentation-free word spotting for ancient manuscripts indexation. In: Proceedings of the eighth international conference on document analysis and recognition, 2005, Seoul. IEEE, pp 533–537
67. Mitra M, Chaudhuri BB (2000) Information retrieval from documents: a survey. *Inf Retr* 2(2):141–163
68. Murugappan A, Ramachandran B, Dhavachelvan P (2011) A survey of keyword spotting techniques for printed document images. *Artif Intell Rev* 1–18
69. Marinai S, Miotti B, Soda G (2011) Digital libraries and document image retrieval techniques: a survey. In: Biba M, Xhafa F (eds) *Learning structure and schemas from documents*. Springer, Berlin/Heidelberg, pp 181–204

## Further Reading

Methods that have been introduced are mainly proposed and experimented on English documents. Other languages or scripts, such as Chinese and Arabic, have quite different patterns and writing styles from English and have special characteristics which cannot be retrieved by methods for English. For example, some features used to differentiate different characters or words in English documents may not be applicable for Chinese words. Readers who are interested in non-English documents may refer to [60–64] which deal with retrieval of words or documents in languages such as Arabic, Chinese, Japanese, and Urdu. Besides, [65, 66] report language independent methods. One method is highlighted below for multilingual documents retrieval as an illustration.

The Character Shape Coding method proposed by Lu et al. [10] can be used for different languages, namely, multilingual tasks. There are five codes: hole, ascender, descender, leftward water reservoir, and rightward water-reservoir. A word is represented by encoding these features from left to right, top to bottom. The same coding scheme is applied in document similarity measurement task for five languages: English, French, German, Italian, and Spanish.

Imaged-based retrieval and keyword spotting in documents have been researched for many years with many more works not possibly covered in this chapter. Readers may wish to consult the following three survey papers for comparative views of different methods. Among these papers, an earlier one [67] covers a broad area of document-based information retrieval which gives some of the historical perspectives of this field, including retrieval of OCR'ed text. The other two are much more recent surveys. One [68] presents different keyword spotting techniques for printed documents, while the other [69] provides a comparative study of document image retrieval in the context of digital libraries involving printed and handwritten documents.

---

# Text Localization and Recognition in Images and Video

# 25

Seiichi Uchida

## Contents

Introduction.....	844
Recognition of Texts in Images.....	845
Background.....	845
Applications.....	845
Recognition of Texts in Video.....	848
Background.....	848
Applications.....	848
Three Tasks and Their Difficulties.....	849
Text Image Acquisition.....	850
Text Localization.....	850
Text Recognition.....	851
Methods to Solve Problems in Image Acquisition.....	852
Low Resolution.....	854
Blurring.....	856
Perspective Distortion.....	857
Nonuniform Lighting.....	860
Methods to Solve Problems in Text Localization.....	861
Typical Process of Text Localization.....	861
Four Aspects Characterizing Text Localization Methods.....	861
Features for Text Localization.....	862
Methods of Text/Non-text Discrimination.....	865
Region Types for Text Localization.....	867
Method of Revision by Considering Neighboring Regions.....	867
Methods to Solve Problems in Text Recognition.....	869
Scene Character Recognition Using Standard OCR Techniques.....	869
Specific Features for Scene Character Recognition.....	873
Recognition of Decorative Characters.....	873
Dataset for Evaluation.....	874

---

S. Uchida

Department of Advanced Information Technology, Kyushu University, Nishi-ku, Fukuoka, Japan  
e-mail: [uchida@ait.kyushu-u.ac.jp](mailto:uchida@ait.kyushu-u.ac.jp)

Conclusion.....	875
Cross-References.....	875
References.....	875
Further Reading.....	882

---

### Abstract

This chapter reviews techniques on text localization and recognition in scene images captured by camera. Since properties of scene texts are very different from scanned documents in various aspects, specific techniques are necessary to localize and recognize them. In fact, localization of scene text is a difficult and important task because there is no prior information on the location, layout, direction, size, typeface, and color of texts in a scene image in general and there are many textures and patterns similar to characters. In addition, recognition of scene text is also a difficult task because there are many characters distorted by blurring, perspective, nonuniform lighting, and low resolution. Decoration of characters makes the recognition task far more difficult. As reviewed in this chapter, those difficult tasks have been tackled with not only modified versions of conventional OCR techniques but also state-of-the-art computer vision and pattern recognition methodologies.

---

### Keywords

Scene text detection • Scene text recognition • Text image acquisition • Text localization • Text/non-text discrimination • Video caption detection • Video caption recognition

---

## Introduction

Nowadays, most textual information is captured by camera rather than scanner. Camera-based OCR is the technology to recognize characters captured by camera. Since camera is far handier than scanner and possible to capture various textual information in scene (such as characters on a signboard), realization of camera-based OCR will develop new applications of OCR. On the other hand, characters captured by camera are different from those in scanned documents and their recognition is very difficult due to various reasons. For example, their appearance is often distorted by perspective, complex background, blur, low resolution, nonuniform lighting, and special decoration. In addition, their localization is also difficult because there is no prior knowledge of their location and there are many character-like non-character patterns. For the solution of those difficulties, we have to not only evolve the traditional image processing methods but also incorporate the state-of-the-art pattern recognition technologies. In addition, camera-based OCR deals with a “real-world” problem; general computer vision technologies are more and more important for OCR. This chapter introduces those problems and solution technologies to the realization of camera-based OCR.

## Recognition of Texts in Images

### Background

In past researches, the target of OCR is mainly document images acquired by scanner; this is because scanner can provide images with sufficiently high resolution for OCR. Assume that each character should have 32 pixels in its height for an OCR. In this case, for an A4-sized document ( $8.3 \times 11.7$  in.) with 40 text lines and line spacing of 18 pixels, the height of the entire document image must be larger than 2,000 pixels. This indicates that resolution of the scanner should be higher than 170 dpi. For dealing with more complex characters, such as Chinese and Japanese characters, a higher resolution, say 300 dpi, is necessary. Fortunately, even old scanners can have enough resolution and thus they have been used for OCR.

Recently, digital camera also becomes an important device to acquire document images for OCR. Even a popular digital camera has a huge number of CCD elements. For example, my small digital camera (whose price was less than 200USD in 2011) can provide images with  $4,608 \times 3,456$  pixels. Clearly, my camera has a potential to capture an A4 document with 50 text lines with a sufficient resolution for OCR. In addition, camera-equipped mobile phones (and smartphones) become very popular. Accordingly, not only fans of photography but also many people carry cameras and have chances to acquire various kinds of text information through their cameras.

This is good news for OCR to discover its new targets. Since digital camera is far handier than scanner, we can consider various new OCR targets. Scanners generally deal with texts on paper sheet (while they have their own improvement [1]). In contrast, digital camera has no limitation on the target. In other words, digital camera can capture any texts around us. For example, digital camera captures big signboards in a town, digital texts on a computer display, precious historical documents, pages of a thick and heavy bound book, a timetable on a station, notes on a blackboard, slides projected on a screen, vehicle license plates in an omnidirectional image, etc.

Consequently, OCR for digital camera images (often called “camera-based OCR”) becomes one of important directions for researches on character recognition and document image analysis. In fact, as introduced in survey papers [2,3] and this chapter, there are many papers which tackle various problems around camera-based OCR. Moreover, camera-based OCR has been incorporated even in commercial services and thus becomes more popular.

### Applications

There are many applications of camera-based OCR. The first and most straightforward application is to recognize various characters and documents without putting them on any scanner.

- Car license plates are popular and important targets (e.g., [4]). The systems recognizing the plates are called automatic number plate recognition (ANPR) and have been used around early 1980s for police systems, traffic control, toll collection, and parking systems. In some scene browsing service, such as Google Street View, car license plates should be detected and hidden for privacy preservation [5].
- Several smart desktop systems have been proposed, where a camera will capture documents or texts on the desk and understand their location and contents (e.g., [6]).
- Business cards are popular targets of camera-based OCR on mobile phone.
- Thick books and precious historical books are also a target. In fact, book scanners capture page images by a single digital camera (or a stereo camera pair) (e.g., [7]). In [8], a complete digitization system based on a stereo camera-based book scanner has been proposed. See also dewarping methods listed in ▶Chap. 4 (Imaging Techniques in Document Analysis Processes).

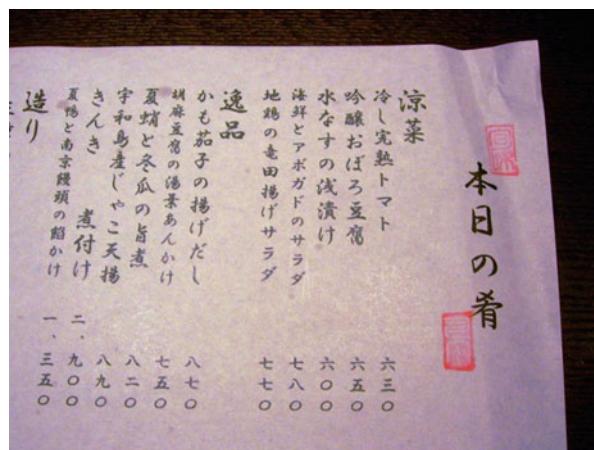
It will be easy to find other targets, such as presentation slides on screen, messages on digital signage, signboards, and posters and plates on wall. Even popular paper documents will be recognized by camera-based OCR because image acquisition by camera is handier than that by scanner.

Image search is also a good application of camera-based OCR. In this application, a personal or large public image database is searched for images including a keyword specified. For example, we can find scene images capturing the word “park” by using the keyword “park.” In 2011, several commercial services, such as Evernote, Google Goggle, and Microsoft OneNote, are already available for image search by camera-based OCR. An important thing is that these services can assign multiple recognition results to each text region for better retrieval accuracy (or better recall rate). Consequently, they may have many excessive and erroneous recognition results. Fortunately, those misrecognition results are not a serious problem for the services, because the misrecognition results are not fatal for their search results.

A similar but different image search application is camera-based document image retrieval. Document image retrieval is a technology of finding the similar or identical document image stored in a database, given a query document image. Its essential difference from the keyword-based image search is that it is not always necessary for document image retrieval to recognize individual characters and words. For example, it is possible to retrieve the identical document image by matching entire document images. Recently, cameras become popular for capturing document images (e.g., [9, 10]). For more discussion related to document retrieval, refer ▶Chaps. 7 (Page Similarity and Classification) and ▶24 (Image Based Retrieval and Keyword Spotting in Documents).

Another application is mobile dictionary or translator (e.g., [11–14]). Assume that now you are traveling around Japan and find a nice restaurant for your dinner. You want to have some special dishes and find the document of Fig. 25.1 on your table. With camera-based OCR and translator on your mobile phone, you will

**Fig. 25.1** A food menu written in Japanese (The photograph was taken by Gustavo Veríssimo and uploaded at Flickr Creative Commons)



understand that the right-most “column” of the document (Japanese texts are often written vertically) means “today’s menu,” even though you cannot understand those Japanese characters at all.

For navigation, scene texts are important clues. In our daily life, we extract text information from scene and then navigate ourselves. In fact, we will lose our way in an unfamiliar place without text. In a shop, we cannot find what we want without text. Sign recognition [15] will be useful for navigation. A similar trial was done to navigate blind persons (e.g., [16, 17]). Another navigation application has been done by cameras equipped on a mobile robot (e.g., [18]).

Handwritten characters are also target of camera-based OCR. For example, in [19], handwritten characters on whiteboard are to be recognized by a video camera system. In [20], an early attempt of a video camera-based system has been proposed, where handwriting patterns are captured by pen-tip tracking. As camera has smaller size and higher resolution, the combination of camera and pen will be examined continuously, like [21].

Technologies used in camera-based OCR are helpful to realize various paper-based user interfaces, as reviewed in [22]. This kind of user interfaces can be enriched by printing or embedding watermarks, small dots, and barcodes. Anoto digital pen technology utilizes small dots printed on paper for not only precisely determining pen-tip position but also identifying the paper itself. These dots are captured by a small camera equipped inside a pen device. Consequently, the Anoto pen can acquire its movement by using the image of dots from the camera. Without embedment, it is possible to utilize “paper fingerprint,” which is a microfiber structure of paper and also can be used for paper identification.

## Recognition of Texts in Video

### Background

A similar research topic to camera-based OCR is OCR for texts in *videos*. This research topic can be further divided into two types [23]. The first type is to recognize *scene texts*, which are the texts in the scene captured in video frames. This is almost the same as the above camera-based OCR except for the fact that we can utilize multiple consecutive frames to recognize a text. Specifically, we can find the multiple images of a word in 30 video frames, if the video camera (30 fps) captured the word for 1 s. We can utilize them for better performance of camera-based OCR.

The second type is to recognize *caption texts*, which are the texts superimposed on video frames. Since caption texts are attached intentionally to explain or summarize the contents of the video frames, they are also useful as an accurate index of the video. Caption texts are also captured in multiple video frames like scene texts and have their own characteristics. For example, the location, color, and size of caption texts are almost constant within a single video sequence. This constancy can simplify the recognition task. Some caption texts are scrolling from bottom to top with a constant speed regardless of background movement.

Caption texts are generally machine-printed characters superimposed on frame, and their poses and positions are often fixed within each video. Thus, caption texts are more constrained than scene texts and thus more tractable. There have been many trials on extraction and recognition of caption texts, such as [24–27]. A review is found in [23].

### Applications

Applications of recognition of texts in video are summarized as follows. Since the applications of scene text recognition are already described in section “[Applications](#),” the applications of caption text recognition are focused here.

The most typical application of caption text recognition is content-based video retrieval and annotation. Since the length of each video often exceeds several hours, manual inspection of large video database, such as broadcast video database, is almost impossible. Thus, automatic retrieval by text query is necessary and many trials have been made.

TRECVID [28] is a famous competition for improving content-based video retrieval and annotation and other video understanding technologies. Its active and long history from 2001 indicates the difficulty of video understanding. In fact, this task includes recognition of general visual objects, which is a well-known difficult problem. For example, if we want to retrieve videos capturing a coffee cup, we need to recognize cups in the videos. This is difficult because of the ambiguity of the object class “coffee cup.” Coffee cups have various appearances (by different

colors, shapes, materials, and poses). In addition, we cannot define the explicit and general difference between bowl and cup.

Captions (i.e., characters) are far less ambiguous than visual object and thus useful for content-based video retrieval and annotation. This is because any character has been developed and used for accurate human communication during thousands of years. In addition, captions are often strongly related to the contents of the video frame. For example, the caption of a news program will be the headline of the current news topic. Consequently, if captions are recognized, video sequences related a query keyword can be retrieved automatically and accurately.

There are other applications than video retrieval. In [24], caption texts are used for video skimming, which is a technique to extract key frames for video compaction. Zhang and Chang [29] have developed an event analysis method for baseball game videos based on caption text recognition. Bertini et al. [30] have tried to recognize not only caption text but also jersey number for identifying soccer players in video. In their system, face is also used as a cue for the identification and they showed those textual cues are helpful to improve reliability of the identification result. TV logo detection and tracking methods have been developed [31, 32] for detecting and skipping commercial block, removing the TV logo, and finding unauthorized distribution. Time stamp on photography, which is not a videotext but similar to a caption text, is also a target [33].

---

## Three Tasks and Their Difficulties

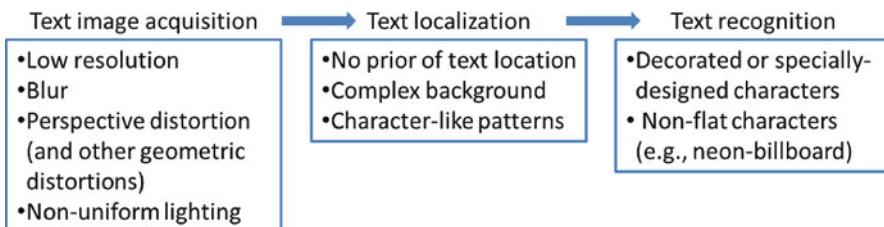
Generally speaking, recognition of texts in scene images and video is comprised of three tasks: *text image acquisition*, *text localization*, and *text recognition*. Figure 25.2 illustrates the three tasks. The text image acquisition task is image capturing and preprocessing for improving quality of the captured image. This preprocessing will make the succeeding text localization and recognition tasks easier. Rectification and deblurring are examples of the preprocessing. The text localization task is to find texts in the image. The text recognition task is to identify the class of each character in the text.

We can find the same three tasks in ordinary OCR; we scan the document and remove its skew, find text lines, and recognize characters and words on the text lines. However, texts in scene images acquired by camera are far more difficult to be localized and recognized than texts in ordinary document images acquired by scanner. Major difficulties of camera-based OCR are listed in Fig. 25.3 and they will be detailed in this section.

In [2], Jung et al. have defined that recognition of scene texts is comprised of five tasks: detection, localization, tracking, extraction and enhancement, and recognition. In their definition, the detection task is to find text regions and the localization task is to group the detected regions. The tracking task is to find the same text regions captured in multiple contiguous frames. The extraction and enhancement task is to convert the localized text regions into a binary image.



**Fig. 25.2** Three tasks of camera-based OCR



**Fig. 25.3** Major difficulties for camera-based OCR

Although the following sections assume the foregoing three tasks (acquisition, localization, and recognition), these five tasks are included in them.

## Text Image Acquisition

The first step of camera-based OCR is to acquire text images through various kinds of still cameras or video cameras. Many difficulties of camera-based OCR are caused by this acquisition step, that is, low resolution, motion blurring, perspective distortion, nonuniform lighting, specular by flash, and occlusion. Those difficulties are particular for the camera-based OCR. Accordingly, if we want to use an OCR engine of the traditional systems, we must apply image-processing techniques to camera-captured images for making them as much similar as scanner-captured images in advance. As noted above, rectification and deblurring are examples of the preprocessing. In the following section, those preprocessing techniques for camera-based OCR will be reviewed.

## Text Localization

The second step of camera-based OCR is text localization, which is the task to detect texts in a scene image or a video frame. If we capture an ordinary document by a

camera, the localization task is similar to text line extraction process for scanned documents. In contrast, if we capture a scene, the localization task becomes more complicated. In the following, the difficulties of text localization in scene images are discussed.

The first difficulty of scene text localization is that texts and words are scattered in the scene image. What is worse, we do not have any prior information of text location, text size, text orientation, and the number of texts. Since there is no formatting rule in scene texts, we cannot localize them by text segmentation methods for document images. Text size is not constant. For example, the text on a signboard close to the camera may be very larger than the text on another signboard far from the camera. The orientations of text lines are not constant. In addition, text lines, such as a label text on a bottle, are often curved.

The second difficulty of scene text localization is complex background. If texts are printed on a solid-colored background, the separation of the texts may be rather easy like ordinary scanned documents. However, in scene images, characters and their background often have a very low contrast and thus the separation is difficult even for the solid-colored background. If texts are printed on a textured background, or a background with color gradation, or a picture, the separation becomes a difficult problem. Captions are often superimposed on video frames directly; that is, they do not have any solid background. This case is one of the most difficult cases of text localization.

The third difficulty is that there are many patterns which seem like characters. In other words, there are many ambiguous and confusing shapes in scene. On the corner of a room, we will find “Y”-shaped edges. Around leaves of trees, we also find dense and fine (i.e., high-frequency) edge structures which look like dense text lines. Conversely, we know that some decorated characters seem like a branch of a tree. This difficulty invites a very important question: *what are character patterns?* More precisely, what is the difference between character patterns and non-character patterns? In fact, we never read the corner as “Y.”

The results of localization will be given as a set of rectangles (or, more generally, arbitrarily shaped connected regions), each of which contains a single character, or a single word. For single-character rectangles, neighboring character rectangles will be grouped (i.e., concatenated) to form a word. As noted above, texts in a scene image are often rotated or curved originally or by non-frontal camera. Accordingly, this grouping process is not trivial.

## Text Recognition

After the localization, the third task, that is, recognition of detected texts, is performed. This step is less severe than the above two tasks, if the target of camera-based OCR is ordinary business document printed on paper. This is because such a document is printed with regular fonts, such as Time-Roman, and thus we can use some traditional character/word recognition engine (if we can expect that the above two tasks are solved sufficiently).



**Fig. 25.4** Various characters in scene texts. (The photographs were taken by Steve Snodgrass and uploaded at Flickr Creative Commons.) The characters in the lower three images are three-dimensional characters

Text recognition, however, becomes very difficult when the target is not ordinary paper document. In fact, general scene texts often contain decorated or specially designed characters. Figure 25.4 shows several examples of those characters. Although some of them seem machine-printed characters, they have particular appearance and thus difficult to be recognized by traditional OCR engines. In captured scene images, we will also encounter more difficult characters to be recognized, such as calligraphic characters, pictorial characters, multicolored characters, textured characters, transparent characters, faded characters, touching characters, broken characters, and very large (or small) characters. In addition, we encounter handwritten characters on notebook and whiteboard and “handwritten fonts” on signboard and poster.

There are further variations in character appearance by the three-dimensional nature of scene. The lower three images of Fig. 25.4 include non-flat characters. Some of them are engraved characters and some are three-dimensionally formed characters (for a neon-sign). In addition, partial occlusion will damage the character appearance very severely.

## Methods to Solve Problems in Image Acquisition

In this section, the methods to solve various problems caused in image acquisition step are reviewed. Table 25.1 lists the problems discussed in this section. Note that several topics are related to scanner-based OCR and thus will not be described in this chapter. For example, the method to tackle with document images on non-flat surface will not be discussed here but in ▶Chap. 4 (Imaging Techniques in

**Table 25.1** Problems on image acquisition and their typical solutions

Problem	Solution	Short description	Remarks
Low resolution	Super-resolution by multiple-frame processing	Formulated as an optimal estimation problem of the original high-resolution image from multiple low-resolution inputs	Cons: Not applicable to still image
	Instance-based super-resolution	Reference to the instances each of which is a pair of a high-resolution image and its low-resolution version	Pros: Applicable to still image Cons: A sufficient number of instances
Blur	PSF estimation	Solving inverse problem with document/character-specific prior knowledge, such as existence of edge, binary color, and font style	
Perspective distortion	Boundary-based method	Parameter estimation using document boundary shape	Pros: Simple and easy implementation Cons: Only for texts on rectangular paper with visible boundary
	Text line-based method	Parameter estimation using the direction of regularly aligned text lines	Pros: Accurate in the direction of text lines Cons: Less accurate in the direction perpendicular to the text lines; not applicable to irregular text
	Character shape-based method	Parameter estimation using the direction of character strokes	Pros: Robust to irregular text Cons: Language-dependent
	Instance-based method	Parameter estimation referring stored instances	Pros: Applicable even to a single character Cons: Large memory and computation
Nonuniform lighting	Local binarization methods	Nonuniform background elimination by using a threshold value optimized locally	See ▶Chap. 4 (Imaging Techniques in Document Analysis Processes) for various binarization methods
	Image composition method	To remove a strong specular reflection, two (or more) images with different lighting conditions are composed into one image	

Document Analysis Processes). Various binarization methods for document images are also described in the same chapter.

## Low Resolution

The image processing to convert a low-resolution image or a set of low-resolution images into a more visible image is a kind of image enhancement processes. This is called *super-resolution* because this image processing often provides higher-resolution images for better visibility. There are two approaches. The first approach is multiple-frame processing and the second is the instance-based approach. The first approach is applicable for videotexts and the second is applicable even for scene texts, i.e., texts in a single still image.

### Multiple-Frame Processing

The first approach based on multiple-frame processing utilizes the fact that the same text appears in multiple frames in a video. Simply speaking, low-resolution images of a text in multiple frames are redundant but erroneous representation of the original text. Thus, by applying some error correction technique (like error-correcting code), the original high-resolution image can be recovered.

A simplest version of super-resolution is simple averaging of multiple frames [34, 35]. Since caption texts are often fixed at a certain position, the averaging operation will enhance the caption text part and cancel its background clutter. If a caption text moves by, for example, scroll up/down, the text should be tracked precisely before averaging. If a text undergoes other deformations, such as affine deformation, the deformation should be estimated and removed before averaging. Mancas-Thillou and Mirmehdi [36] have improved this averaging approach by adding a high-frequency component. This high-frequency component is extracted from the low-resolution images by the Teager filter, which is a kind of high-pass filters.

Capel and Zisserman [37] have examined performance of a maximum likelihood (ML)-based approach, where observed low-resolution images are assumed via a Gaussian blurring process and the original image which most likely produces those low-resolution images are derived. In their paper, starting from this simple ML-based approach, its iterative version, called back projection [38], is also examined. They also proposed two maximum a posteriori (MAP)-based approaches by introducing a prior into the ML approach. One incorporates a prior to evaluate the total image gradient, and the other incorporates a prior to evaluate the total variation.

Donaldson and Myers [39] have proposed an improved version of [37] by introducing a new prior to the MAP framework. This prior is based on the bimodal (i.e., black and white) property of document images. They also introduced a smoothness term not to lose the step discontinuity around the boundary of characters.

Like the bimodal prior of [39], it is possible to introduce other priors suitable for MAP-based document image super-resolution. Banerjee and Jawahar [40]

have proposed a MAP-based super-resolution method where the bimodal prior is employed along with an edge-preserving smoothness term. Bayarsaikhan et al. [41] have proposed a prior evaluating the total variance. Their prior is different from the simple total variance prior on the point of using a weight to the local direction of edges for avoiding over-smoothing around character edge.

An important point for multiple-frame processing methods is that they require precise geometrical alignment of multiple frames in advance to averaging procedure or MAP estimation. Consequently, some tracking method in sub-pixel accuracy is often employed for caption texts. More generally, some nonlinear image registration is necessary for two or more consecutive frames. For example, RANSAC is utilized in [36]. In [42, 43], document mosaicing methods are proposed for perspective registration of multiple images.

### **Instance-Based Approach**

The instance-based approach does not require multiple video frames as its input. Instead, this approach prepares many instances in advance. Each instance is a pair of an original high-resolution image and its low-resolution version. This instance is an example representing how a high-resolution image becomes in its low-resolution version. Conversely, the instance also represents how a low-resolution image becomes in its high-resolution version. Consequently, this instance provides a well-grounded evidence for super-resolution.

An important point of this approach is that a sufficient number of instances are necessary for better super-resolution results. Clearly from the above principle, if we cannot find a low-resolution image instance close to the input low-resolution image, it is impossible to guess the original high-resolution image of the input image. In other words, if we want to apply the method for a specific class of documents, it is better to collect a sufficient number of document images of the class. Another important point is that this approach is free from the problem of tuning priors, such as smoothness.

Baker and Kanade [44] have proposed a super-resolution method based on this approach. In their method, called *hallucination*, high-resolution image instances are firstly collected into a database. Then, each of them is represented in multiple resolutions and stored in the database together. When a super-resolution image of an input low-resolution image is necessary, the database is searched for the nearest neighbor of each local part of the input image. Since the image of the nearest-neighbor part in its original high resolution is known, it is possible to reconstruct a super-resolution image. Although this method is mainly applied for face images, several results for document images are also shown in [44]. Dalley et al. [45] also have developed a similar method independently based on their past trial [46] on instance-based super-resolution for general images.

Like the multiple-frame approach, the instance-based approach also can be considered to be based on a prior model. In fact, in [44], the instance is considered as a “recognition-based prior.” In this sense, the instance-based approach also can incorporate other priors, such as a bimodal prior. Park et al. [47] have proposed a

hybrid method where the recognition-based prior by instances is combined with a bimodal prior in a Bayesian framework.

A more direct approach for dealing with low-resolution text images has been developed by Jacobs et al. [48]. In their method, low-resolution text images are directly processed by a word recognizer which is trained by low-resolution images. Specifically, a low-resolution word image is first decomposed into fixed-size sub-images by a sliding window and each sub-image is then processed by a character recognizer based on a convolutional neural network. The character recognizer is trained by a sufficient number of camera-captured, i.e., degraded character, examples. The final word recognition result is obtained by a dynamic programming search, like general analytical word recognition methods.

## Blurring

Camera-captured document images are often blurred. Since no blur has happened in scanned document image, blurry image is a difficult target for conventional OCR. Several trials have been done for blur removal, or deblurring. Note that deblurring is a kind of image enhancement process and thus related to the other image enhancement process, i.e., super-resolution.

There are two reasons of blurring: imperfect focus and motion. For both cases, a blurred image is considered as the result of convolution between an original image and point spread function (PSF). PSF represents how a single pixel in the original image is spread over in the blurred image.

Consequently, for deblurring, we need to estimate two signals, that is, PSF and the original image, simultaneously from a single input image. This simultaneous estimation problem is so-called *blind deconvolution*, which is a well-known ill-posed inverse problem. A very intuitive explanation of this problem is as follows: a number “24” is given and it is known that this number is the result of multiplication of two numbers. What are those two numbers? Of course, there is no unique answer. They may be 1 and 24, or 4 and 6, or -4 and -6, etc. So, we need some additional information, i.e., a prior, to choose the best answer from those three candidates.

In the above naïve example, if we know one of those two numbers, the other number is immediately determined. This fact suggests that if we can estimate the PSF in any way, the original image is directly recovered through the inverse filtering based on the estimated PSF. Unfortunately, for the deblurring problem, even if the PSF is given, the determination of the original image is still not straightforward. For example, in [49], a regularization-based method is proposed for determining the original image with a given PSF.

A key idea for deblurring camera-captured document images is to utilize specific properties of document images as prior knowledge for regularizing the PSF estimation problem. For example, Tian and Ming [50] have proposed a method to estimate two-dimensional PSF by observing character edges. Specifically, the intensity changes across an edge in horizontal and vertical directions are directly considered

as the horizontal and vertical projections of PSF, respectively. The second idea is that PSF is estimated locally. This is because the degree of blur depends on the depth (i.e., distance from the lens) of target scene and thus multiple objects located in different depths have different blurs. In [50], the entire image is simply divided into several sub-images and for each sub-image its local PSF is estimated in the above way.

In addition to the existence of edges, other kinds of prior knowledge on the target document image can further improve the estimation condition. Kanungo and Zheng [51] have proved that the knowledge of the font type of the target document is helpful to estimate the parameters of document image degradations, such as blurring. Cheng et al. [52] used the “bimodal” prior which indicates document images are generally binary images. In [49], a traditional smoothness prior has also introduced for deblurring with special consideration to suppress its side effect around character edges.

Different from isotropic PSF of imperfect focus, PSF of motion blur has a direction which reflects motion direction. Accordingly, in order to remove motion blur, it is necessary to estimate not only blur size but also blur direction [53]. In [53], the motion blur is estimated in the two-dimensional Fourier spectrum domain because the motion blur diminishes the high-frequency components of the motion direction.

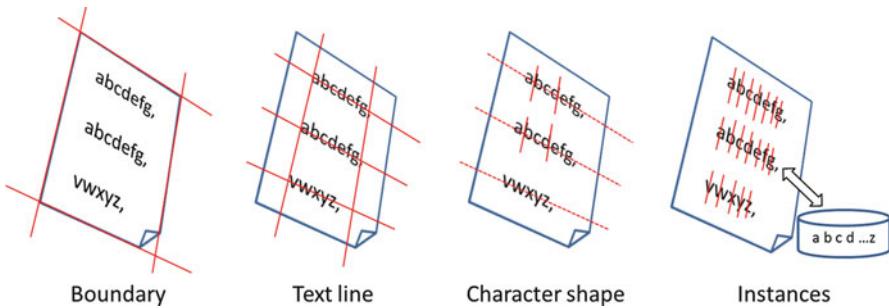
## Perspective Distortion

When document image is captured by camera, the target image often undergoes *perspective distortion* due to non-frontal camera. Since perspective distortion will degrade performance of text line extraction, word/character segmentation, and character recognition, it should be removed. For image capturing by scanner, the target image undergoes just skew distortion, i.e., rotation distortion. Skew is represented by a single parameter (rotation angle) and thus rather easier to be estimated and removed. In contrast, perspective distortion is represented by more parameters because camera posture (position and direction) relative to the target document has much more freedom than scanner. Consequently, the estimation and removal of perspective distortion, or rectification, becomes more difficult.

Figure 25.5 illustrates four typical approaches to estimate perspective distortion. Each of these approaches is detailed in the following. Note that skew removal (called deskew) is described in ▶Chap. 4 (Imaging Techniques in Document Analysis Processes). Removal of nonlinear distortion of curled documents and crumpled documents is called dewarping and is also described in that chapter.

### Estimation of Perspective Distortion by Document Boundary

The simplest approach to estimate perspective distortion is to utilize the four boundary edges of document sheet. Clark and Mirmehdi [54] have proposed a method on this approach. In their method, line segments are found by Hough transform and



**Fig. 25.5** Four approaches for estimating perspective distortion

then four line segments forming a quadrilateral are selected as a candidate of a document sheet region. If the inside of the candidate quadrilateral is evaluated to contain characters, it is determined as a document region. The evaluation is done by checking the sum of edge vectors. The shape of the quadrilateral shows the perspective distortion and the distortion can be removed. A boundary-based removal of affine transformation is proposed in [15].

### Estimation of Perspective Distortion by Text Lines

When the document boundary is difficult to be found, the direction of linear text lines can be utilized to estimate perspective distortion. There are two types of the methods which are utilizing text lines for estimating perspective distortion. In the first type, text lines are not extracted explicitly. Instead, a projection profile along a certain direction is derived to evaluate how the direction is valid as the direction of text lines. In the second type, text lines are explicitly extracted by some line tracking method.

The methods of the first type are as follows. Clark and Mirmehd [55] have utilized this fact to determine two vanishing points. Specifically, the horizontal vanishing point is determined by using projection histogram. For each candidate point, a projection profile is created by counting the number of black pixels on each of lines radially emerged from the point. The point where the large peaks are found is selected as the optimal horizontal vanishing point. (More precisely, the point with the maximum squared sum of histogram values is selected.) The vertical vanishing point is then determined by using the optimal horizontal vanishing point. The task of determining the vertical vanishing point is more difficult because no line structure is expected in the vertical direction. In [55], two end points (the left end and the right end) and the center point are detected along each of the above lines radially emerged from the horizontal vanishing point and their arrangement is used for the task. Dance [56] has proposed a similar method of estimating two vanishing points.

The methods of the second type are as follows. Myers et al. [57] have proposed a method based on text line extraction. They extracted each text line by a simple

blob-linking process. Then the bounding box of each individual text line is determined and rectified. Since the characters in the resulting image still undergo slant deformation, it is estimated by observing vertical projection profile. An important point is that those processes are performed on each text line independently. In other words, this method does not assume that the target image contains several text lines. Consequently, this method can be used even for isolated short text line images, such as a signboard showing a restaurant name.

A similar text line extraction process is utilized in [7]. Different from [57], this method tries to utilize directions of multiple text lines for estimating perspective distortion. Specifically, RANSAC is used to estimate the optimal horizontal component of perspective distortion for the entire document. Then the remaining vertical component is estimated by using the arrangement of ending points of the text lines like [55].

In Yamaguchi et al. [58], the perspective distortion of each individual character is estimated. Their method deals with a single text line of digits. After extracting digit candidate regions by using several heuristics, global skew is first estimated by Hough transform. Then the slant of each candidate is estimated by fitting a tilted rectangle. This two-step approach is similar to [57].

### Estimation of Perspective Distortion by Character Shape

In addition to document boundary and text line, character shape has also been utilized for estimating perspective distortion. Characters of many languages have vertical and horizontal strokes. For example, some Latin characters, such as “E,” “I,” “d,” and “h,” have a vertical stroke and some characters, such as “H,” “T,” “e,” and “t,” have a horizontal stroke. Among those strokes, the vertical strokes are helpful to estimate the vertical vanishing point. In fact, the estimation of the vertical vanishing point is generally more difficult than that of the horizontal one. As we see the above, the horizontal vanishing point can be estimated by using the directions of text lines but the vertical one cannot be estimated by them. Consequently, the above methods have used the arrangement of the end points or the center points of the text lines, although they are often unstable.

In Lu et al. [59], stroke boundary and top and bottom tip points are first detected at each character. Then, vertical strokes are detected from the stroke boundary. The vertical strokes are used for estimating the vertical component of perspective distortion and the tip points are used for the horizontal component. Liang et al. [60] also have developed a similar method where vertical strokes are used for the vertical component. The difference from [59] is that their method estimates horizontal text line direction in a local manner and thus is applicable for non-flat surface deformation.

The estimation of perspective distortion using character shape generally requires more computations than the document boundary-based methods and the text line-based methods. Yin et al. [61] have proposed a multistage method where first a boundary-based estimation is performed. If its result is not reliable, a text line-based

method is applied. The reliability of this result is also evaluated, and if it is not reliable, a character shape-based method is finally applied.

### Estimation of Perspective Distortion by Instances

The above methods generally assume the linearity (i.e., one dimensionality) of text line as an important clue for estimating perspective distortion. Camera-captured texts, however, sometimes do not have clear linearity. In a case, characters are laid out freely. In another case, texts are captured only partially and thus not enough to have a linear structure.

Uchida et al. [62] have proposed an instance-based approach for estimating text skew (i.e., rotation), which is considered as a constrained case of perspective distortion. In their method, the value of a skew variant, such as the size of the upright bounding box, is calculated for each connected component. The skew of the connected component is determined by referring the instance which describes the relationship between the skew variant and the skew angle. For example, the character “I” has a small bounding box size without skew and has a larger size with a skew and thus we can guess the skew by the size. Since at the skew estimation step, the character class (such as “I”) is not known, they used skew invariants to refer a proper instance. This method is extended as a part-based method [63] where local skew is estimated at each local part detected by a keypoint detector, such as SIFT and SURF.

The skew estimation method by Lu and Tan [64] is a more radical and promising instance-based approach. From each connected component, vertical component cut is first calculated. The vertical component cut counts how many times the center vertical line crosses strokes and thus will change according to skew. The normalized histogram of the vertical component cuts for all connected components is considered as a feature vector of the document. The nearest neighbor is searched for the database containing the feature vectors of various skewed documents and the skew angle of the nearest neighbor is determined as the estimation result. An interesting point is that this method detects languages simultaneously by preparing enough document instances printed in those languages.

### Nonuniform Lighting

Since camera-based OCR cannot expect a controlled uniform lighting environment, it should deal with nonuniform lighting, such as shade by a non-frontal light and specular reflection by a flashlight. Many camera-based OCR researches have tried to remove shade in their binarization process. The binarization process generally employs some local thresholding technique, which is detailed in ▶Chap. 4 (Imaging Techniques in Document Analysis Processes). The effect of specular reflection can be reduced by the local thresholding technique; however, under strong reflection, it is impossible to restore the text image around the reflecting area. In [7], a dual-flash system is introduced, where two pictures are taken under different flashlights

and then composed as a single picture without reflecting area. Koo et al. [65] have proposed a similar image composition method.

---

## Methods to Solve Problems in Text Localization

As noted in section “[Text Localization](#),” text localization is a very difficult task due to three main reasons:

1. No prior information of text location, size, orientation, etc.
2. Complex background
3. Character-like non-characters

For dealing with this difficult task, many researches have been conducted. In fact, text localization is, currently, the hottest topic of camera-based OCR and fascinating not only OCR specialists but also computer vision and machine-learning researchers. A huge variety of approaches are developed and thus difficult to cover all of them. A couple of survey papers are found as [2, 23].

It is noteworthy that competitions on text localization, called “ICDAR Robust Reading Competitions,” have been organized [66–68] and played an important role to make this research topic very active. The result of the latest competition in 2011 [68] reported that the best method have achieved detection accuracy of around 70 % in f-value on a dataset of scene images including texts. This means that we now (2012) can extract 70 % true text areas with 30 % erroneous non-text areas.

## Typical Process of Text Localization

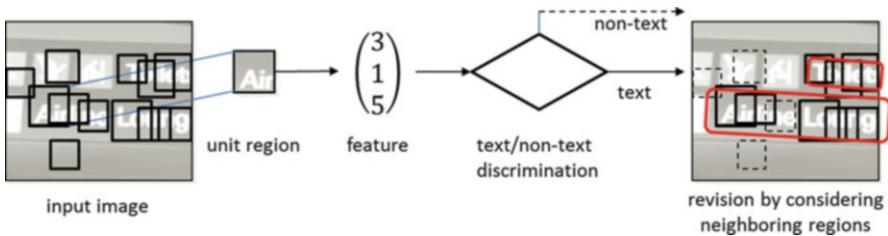
Figure 25.6 illustrates a typical process of text localization. Its first step is feature extraction from a region. Then, the region is classified as a text region or non-text region. As an optional step, this classification result is revised by considering the classification results of its neighboring regions. If this revision tries to form a cluster of text regions, they are combined as a larger text region, such as a word region.

Of course, there may be text localization processes which do not follow the process of Fig. 25.6 exactly. For example, we can consider a process where the first text/non-text discrimination step is performed at neighboring regions simultaneously. In addition we can consider a process where different features are extracted from different regions and then combined for discrimination.

## Four Aspects Characterizing Text Localization Methods

As noted above, there are a huge number of text localization methods and thus it is impossible to detail individual methods. This section, therefore, tries to classify them by four aspects according to Fig. 25.6:

- Feature
- Method of text/non-text discrimination



**Fig. 25.6** Typical process of text localization

- Region type
- Method of revision by considering neighboring regions

Each localization method is characterized by their choice on these aspects. For example, a method will extract a color feature (“feature”) from a fix-sized block (“region type”), then perform an SVM classification (“method of text/non-text discrimination”), and finally revise the discrimination result by an MRF (“method of revision by considering neighboring regions”).

## Features for Text Localization

Table 25.2 lists typical features for text localization. Those features have been used because they are considered to be good for text/non-text discrimination. In other words, they are expected to capture some “character-ness.” For example, a contrast feature is expected to have a large value around text regions because the characters and their background often have a large difference in their intensity values and/or colors of characters. Note that most features, especially low-level features such as color, contrast, and gradient, have been designed by researchers’ intuitive expectation. On the other hand, we can use a feature selection method, which is a kind of machine-learning method, for selecting good features. For example, we can use AdaBoost or Random Forest to select good features from a bank of features.

Among the listed features, context features have been used by a different purpose. Context features try not to evaluate “character-ness” directly but “contextual suitability for text existence.” In fact, we have many empirical priors for text existence. In the sky, there may be no character. On leaves, there may also be no character. On a planar area, a text, i.e., a sequence of characters, can exist. Gandhi et al. [102] have tried to detect texts by finding planar areas using a shape-from-motion technique. Park et al. [103] have tried to estimate the position of caption texts by the existence of a speaker. Kunishige et al. [104] have utilized environment recognition results (where possible classes of the environment were ground, sky, green, building, etc.) for text/non-text recognition.

Note that we can consider that the recognized texts as context information for recognizing other objects in the scene. In other words, a text in a scene will be useful to recognize its surrounding object and the scene itself. A clear example is that a word “cola” on a cylindrical object indicates that the object is a can or

**Table 25.2** Features for text localization

Features	Short description	Remarks	Example
Color/intensity	A text region is expected to have a higher contrast level or a clear bimodality in color histogram	Shivakumara et al. [69] recommended using different discrimination criteria according to the contrast level	Ezaki et al. [71] (Bimodality by Fisher's Discriminant Rate)
Edge	A text region expected to have more edges than non-text region		Kuwano et al. [72] (Edge pair on scanline), Sin et al. [73] (Spectrum of edge image), Liu and Samarabandu [74] (Multiscale edge)
Corner	A text region is expected to have corner points more densely than non-text regions	Corners are detected as <i>keypoints</i> and utilized in visual object recognition	Bertini et al. [75] (Harris corner), Huang and Ma [76] (Dense corner point area), Zhao et al. [77] (Harris corner)
One-dimensional gradient	A scanline across a text region is expected to have a specific gradient pattern	Need of combination of results on adjacent scanlines for two-dimensional regions	Wolf et al. [78], Wong and Chen [79], Kim and Kim [80] (Precise gradient around character edge), Phan et al. [81] (Maximum gradient difference)
Two-dimensional gradient	A text region is expected to have a specific gradient pattern	So-called local features such as SIFT and SURF, evaluate the 2D gradient of a region around a keypoint. HOG (histogram of oriented gradients) also evaluates 2D gradient	Wang and Belongie [82] (HOG), Uchida et al. [83] (SURF), Mishra et al. [84] (HOG)
Local texture	A text region is expected to have a specific local texture	Pros: For example, local binary pattern (LBP) is a texture features invariant to intensity change	Anthimopoulos et al. [85] (LBP)
DCT coefficients	Spectrum features by DCT. As a kind of texture, a text region is expected to have a specific frequency band	Pros: JPEG and MPEG also employ DCT	Chaddha et al. [86], Zhong et al. [87], Goto [88]
Wavelet/Gabor		Pros: Multiresolution is suitable to scale-invariant character detection	Jain and Bhattacharjee [89] (Gabor) Haritaoglu [13] (Edgeness by DWT), Saoi et al. [90], Kumar et al. [91] (Matched DWT), Weinman et al. [92] (Gabor)

(continued)

**Table 25.2** (continued)

Features	Short description	Remarks	Example
Hybrid	Combination of the above (rather simple) features	Pros: Robustness by complementary feature usage Note: Many features are introduced in [70]	Kim et al. [93], Chen et al. [94] (Edge + color), Pan et al. [95] (HOG + LBP), Peng et al. [96] (Edge + corner + Gabor), Yi et al. [17] (Gradient + edge)
Linearity of CC alignment	CCs (or other elements) aligned in a line are considered as characters	This feature is generally considered in the postprocessing step to combine neighboring regions	See Table 25.5
Character elements	Representation of the target region by a set of typical character elements, or substrokes	Machine learning methods are utilized to prepare the typical character elements	Pan et al. [97], Coates et al. [98], Yi and Tian [99]
Visual saliency	A scene text region should have enough visual saliency to attract potential readers	Since visual saliency is defined by combining low-level image features, such as color contrast and edge orientation, it can be considered as a hybrid feature	Shahab et al. [100], Shahab et al. [101]
Context	Context, i.e., the area surrounding a target region, will be a strong prior. For example, in a region surrounded by sky, it has a small probability to be a text region	Pros: Good for reducing false detection Cons: A further effort is necessary to understand the context	Gandhi et al. [102], Park et al. [103], Kunishige et al. [104]
Language-dependent feature	Feature unique to the target language are utilized as detection clue		Bhattacharya et al. [105] (Horizontal line of Indian script)
User interaction	User's action is utilized as a strong clue of indicating a text region		Kim et al. [106] (Focus indicated by user is used as the region for first color clustering)

glass. A word “menu” indicates restaurant scenery. In [30], the numbers detected and recognized in a soccer game video are used to recognize player’s face.

User interaction is also a promising approach to ease the difficult text localization problem. Nowadays, we have various pointing interfaces, such as touch pads, and thus it is easy to tell the location of texts to machine. If the machine knows the

rough text position by user’s interaction, it can extract the corresponding text region precisely by using, for example, *grabcut* [107]. Focusing operation on a digital camera is also a promising user interaction which can be used as a prior for character detection [106].

## Methods of Text/Non-text Discrimination

Table 25.3 lists typical methods of text/non-text discrimination. If the discrimination result by the feature extracted at a specific region becomes “text,” the region is considered to be a text region; that is, a text region is detected. The possible features have already been overviewed in Table 25.2.

### Discrimination by Machine Learning

Nowadays, the most popular discrimination strategy is to employ a machine learning method for training a text/non-text classifier. Multilayer perceptron (MLP) and Support Vector Machine (SVM) are traditional choices. Several methods, called classifier ensemble methods, integrate results by multiple classifiers to have a final result. The classifier ensemble methods, such as AdaBoost and Random Forest, can provide a reliable result by their weighted-voting strategy. Note that AdaBoost and Random Forest have a promising function of feature selection. They automatically select discriminative features and thus can relieve the problem of “the curse of dimensionality.”

Machine learning methods are useful because they do not need “manual optimizations” of classification rules; however, they need a sufficient number of training samples. In fact, the performance of the trained classifier heavily depends on the quantity and quality of training samples.

- For the quantity, we should not forget “the curse of dimensionality” and “overfitting”; if we use a higher dimensional feature and/or a flexible classifier with a larger number of parameters, we need to prepare (exponentially) more training samples. Although this problem is relieved by feature selection as noted above, feature selection also needs an enough number of training samples.
- For the quality of training samples, we should be careful of large variations of not only 1 character shapes but also non-character shapes. Especially, it is almost impossible to cover all non-character shapes by a limited number of training samples. One possible remedy on this problem is to use a 1-class classifier, where a classifier is trained by using only positive samples (i.e., samples from text regions).

### Recognition-Based Localization

Recognition-based localization utilizes the discrimination power of OCR for text localization. This approach is similar to so-called recognition by segmentation (or recognition-based segmentation) for text recognition. The conventional recognition-by-segmentation techniques are applicable to the one-dimensional segmentation problem, where an input text image is partitioned into individual characters from left to right. In contrast, the text localization problem of

**Table 25.3** Methods of text/non-text discrimination

Discrimination approaches	Short description	Remarks	Example
Heuristics	Discrimination by a set of if-then rules, each of which is manually designed	Pros: Intuitive Cons: Weak theoretical validity; verification is often necessary	Messelodi and Modena [108], Chen et al. [25], Gatos et al. [109], Ezaki et al. [16]
Machine learning	Text/non-text discrimination using a classifier by some machine learning technique	Pros: Automatic and robust Cons: A sufficient number of training samples are necessary. For text/non-text discrimination, it is often difficult not only to balance the number of training samples, but also to cover all possible variations	Chen et al. [25] (MLP + SVM for verification), Chen and Yuille [110] (AdaBoost), Hu and Chen [111] (SVM), Pan et al. [95] (AdaBoost), Xu et al. [112] (AdaBoost), Hanif and Prevost [113] (AdaBoost), Peng et al. [96] (SVM + CRFwGC), Kunishige et al. [104] (Random Forest), Ma et al. [70] (Random Forest)
Recognition-based localization	OCR as a verifier	If the target region is recognized as a character or a text, the region is determined as a text region	Ohya et al. [114]
Exhaustive application of OCR	While sliding the location of a window, character recognition is performed at individual locations. If the window can provide any recognition result, the window is determined as a character	Pros: Accurate discrimination by the power of OCR Cons: Sensitive to the ability of OCR	Kusachi et al. [115]
Aggregation	After generating multiple (and tentative) recognition results around the target region, their coherence is checked	Pros: Reasonable realization of localization-by-recognition Cons: Prohibitive computations; a large number of training samples	Li et al. [129], Rong et al. [130]
Multiframe analysis	A temporally stable region is determined as a caption text	For moving captions, text tracking [31, 32] is often employed	Bertini et al. [75] (Temporal stability of Harris corner), Antani et al. [116] (Temporal stability of extracted CCs)

scene images becomes a two-dimensional segmentation problem and thus the conventional techniques cannot be applicable.

Two possible styles of recognition-based localization are as follows:

- OCR as a verifier: A pre-detection operation of a text region is performed and then the region is verified by OCR (e.g., [114]). If OCR provides any valid character string, the region is verified as a text region.
- Exhaustive application of OCR: With a sliding window technique, OCR is performed at every window (e.g., [115]).

Although recognition-based localization is promising, we have to recall that scene characters are often different from document characters and difficult to be recognized. Researches for recognizing various (often decorated) scene characters are now active, but their recognition accuracy is less than OCR for ordinary documents. Consequently, there is still an inevitable risk of rejecting text regions by false rejection by OCR.

## Region Types for Text Localization

For extracting a feature for text/non-text discrimination, we have to define the region where the feature is extracted. This is not a trivial problem because the size of the target character is unknown in advance to text localization.

Table 25.4 lists possible region types. One reasonable choice to avoid the above “unknown size” problem is to use connected components after a binarization process. If a character is printed on a uniform background with a sufficient contrast value, the character will be extracted as a connected component by the binarization process. Color clustering and stroke width transform (SWT) [118] are also possible choices for extracting the appropriate connected component.

However, the extraction of the appropriate connected component is also difficult due to various factors. Figure 25.7 shows binarization results under different global threshold values. It is possible to observe that the shape of each connected component depends on the threshold. Especially, the shape of “M” changes drastically because a shadow is casted on it. Recently, MSER [119] has often been chosen as a more stable method for connected component analysis; however, even MSER may not be perfect on, for example, the case of this “M.”

Extraction of multiple connected components under different parameters and their combinatorial use is a possible remedy. This will be followed by some recognition-based localization approach; the individual connected components are recognized and their recognition results are aggregated to have the final result.

## Method of Revision by Considering Neighboring Regions

Although the above three aspects (features, discrimination, and region type) can characterize a text localization method, that is, can determine localization results, a revision process is often employed as an optional process. Revision of the

**Table 25.4** Region types for text localization

Region type	Short description	Remarks	Example
Single pixel, keypoint	Feature extraction and discrimination at individual pixels	Pros: Precise evaluation Cons: Large computation; need of gap filling, less stability	Uchida et al. [83], Shahab et al. [101]
Superpixel	A group of adjacent and coherent pixels [117]		Cho et al. [120]
Line	1-D gradient-based discrimination methods extract gradient features on a line	Need of combination of results on adjacent scanlines for two-dimensional regions	Wolf et al. [78], Wong and Chen [79], Kim and Kim [80], Phan et al. [81]
Block	Generally, a fix-sized square region	Pros: Simple Cons: Rough. A single character may be divided into multiple blocks and thus postprocessing is often necessary like [84]	Hu and Chen [111], Hanif and Prevost [113], Mishra et al. [84], DCT-based methods
Connected component	Binarization	CC containing pixels with similar intensity values. For binarization methods, see ▶Chap. 4 (Imaging Techniques in Document Analysis Processes)	Gatos et al. [109] (Binarization result) Wang and Kangas [122], Mancas-Thillou and Gosselin [123]
Color clustering	CC containing pixels with similar colors	Pros: Simple; many clustering methods Cons: Weak against multicolored characters	Wang et al. [121], Ezaki et al. [16], Wang and Kangas [122], Mancas-Thillou and Gosselin [123]
Stroke filter	For example, stroke width transform (SWT) [118] enhances the region surrounded by a pair of parallel edges	Pros: Accurate extraction of stroke-shaped area	Kuwano et al. [72], Liu et al. [124], Epshtain et al. [118], Ma et al. [70](SWT)
Maximally stable extremal region (MSER)	MSER [119] is a region with sufficient difference from its surroundings	Pros: Accurate extraction of high-contrast area	Donoser et al. [125], Neumann et al. [126], Merino-Gracia et al. [127], Niemann and Matas [128]
Multiple CCs	Multiple CCs are extracted by different criteria and therefore overlapped	Pros: Robust to variations Cons: Aggregation step is necessary	Li et al. [129], Chen et al. [25], Rong et al. [130]



**Fig. 25.7** Binarization results under different (global) threshold values. It is possible to observe that the shape of connected components varies by the threshold. *Green* annotations are OCR results of connected components

discrimination result is necessary because the text/non-text discrimination is done on individual regions independently and thus unstable. On the other hand, a scene image usually contains one or more words and sentences and each of them is linearly aligned characters. Hence, the neighboring regions are often mutually dependent and this dependence can be used for the revision. For example, if three neighboring regions are discriminated independently as text, non-text, and text, the middle region may be a text region.

Table 25.5 lists the revision methods by considering neighboring regions. The most typical revision method is the optimization-based revision, where a certain graphical model, such as MRF, is used. In MRF, each region is treated as a node and nodes of neighboring regions are connected by an edge. Each node has some cost, such as a discrimination cost, and each edge also has some cost, such as coherence of the neighboring nodes. The latent variable at each node takes a text or non-text label. The value of the latent variable is optimized in various strategies, such as dynamic programming, belief propagation, and graph cut.

---

## Methods to Solve Problems in Text Recognition

### Scene Character Recognition Using Standard OCR Techniques

Since characters in images and video have wider variations in their appearance than those in scanned documents, their recognition module should be robust enough to deal with the variations. As listed in Table 25.6, there are two major approaches to make standard OCR techniques robust.

**Table 25.5** Methods of revision by considering neighboring regions

Revision methods	Short description	Remarks	Example	
Forced connection	Connect all neighboring text-candidate regions	Need of further text/non-text discrimination on the connected result	Hanif and Prevost [113] (MLP was used for the final discrimination)	
Heuristics	Connect neighboring candidate regions if they satisfy handmade rules	Pros: Simple Cons: Manual parameter setting is necessary	Goto and Aso [131], Wang and Kangas [122], Epshteyn et al. [118]	
Morphology	Neighboring text-candidate regions are connected by a morphological operation	Pros: Simple Cons: No function to exclude non-character regions	Huang and Ma [75] (Filling space among corner points)	
Optimization	Graph-based representation and optimization; Markov random field (MRF), Conditional random field (CRF)	After creating a graph connecting text-candidate regions, their final and optimal text/non-text classification is performed under a criterion	Pros: Erroneous candidates can be excluded by considering neighbor regions Cons: Evaluation functions (e.g., edge cost) should be designed on the graph Note: Many evaluation functions are introduced in [70] Zhang and Chang [132] (Higher-order MRF), Kumar et al. [91] (MRF), Pan et al. [95] (MRF), Xu et al. [112] (MRF), Wang and Belongie [82], Cho et al. [120] (CRF), Peng et al. [96] (CRF)	Zhang and Chang [132] (Higher-order MRF), Kumar et al. [91] (MRF), Pan et al. [95] (MRF), Xu et al. [112] (MRF), Wang and Belongie [82], Cho et al. [120] (CRF), Peng et al. [96] (CRF)
	Stochastic relaxation		Hase et al. [133]	
Word lexicon	Neighboring regions are combined if they form a word		Mishra et al. [84]	

- The first approach is to use sufficient preprocessing methods for normalizing the target text region like a scanned document before applying the standard OCR. Any preprocessing method listed in Table 25.1 can be used for this purpose. For example, if we want to recognize low-resolution characters, we first apply a certain super-resolution technique to the input image and then the standard OCR.
- The second approach is to train the character recognizer by using a set of training samples covering the variations of scenery characters. For example, if we want to recognize low-resolution characters, we will train our recognizer with low-resolution training samples. Note that training samples can be generated

**Table 25.6** Methods for recognizing texts in images and video

Recognition approach	Short description	Remark	Example
Use of standard OCR techniques	Direct use of standard OCR after sufficient preprocessing steps	By normalizing the input image by a set of preprocessing steps, some standard OCR technique is applied	In [24, 84], word lexicon is employed like a traditional analytical word recognizer. Similarly, other recognition methods are found as the “recognition-based localization” entries of Table 25.3, as well as Table 25.5
Training OCR by scene character samples	Scene characters are directly used for training a standard OCR technique		Sawaki et al. [141], Jacobs et al. [48], Saidane and Garcia [142], Weinman et al. [92], Netzer et al. [143]
Recognition by specific features or similarity evaluation	Bag-of-features	Usually, local features are extracted and then quantized to form a histogram	Pros: Robust to global deformation since each character is decomposed into a set of parts (local features)
	Invariant features	Use of deformation (e.g., perspective) invariant feature	In [134, 135], the matching process is accelerated by a hash technique
Deformable template	Similarity evaluation after fitting a template nonlinearly to the input image	Theoretically, it provides a deformation-invariant similarity. A review of deformable templates is found in [136]	Yokobayashi and Wakahara [147]

(continued)

**Table 25.6** (continued)

Revision methods		Short description	Remarks	Example
Recognition of special characters	Recognition of highly decorative characters		Increase of font templates is a naïve solution. Another idea is to understand the global character structure [137]. The above deformable template is also a possible choice	Omachi et al. [137], Yokobayashi and Wakahara [147]
	Recognition of 3D characters	Recognition of characters with self-shadow	In [138], engraved characters are recognized	Mancas-Thillou and Mancas [138]
	Recognition of caption text	Since caption texts are degraded in various ways, special consideration is necessary		See the papers cited in section “ <a href="#">Methods of Text/Non-text Discrimination</a> ” of this chapter
	Recognition of a text captured by a handheld video camera	Combination of video mosaicing and text recognition	A simultaneous optimization of video mosaicing and text recognition is found in [139]	Uchida et al. [139]
	Recognition of license plate characters		In [4], there is a comprehensive list of trials to recognize license plate characters	Anagnostopoulos et al. [4]
	Recognition of characters designed for camera-based OCR	Development of so-called “OCR fonts” for camera-based OCR	The literature [140] tried to embed class information into the character shape by invariants	Uchida et al. [140]
Post correction by a trained model		Post correction to various errors due to the problems particular to scene characters	Typical misrecognitions (such as “o” to “n” due to the cut of lower part) are learned and modeled by a graphical model	Beaufort and Mancas-Thillou [148]

(i.e., synthesized) by applying “degradation models” to an original training sample set. For example, we can generate low-resolution character images by applying a down-sampling operation to the original high-resolution character images. Similarly, we can generate character images with blur and perspective by convolution with a PSF and transformation with a projection matrix.

Both of the above approaches are related to text localization. In fact, as we discussed in section “[Methods to Solve Problems in Text Localization](#)” of this chapter, there are methods of “[recognition-based localization](#),” where the location with a high score by a recognizer is detected as a text region. Clearly, these methods provide the recognition result of a text when the text is localized.

## Specific Features for Scene Character Recognition

Table [25.6](#) also lists more elaborated approaches with features which are not typical for ordinary character recognition. The “bag-of-features” approach uses a histogram-based feature vector. This approach decomposes a target character into parts, or local features, and makes their histogram through a quantization process. Its important property is that it will not represent global structure of the target character. This property realizes a recognizer which is robust to global and severe deformations in scene characters. (This property has been utilized in general object recognition.)

Invariant features are also a promising approach. Since their feature value does not change under a family of deformations (such as affine deformation and perspective), they have been employed in the recognizers. Theoretically, if we use an invariant feature, say, a perspective invariant feature, we can recognize characters perfectly regardless of perspective distortions. Furthermore, it is also beneficial that we need not to remove the perspective distortion by a preprocessing step. One point to be considered is that invariant features sometimes become less effective for character discrimination. For example, if a rotation invariant feature is used without any limitation, two different symbols “+” and “x” become identical.

## Recognition of Decorative Characters

In scenery images, we find various decorative characters. Figure [25.8](#) shows a very limited number of “A’s. Some of them are rather popular fonts and the others are highly decorative fonts. It is rather easy to deal with the popular fonts because we can add them to the training samples. Highly decorative fonts are still difficult to be recognized because it is impossible to cover huge variations of all possible decorative fonts. In fact, highly decorative fonts are often unreadable even by human without context. Figure [25.8c](#) shows an extreme example of “A.” It is difficult not only to recognize as “A” but also to detect as a character.

One possible strategy to deal with highly decorative fonts is to extract a topological structure, which is (nearly) invariant to decorations. Despite of the importance of this strategy, there have been only a few research trials, such as [[137](#)].



**Fig. 25.8** Various appearances of “A.” (a) Popular “A”’s and decorative “A”’s. (b) “A” as an initial capital. (c) A highly decorated “A” from the font set named “asphalt road surface” (By Yokokaku, Japan)

Feature representation by a set of character elements (introduced in Table 25.2), as well as bag-of-features approach, is also promising because the shape of character elements, i.e., local parts of a character, is often preserved even under decorations. Similarly, deformable template (also called image warping, nonlinear registration, and elastic matching) [136] is a possible choice. Since it fits two images by nonlinear mapping, it can minimize the difference between a standard font image and a decorated font image. In any case, recognition of decorated characters is still an open problem and further researches are expected.

---

## Dataset for Evaluation

As we saw in this chapter, camera-based OCR and videotext OCR are comprised of various modules, and therefore we need various datasets to evaluate those modules. The most widely used datasets are provided at ICDAR Robust Reading Competitions [66–68]. Their ICDAR2011 version contains 485 scene images where 1,564 words exist. There are other datasets for camera-based OCR, that is, KAIST scene character dataset (3,000 images), Google Street View dataset (249 images with 647 words and 3,796 letters), the Street View House Numbers dataset (600,000 digit images), IUPR dataset of camera-captured document images (100 images of warped document pages), NEOCR dataset (659 images with 5,238 text boxes), and the Chars74K dataset (7,705 scene character images together with handwritten and machine-printed characters). See also the list in ▶Chap 29 (Datasets and Annotations for Document Analysis and Recognition).

Unfortunately, the sizes of the above datasets are not sufficient for covering huge variations in character shapes, appearances, lighting conditions, etc. In addition, if we want to train a classifier by any machine learning method, we need to prepare training samples as many as possible. Consequently, we need to increase its size,

although it is highly costly to create a large dataset. A good news is that nowadays a huge number of scene images are freely downloadable from the Web (e.g., Flickr). Another good news is that it is possible to use a crowd-sourcing service (e.g., Amazon Mechanical Turk) for labeling images.

---

## Conclusion

This chapter described the tasks and its solutions of text localization and recognition in camera-captured images and videos. In some cases, such as recognition of camera-captured document images, the recognition task can be reduced to the ordinary OCR task by applying sufficient preprocessing steps, such as binarization, deblurring, estimation, and removal of perspective distortion. In a more general case, such as recognition of the shop name on a signboard, the task is far more difficult than the ordinary OCR task because it is necessary to localize the target text region (often without any prior) and then recognize the characters. This text localization step is a difficult problem despite of a large number of trials, which were introduced in this chapter in detail. Furthermore, recognition of scene characters is not a simple task; those characters are often different from those of ordinary documents and sometimes highly decorative. To overcome those difficulties, we need to integrate not only traditional image processing and character recognition techniques but also recent machine learning and computer vision techniques.

---

## Cross-References

- ▶ [Datasets and Annotations for Document Analysis and Recognition](#)
- ▶ [Imaging Techniques in Document Analysis Processes](#)

---

## References

1. Impedovo S, Modugno R, Ferrante A, Stasolla E (2009) New trends in digital scanning processes. In: International conference on document analysis and recognition (ICDAR2009), Barcelona, pp 1071–1075
2. Jung K, Kim KI, Jain AK (2004) Text information extraction in images and video: a survey. *Pattern Recognit* 37:977–997
3. Liang J, Doermann D, Li H (2005) Camera-Based analysis of text and documents: a survey. *Int J Doc Anal Recognit* 7(2–3):84–104
4. Anagnostopoulos CNE, Anagnostopoulos IE, Loumos V, Kayafas E (2006) A license plate-recognition algorithm for intelligent transportation system applications. *IEEE Trans Intell Transp Syst* 7(3):377–391
5. Frome A, Cheung G, Abdulkader A, Zennaro M, Wu B, Bissacco A, Adam H, Neven H, Vincent L (2009) Large-scale privacy protection in Google Street View. In: International conference on computer vision (ICCV2009), pp 2373–2380

6. Newman W, Dance C, Taylor A, Taylor S, Taylor M, Aldhous T (1999) CamWorks: a video-based tool for efficient capture from paper source documents. In: International conference on multimedia computing and systems (ICMCS1999), Florence, pp 647–653
7. Pollard S, Pilu M (2005) Building cameras for capturing documents. *Int J Doc Anal Recognit* 7(2–3):123–137
8. Shafait F, Cutter MP, van Beusekom J, Bukhari SS, Breuel TM (2011) Decapod: a flexible, low cost digitization solution for small and medium archives. In: International workshop on camera-based document analysis and recognition (CBDAR2011), Beijing, pp 41–46
9. Nakai T, Kise K, Iwamura M (2005) Hashing with local combinations of feature points and its application to camera-based document image retrieval – retrieval in 0.14 second from 10,000 pages. In: International workshop on camera-based document analysis and recognition (CBDAR2005), Seoul, pp 87–94
10. Liu X, Doermann D (2008) Mobile retriever: access to digital documents from their physical source. *Int J Doc Anal Recognit* 11(1):19–27
11. Okada Y, Takeda T, Kim Y-B, Watanabe Y (1998) Translation camera. In: International conference on pattern recognition (ICPR1998), Brisbane, pp 613–617
12. Gao J, Yang J (2001) An adaptive algorithm for text detection from natural scenes. In: IEEE computer society conference on vision and pattern recognition (CVPR2001), Kauai, vol 2, pp 84–89
13. Haritaoglu I (2001) Scene text extraction and translation for handheld devices. In: IEEE computer society conference on computer vision and pattern recognition (CVPR 2001), Kauai, vol 2, pp 408–413
14. Yokomizo K, Sono K, Watanabe Y, Okada Y (2003) Translation camera on mobile phone. In: International conference on multimedia and expo (ICME2003), Baltimore, pp 177–180
15. Chen X, Yang J, Zhang J, Waibel A (2004) Automatic detection and recognition of signs from natural scenes. *IEEE Trans Image Process* 13(1):87–99
16. Ezaki N, Bulacu M, Schomaker L (2004) Text detection from natural scene images: towards a system for visually impaired persons. In: International conference on pattern recognition (ICPR2004), Cambridge, vol 2, pp 683–686
17. Yi C, Tian Y (2011) Assistive text reading from complex background for blind persons. In: International workshop on camera-based document analysis and recognition (CBDAR2011), Beijing, pp 21–26
18. Liu Y, Yamamura T, Tanaka T, Ohnishi N (1999) Character-Based mobile robot navigation. In: International conference on intelligent robots and systems (IROS1999), Kyongju, Korea vol 2, pp 610–616
19. Wienecke M, Fink GA, Sagerer G (2005) Toward automatic video-based whiteboard reading. *Int J Doc Anal Recognit* 7(2–3):188–200
20. Munich ME, Perona P (1996) Visual input for pen-based computers. In: International conference on pattern recognition (ICPR1996), Vienna, pp 33–37
21. Iwata K, Kise K, Iwamura M, Uchida S, Omachi S (2010) Tracking and retrieval of pen tip positions for an intelligent camera pen. In: International conference on frontiers in handwriting recognition (ICFHR2010), Kolkata, pp 277–283
22. Liu Q, Liao C (2011) PaperUI. In: International workshop on camera-based document analysis and recognition (CBDAR2011), Beijing, pp 3–11
23. Zhang J, Kasturi R (2008) Extraction of text objects in video documents: recent progress. In: International workshop on document analysis systems (DAS2008), Nara, pp 5–17
24. Sato T, Kanade T, Hughes EK, Smith MA (1998) Video OCR for digital news archives. In: IEEE international workshop on content-based access of image and video database, Bombay, pp 52–60
25. Chen D, Odobez J-M, Bourlard H (2004) Text detection and recognition in images and video frames. *Pattern Recognit* 37(3):595–608
26. Kim W, Kim C (2009) A new approach for overlay text detection and extraction from complex video scene. *IEEE Trans Image Process* 18(2):401–411

27. Shivakumara P, Huang W, Phan TQ, Tan CL (2010) Accurate video text detection through classification of low and high contrast images. *Pattern Recognit* 43(6):2165–2185
28. Smeaton AF, Over P, Kraaij W (2006) Evaluation campaigns and TRECVID. In: ACM international workshop on multimedia information retrieval (MIR2006), Santa Barbara, pp 321–330
29. Zhang D, Chang S-F (2002) Event detection in baseball video using superimposed caption recognition. In: ACM international conference on multimedia (MULTIMEDIA '02), Juan-les-Pins, pp 315–318
30. Bertini M, Del Bimbo A, Nunziati W (2006) Automatic detection of player's identity in soccer videos using faces and text cues. In: ACM international conference on multimedia (MULTIMEDIA '06), Santa Barbara, pp 663–666
31. Wang J, Duan L, Li Z, Liu J, Lu H, Jin JS (2006) Robust method for TV logo tracking in video streams. In: IEEE international conference on multimedia and expo (ICME2006), Toronto, pp 1041–1044
32. Özay N, Sankur B (2009) Automatic TV logo detection and classification in broadcast videos. In: European signal processing conference (EUSIPCO2009), Glasgow, Scotland, pp 839–843
33. Shahab A, Shafait F, Dengel A (2011) Bayesian approach to photo time-stamp recognition. In: International conference on document analysis and recognition (ICDAR2011), Beijing, pp 1039–1043
34. Li H, Doermann D (1999) Text enhancement in digital video using multiple frame integration. In: ACM international conference on multimedia (Part 1) (MULTIMEDIA '99), Orlando, pp 19–22
35. Li H, Doermann D (2000) Superresolution-Based enhancement of text in digital video. In: International conference on pattern recognition (ICPR2000), Barcelona, vol 1, pp 847–850
36. Mancas-Thillou C, Mirmehdhi M (2005) Super-Resolution text using the Teager filter. In: International workshop on camera-based document analysis and recognition (CBDAR2005), Seoul, Korea, pp 10–16
37. Capel D, Zisserman A (2000) A super-resolution enhancement of text image sequences. In: International conference on pattern recognition (ICPR2000), Barcelona, vol 1, pp 600–605
38. Irani M, Peleg S (1991) Improving resolution by image registration. *Graph Models Image Process* 53(3):231–239
39. Donaldson K, Myers GK (2005) Bayesian super-resolution of text in video with a text-specific bimodal prior. *Int J Doc Anal Recognit* 7(2–3):159–167
40. Banerjee J, Jawahar CV (2008) Super-resolution of text images using edge-directed tangent field. In: International workshop on document analysis systems (DAS2008), Nara, pp 76–83
41. Bayarsaikhan B, Kwon Y, Kim JH (2008) Anisotropic total variation method for text image super-resolution. In: International workshop on document analysis systems (DAS2008), Nara, pp 473–479
42. Zappalá A, Gee A, Taylor M (1999) Document mosaicking. *Image Vis Comput* 17: 589–595
43. Sato T, Ikeda S, Kanbara M, Iketani A, Nakajima N, Yokoya N, Yamada K (2004) High-resolution video mosaicing for documents and photos by estimating camera motion. *Proc SPIE Electron Imaging* 5299:246–253
44. Baker S, Kanade T (2002) Limits on super-resolution and how to break them. *IEEE Trans Pattern Anal Mach Intell* 24(9):1167–1183
45. Dalley G, Freeman B, Marks J (2004) Single-frame text super-resolution: a Bayesian approach. In: International conference on image processing (ICIP2004), Singapore, vol 5, pp 3295–3298
46. Freeman WT, Jones TR, Pasztor EC (2002) Example-Based super-resolution. *IEEE Comput Graph Appl* 22(2):56–65
47. Park J, Kwon Y, Kim JH (2005) An example-based prior model for text image super-resolution. In: International conference on document analysis and recognition (ICDAR2005), Seoul, pp 374–378

48. Jacobs C, Simard PY, Viola P, Rinker J (2005) Text recognition of low-resolution document images. In: International conference on document analysis and recognition (ICDAR2005), Seoul, vol 2, pp 695–699
49. Taylor MJ, Dance CR (1998) Enhancement of document images from cameras. Proc SPIE 3305:230–241
50. Tian Y, Ming W (2009) Adaptive deblurring for camera-based document image processing. Lect Notes Comput Sci 5876:767–777
51. Kanungo T, Zheng Q (2004) Estimating degradation model parameters using neighborhood pattern distributions: an optimization approach. IEEE Trans Pattern Anal Mach Intell 26(4):520–524
52. Chen X, He X, Yang J, Wu Q (2011) An effective document image deblurring algorithm. In: IEEE computer society conference on computer vision and pattern recognition (CVPR2011), Colorado Springs, pp 369–376
53. Qi XY, Zhang L, Tan CL (2005) Motion deblurring for optical character recognition. In: International conference on document analysis and recognition (ICDAR2005), Seoul, pp 389–393
54. Clark P, Mirmehdi M (2000) Location and recovery of text on oriented surfaces. SPIE Conf Doc Recognit Retr VII 3967:267–277
55. Clark P, Mirmehdi M (2001) Estimating the orientation and recovery of text planes in a single image. In: British machine vision conference (BMVC2001), Manchester, pp 421–430
56. Dance CR (2002) Perspective estimation for document images. SPIE Doc Recognit IX 20–25
57. Myers GK, Bolles RC, Luong Q-T, Herson JA, Aradhya HB (2004) Rectification and recognition of text in 3-D scenes. Int J Doc Anal Recognit 7(2–3):147–158
58. Yamaguchi T, Maruyama M, Miyao H, Nakano Y (2004) Digit recognition in a natural scene with skew and slant normalization. Int J Doc Anal Recognit 7(2–3):168–177
59. Lu S, Chen BM, Ko CC (2005) Perspective rectification of document images using fuzzy set and morphological operations. Image Vis Comput 23(5):541–553
60. Liang J, DeMenthon D, Doermann D (2008) Geometric rectification of camera-captured document images. IEEE Trans Pattern Anal Mach Intell 30(4):591–605
61. Yin X-C, Sun J, Naoi S, Fujimoto K, Takebe H, Fujii Y, Kurokawa K (2007) A multi-stage strategy to perspective rectification for mobile phone camera-based document images. In: International conference on document analysis and recognition (ICDAR2007), Curitiba, pp 574–578
62. Uchida S, Sakai M, Iwamura M, Omachi S, Kise K (2008) Skew estimation by instances. In: International workshop on document analysis systems (DAS2008), Nara, pp 201–208
63. Shiraishi S, Feng Y, Uchida S (2012) A part-based skew estimation method. In: International workshop on document analysis systems (DAS2012), Gold Coast, pp 185–189
64. Lu S, Tan CL (2007) Automatic detection of document script and orientation. In: International conference on document analysis and recognition (ICDAR2007), Curitiba, pp 237–241
65. Koo HI, Kim J, Cho NI (2009) Composition of a dewarped and enhanced document image from two view images. IEEE Trans Image Process 18(7):1551–1562
66. Lucas SM et al (2005) ICDAR 2003 robust reading competitions: entries, results and future directions. Int J Doc Anal Recognit 7:105–122
67. Lucas SM (2005) ICDAR 2005 text locating competition results. In: International conference on document analysis and recognition (ICDAR2005), Seoul, pp 80–84
68. Shahab A, Shafait F, Dengel A (2011) ICDAR 2011 robust reading competition challenge 2: reading text in scene images. In: International conference on document analysis and recognition (ICDAR2011), Beijing, pp 1491–1496
69. Shivakumara P, Huang W, Phan TQ, Tan CL (2010) Accurate video text detection through classification of low and high contrast images. Pattern Recognit 43(6):2165–2185
70. Ma Y, Liu W, Bai X, Yao C, Tu Z (2012) Detecting texts of arbitrary orientations in natural images. In: IEEE conference on computer vision and pattern recognition (CVPR2012), Providence, pp 1083–1090

71. Ezaki N, Kiyota K, Minh BT, Bulacu M, Schomaker L (2005) Improved text-detection methods for a camera-based text reading system for blind persons. In: International conference on document analysis and recognition (ICDAR2005), Seoul, pp 257–261
72. Kuwano H, Taniguchi Y, Arai H, Mori M, Kurakake S, Kojima H (2000) Telop-on-demand: video structuring and retrieval based on text recognition. In: IEEE international conference on multimedia and expo (ICME 2000), New York, vol 2, pp 759–762
73. Sin B-K, Kim S-K, Cho B-J (2002) Locating characters in scene images using frequency features. In: International conference on pattern recognition (ICPR2002), Quebec City, vol 3
74. Liu X, Samarabandu J (2006) Multiscale edge-based text extraction from complex images. In: IEEE international conference on multimedia and expo (ICME2006), Toronto, pp 1721–1724
75. Bertini M, Colombo C, Del Bimbo A (2001) Automatic caption localization in videos using salient points. In: IEEE international conference on multimedia and expo (ICME'01), Tokyo, pp 69–72
76. Huang X, Ma H (2010) Automatic detection and localization of natural scene text in video. In: International conference on pattern recognition (ICPR2010), Istanbul, pp 3216–3219
77. Zhao X, Lin K-H, Fu Y, Hu Y, Liu Y, Huang TS (2011) Text from corners: a novel approach to detect text and caption in videos. *IEEE Trans Image Process* 20(3):790–799
78. Wolf C, Jolion J-M, Chassaing F (2002) Text localization, enhancement and binarization in multimedia documents. In: International conference on pattern recognition (ICPR2002), Quebec City, vol 2, pp 1037–1040
79. Wong EK, Chen M (2003) A new robust algorithm for video text extraction. *Pattern Recognit* 36(6):1397–1406
80. Kim W, Kim C (2009) A new approach for overlay text detection and extraction from complex video scene. *IEEE Trans Image Process* 18(2):401–411
81. Phan TQ, Shivakumara P, Tan CL (2009) A Laplacian method for video text detection. In: International conference on document analysis and recognition (ICDAR 2009), Barcelona, pp 66–70
82. Wang K, Belongie S (2010) Word spotting in the wild. In: European conference on computer vision (ECCV2010), Heraklion, pp 591–604
83. Uchida S, Shigeyoshi Y, Kunishige Y, Feng Y (2011) A keypoint-based approach toward scenery character detection. In: International conference on document analysis and recognition (ICDAR 2011), Beijing, pp 819–823
84. Mishra A, Alahari K, Jawahar CV (2012) Top-down and bottom-up cues for scene text recognition. In: IEEE conference on computer vision and pattern recognition (CVPR2012), Providence, pp 2687–2694
85. Anthimopoulos M, Gatos B, Pratikakis I (2010) A two-stage scheme for text detection in video images. *Image Vis Comput* 28(9):1413–1426
86. Chaddha N, Sharma R, Agrawal A, Gupta A (1994) Text segmentation in mixed-mode images. In: Asilomar conference on signals, systems and computers, Pacific Grove, vol 2, pp 1356–1361
87. Zhong Y, Zhang H, Jain AK (2000) Automatic caption localization in compressed video. *IEEE Trans Pattern Anal Mach Intell* 22(4):385–392
88. Goto H (2008) Redefining the DCT-based feature for scene text detection. Analysis and comparison of spatial frequency-based features. *Int J Doc Anal Recognit* 11(1):1–8
89. Jain AK, Bhattacharjee S (1992) Text segmentation using Gabor filters for automatic document processing. *Mach Vis Appl* 5(3):169–184
90. Saoi T, Goto H, Kobayashi H (2005) Text detection in color scene images based on unsupervised clustering of multi-channel wavelet features. In: International conference on document analysis and recognition (ICDAR 2005), Seoul, pp 690–694
91. Kumar S, Gupta R, Khanna N, Chaudhury S, Joshi SD (2007) Text extraction and document image segmentation using matched wavelets and MRF model. *IEEE Trans Image Process* 16(8):2117–2128
92. Weinman JJ, Learned-Miller E, Hanson AR (2009) Scene text recognition using similarity and a lexicon with sparse belief propagation. *IEEE Trans Pattern Anal Mach Intell* 31(10):

- 1733–1746
93. Kim KC, Byun HR, Song YJ, Choi YW, Chi SY, Kim KK, Chung YK (2004) Scene text extraction in natural scene images using hierarchical feature combining and verification. In: International conference on pattern recognition (ICPR2004), Cambridge, vol 2, pp 679–682
  94. Chen X, Yang J, Zhang J, Waibel A (2004) Automatic detection and recognition of signs from natural scenes. *IEEE Trans Image Process* 13(1):87–99
  95. Pan Y-F, Hou X, Liu C-L (2008) A robust system to detect and localize texts in natural scene images. In: International workshop on document analysis systems (DAS2008), Nara, pp 35–42
  96. Peng X, Cao H, Prasad R, Natarajan P (2011) Text extraction from video using conditional random fields. In: International conference on document analysis and recognition (ICDAR2011), Beijing, pp 1029–1033
  97. Pan W, Bui TD, Suen CY (2008) Text detection from scene images using sparse representation. In: International conference on pattern recognition (ICPR2008), Tampa, pp 1–5
  98. Coates A, Carpenter B, Case C, Satheesh S, Suresh B, Wang T, Wu DJ, Ng AY (2011) Text detection and character recognition in scene images with unsupervised feature learning. In: International conference on document analysis and recognition (ICDAR2011), Beijing, pp 440–445
  99. Yi C, Tian Y (2011) Text detection in natural scene images by stroke Gabor words. In: International conference on document analysis and recognition (ICDAR2011), Beijing, pp 177–181
  100. Shahab A, Shafait F, Dengel A (2011) Bayesian approach to photo time-stamp recognition. In: International conference on document analysis and recognition (ICDAR2011), Beijing, pp 1039–1043
  101. Shahab A, Shafait F, Dengel A, Uchida S (2012) How salient is scene text. In: International workshop on document analysis systems (DAS2012), Gold Coast, pp 317–321
  102. Gandhi T, Kasturi R, Antani S (2000) Application of planar motion segmentation for scene text extraction. In: International conference on pattern recognition (ICPR2000), Barcelona, vol 1, pp 1445–1449
  103. Park S-B, Oh K-J, Kim H-N, Jo G-S (2008) Automatic subtitles localization through speaker identification in multimedia system. In: IEEE international workshop on semantic computing and applications (IWSCA2008), Incheon, Korea, pp 166–172
  104. Kunishige Y, Feng Y, Uchida S (2011) Scenery character detection with environmental context. In: International conference on document analysis and recognition (ICDAR2011), Beijing
  105. Bhattacharya U, Parui SK, S Mondal (2009) Devanagari and Bangla text extraction from natural scene images. In: International conference on document analysis and recognition (ICDAR2009), Barcelona, pp 171–175
  106. Kim E, Lee SH, Kim JH (2009) Scene text extraction using focus of mobile camera. In: International conference on document analysis and recognition (ICDAR 2009), Barcelona, pp 166–170
  107. Rother C, Kolmogorov V, Blake A (2004) GrabCut: interactive foreground extraction using iterated graph cuts. In: Proceedings of the SIGGRAPH, Los Angeles, pp 309–314
  108. Messelodi S, Modena CM (1999) Automatic identification and skew estimation of text lines in real scene images. *Pattern Recognit* 32:791–810
  109. Gatos B, Pratikakis I, Perantonis SJ (2005) Text detection in indoor/outdoor scene images. In: International workshop on camera-based document analysis and recognition (CBDAR2005), Seoul, Korea, pp 127–132
  110. Chen X, Yuille AL (2004) Detecting and reading text in natural scenes. In: IEEE conference on computer vision and pattern recognition (CVPR2004), Washington, DC, vol 2, pp 366–373
  111. Hu S, Chen M (2005) Adaptive Frechet kernel based support vector machine for text detection. In: International conference on acoustics, speech, and signal processing (ICASSP2005), Philadelphia, vol 5, pp 365–368

112. Xu L, Nagayoshi H, Sako H (2008) Kanji character detection from complex real scene images based on character properties. In: International workshop on document analysis systems (DAS2008), Nara, pp 278–285
113. Hanif SM, Prevost L (2009) Text detection and localization in complex scene images using constrained AdaBoost algorithm. In: International conference on document analysis and recognition (ICDAR2009), Barcelona, pp 1–5
114. Ohya J, Shio A, Akamatsu S (1994) Recognizing characters in scene images. *IEEE Trans Pattern Anal Mach Intell* 16(2):214–220
115. Kusachi Y, Suzuki A, Ito N, Arakawa K (2004) Kanji recognition in scene images without detection of text fields – robust against variation of viewpoint, contrast, and background texture. In: International conference on pattern recognition (ICPR2004), Cambridge, vol 1, pp 457–460
116. Antani S, Crandall D, Kasturi R (2000) Robust extraction of text in video. In: International conference on pattern recognition (ICPR2000), Barcelona, vol 3, pp 831–834
117. Ren X, Malik J (2003) Learning a classification model for segmentation. In: International conference on computer vision (ICCV2003), Nice, vol 1, pp 10–17
118. Epshtain B, Ofek E, Wexler Y (2010) Detecting text in natural scenes with stroke width transform. In: IEEE conference on computer vision and pattern recognition (CVPR2010), San Francisco, pp 2963–2970
119. Matas J, Chum O, Urban M, Pajdla T (2004) Robust wide-baseline stereo from maximally stable extremal regions. *Image Vis Comput* 22:761–767
120. Cho MS, Seok J-H, Lee S, Kim JH (2011) Scene text extraction by superpixel CRFs combining multiple character features. In: International conference on document analysis and recognition (ICDAR2011), Beijing, pp 1034–1038
121. Wang X, Ding X, Liu C (2001) Character extraction and recognition in natural scene images. In: International conference on document analysis and recognition (ICDAR2001), Seattle, pp 1084–1088
122. Wang K, Kangas JA (2003) Character location in scene images from digital camera. *Pattern Recognit* 36(10):2287–2299
123. Mancas-Thillou C, Gosselin B (2007) Color text extraction with selective metric-based clustering. *Comput Vis Image Underst* 107(1–2):97–107
124. Liu Q, Jung C, Moon Y (2006) Text segmentation based on stroke filter. In: Annual ACM international conference on multimedia (MULTIMEDIA 2006), Santa Barbara, pp 129–132
125. Donoser M, Arth C, Bischof H (2007) Detecting, tracking and recognizing license plates. In: Asian conference on computer vision (ACCV2007), Tokyo, vol II, pp 447–456
126. Neumann L, Matas J (2010) A method for text localization and recognition in real-world images. In: Asian conference on computer vision (ACCV2010), Queenstown, vol III, pp 770–783
127. Merino-Gracia C, Lenc K, Mirmehdi M (2011) A head-mounted device for recognizing text in natural scenes. In: International workshop on camera-based document analysis and recognition (CBDAR2011), Beijing, pp 27–32
128. Neumann L, Matas J (2012) Real-time scene text localization and recognition. In: IEEE conference on computer vision and pattern recognition (CVPR2012), Providence, pp 3538–3545
129. Li C, Ding X, Wu Y (2001) Automatic text location in natural scene images. In: International conference on document analysis and recognition (ICDAR 2001), Seattle, pp 1069–1073
130. Huang R, Oba S, Palaiahnakote S, Uchida S (2012) Scene character detection and recognition based on multiple hypotheses framework. In: International conference on pattern recognition (ICPR2012), Tsukuba, pp 717–720
131. Goto H, Aso H (2001) Character pattern extraction from documents with complex backgrounds. *Int J Doc Anal Recognit* 4(4):258–268
132. Zhang D-Q, Chang S-F (2000) Learning to detect scene text using a higher-order MRF with belief propagation. In: Conference on computer vision and pattern recognition workshop (CVPRW2004), Washington, DC, pp 101–108

133. Hase H, Shinokawa T, Yoneda M, Suen CY (2001) Character string extraction from color documents. *Pattern Recognit* 34(7):1349–1365
134. Iwamura M, Tsuji T, Kise K (2010) Memory-based recognition of camera-captured characters. In: International workshop on document analysis systems (DAS2010), Boston, pp 89–96
135. Pan P, Zhu Y, Sun J, Naoi S (2011) Recognizing characters with severe perspective distortion using hash tables and perspective invariants. In: International conference on document analysis and recognition (ICDAR 2011), Beijing, pp 548–552
136. Uchida S, Sakoe H (2005) A survey of elastic matching techniques for handwritten character recognition. *IEICE Trans Inf Syst* E88-D(8):1781–1790
137. Omachi S, Inoue M, Aso H (2001) Structure extraction from decorated characters using multiscale images. *IEEE Trans Pattern Anal Mach Intell* 23(3):315–322
138. Mancas-Thillou C, Mancas M (2007) Comparison between pen-scanner and digital camera acquisition for engraved character recognition. In: International workshop on camera-based document analysis and recognition (CBDAR 2007), Curitiba, Brazil, pp 130–137
139. Uchida S, Miyazaki H, Sakoe H (2008) Mosaicing-by-recognition for video-based text recognition. *Pattern Recognit* 41(4):1230–1240
140. Uchida S, Sakai M, Iwamura M, Omachi S, Kise K (2007) Extraction of embedded class information from universal character pattern. In: International conference on document analysis and recognition (ICDAR 2007), Curitiba, vol 1, pp 437–441
141. Sawaki M, Murase H, Hagita N (2000) Automatic acquisition of context-based image templates for degraded character recognition in scene images. In: International conference on pattern recognition (ICPR2000), Barcelona, vol 4, pp 15–18
142. Saidane Z, Garcia C (2007) Automatic scene text recognition using a convolutional neural network. In: International workshop on camera-based document analysis and recognition (CBDAR 2007), Curitiba, Brazil, pp 100–106
143. Netzer Y, Wang T, Coates A, Bissacco A, Wu B, Ng AY (2011) Reading digits in natural images with unsupervised feature learning. In: NIPS workshop on deep learning and unsupervised feature learning, Granada, Spain.
144. de Camposm TE, Babu BR, Varma M (2009) Character recognition in natural images. In: International conference on computer vision theory and applications (VISAPP2009), Lisboa, pp 273–280
145. Lu S, Tan CL (2006) Camera text recognition based on perspective invariants. In: International conference on pattern recognition (ICPR 2006), Hong Kong, vol 2, pp 1042–1045
146. Li L, Tan CL (2010) Recognizing planar symbols with severe perspective deformation. *IEEE Trans Pattern Anal Mach Intell* 32(4):755–762
147. Yokobayashi M, Wakahara T (2006) Binarization and recognition of degraded characters using a maximum separability axis in color space and GAT correlation. In: International conference on pattern recognition (ICPR 2006), Hong Kong, vol 2, pp 885–888
148. Beaufort R, Mancas-Thillou C (2007) A weighted finite-state framework for correcting errors in natural scene OCR. In: International conference on document analysis and recognition (ICDAR 2007), Curitiba, vol 2, pp 889–893

## Further Reading

Since scene text recognition and caption text recognition are one of the hottest topics for document analysis researchers, papers introducing a new technique are easily found in various document-related conferences, such as ICDAR (International Conference on Document Analysis Recognition) and DAS (International Workshop on Document Analysis Systems). CBDAR (International Workshop on Camera-Based Document Analysis and Recognition) is a rather new workshop specialized for this

topic. Note that several competitions (called “Robust Reading Competition”) on this topic have been conducted at ICDAR and provided very good reference information to understand the performance of the state-of-the-art techniques. An interesting thing is that this topic has attracted researchers with various interests; many papers, therefore, also can be found in more general conferences, such as CVPR (IEEE Conference on Computer Vision and Pattern Recognition).

Journal papers on the topic are found in IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI), Pattern Recognition (PR), Pattern Recognition Letters (PRL), and International Journal of Document Analysis and Recognition (IJDAR), like the papers on other topics of document analysis and character recognition.

Since this is an emerging topic, there has been no complete book specialized for this topic (to my best knowledge). A couple of review papers have been published as introduced in this chapter. As noted above, for realizing practical scene text localization and recognition methods, we need to integrate various machine learning techniques and/or computer vision techniques. Consequently, introductory books on those techniques are useful for developing new techniques, as well as for understanding past trials.

---

## Part F

### Analysis of Online Data

We have now been discussing the analysis of static documents – mostly images of scanned documents – for several hundred pages. But what about documents which are dynamically created, that is, data for which we have not only the final result of a document image or representation, but also information about the way it was drafted, written, created? In this part, we will discuss the extension of document analysis and recognition to such online data.

The most immediate application, known to many users since it has become standard practice on tablets and smart phones, is that of the recognition of online handwriting. How can the full set of information about the way the writing was done help in the recognition of free or constrained text written by hand? Jin Hyung Kim and Bong-Kee Sin address this topic in ►Chap. 26 (Online Handwriting Recognition).

A specific problem with strong requirements for reliability is that of online signature verification, as Réjean Plamondon, Donato Impedovo, and Giuseppe Pirlo explain in ►Chap. 27 (Online Signature Verification).

But it is not only text which can be drawn by hand. Sketches, drawings, diagrams, etc., can be drafted on a tablet or some other interactive device; the question is then how good interfaces and efficient recognition software can interpret the user's intention and convert it into clean, precise, and structured graphical primitives. Liu Wenyin and Tong Lu have more on that in ►Chap. 28 (Sketching Interfaces).

---

# Online Handwriting Recognition

# 26

Jin Hyung Kim and Bong-Kee Sin

## Contents

Introduction.....	888
Overview.....	888
Online Handwriting.....	889
Script Classification.....	890
Organization of This Chapter.....	892
Preprocessing.....	892
Segmentation.....	893
Noise Filtering.....	893
Normalization.....	894
Feature Extraction.....	895
Recognition Methods.....	896
Simple String Recognition.....	897
Handwriting Recognition Techniques.....	900
Example Systems and Softwares.....	910
Research Systems.....	910
Commercial Recognizers.....	911
Conclusion.....	912
Executive Summary.....	912
Cross-References.....	913
References.....	913
Further Reading.....	915

---

J.H. Kim (✉)

Department of Computer Science, Korea Advanced Institute of Science and Technology,  
Yuseong-gu, Daejeon, Korea  
e-mail: [jkim@cs.kaist.ac.kr](mailto:jkim@cs.kaist.ac.kr)

B.-K. Sin

Department of IT Convergence and Applications Engineering, Pukyong National University,  
Namgu, Busan, Korea  
e-mail: [bkshin@pknu.ac.kr](mailto:bkshin@pknu.ac.kr)

**Abstract**

Online handwriting recognition has long been studied, and the technology has already been commercialized to some extent. But limited success stories from the market imply that further research is needed on electronic pen interface in general and recognition methods in particular. This chapter describes some of the basic techniques required to build a complete recognition software. At the turn of the millennium, there has been a renewed interest with focus shifted and diversified to multiple languages. Along with this trend, a lot of new ideas and efforts have been made to deal with different characteristics of different scripts. The difference notwithstanding, it should be helpful for designers to revert to the basics and review the established techniques and new ideas developed from afar the field before figuring out new – or maybe not very new – solutions to one's own language. Current big hurdles in online handwriting recognition include stroke order variation and multiple delayed strokes in addition to shape and style variations. This chapter ends with a short list of example systems and softwares, research based or commercial, that have been sort of landmarks or cited more often than not in the field.

**Keywords**

Delayed stroke • Handwriting recognition • Level-building algorithm • Spatial information • Stroke order variation

---

## Introduction

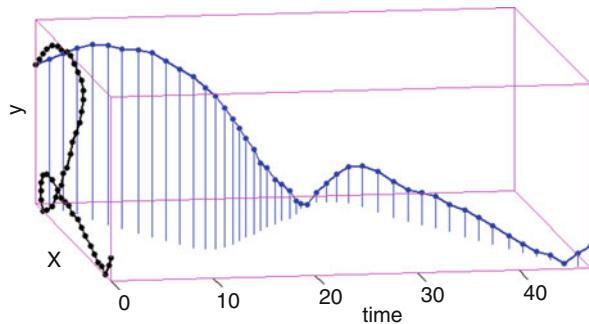
Although not in wide use even today, online handwriting is still considered a viable interface for future generation mobile devices and text input across many languages. This chapter introduces the basic issues of online script recognition with a special emphasis on differences among scripts and possible recognition strategies thereof. Given that perspective, an in-depth discussion is given around four sources of difficulty in modeling variability of online script patterns.

---

## Overview

The field of online handwriting has a long history dating back to the 1980s with the development of accurate and compact digitizer devices along with improved computing power for real-time recognition [1]. As reliable electronic pens became available, the idea of recognizing online handwriting led to a high hope of introducing a new modal of human-computer interface. This, however, has not been very successful due to unforeseen difficulties of character recognition. This chapter analyzes the difficulties and then discusses practical ideas and techniques that have been developed thus far to overcome those difficulties.

**Fig. 26.1** A spatiotemporal trajectory representation of an online handwriting



Up until 1990s the global research interest in online handwriting peaked higher and higher with most research activities targeting the recognition of Latin-based scripts, Kanji or Chinese characters used in Japan, and, to a lesser extent, Korean Hangul characters [2–4]. The research continued on and soon the market began to see a number of commercial products.

More recently we have come to witness a strong renewed interest in this area, this time with applications to a wider range of scripts which include Arabic [4–7], Devanagari, and related scripts in South and Southeast Asia [8, 9].

Since 2007, the popularity of smart-phones and tablet computers has driven the demand and wide acceptance of finger-based touch screens. This, however, have not helped the pen-based interface win users' attention. This can be construed as telling us that online handwriting has its place not in our everyday life but rather in more serious tasks like text editing with pen gestures and in special tasks like Chinese character input or mathematical expression input. For now it remains to be seen how things will develop for future online handwriting recognition.

## Online Handwriting

Online handwriting signal is a spatiotemporal signal that is intended to describe a geometrical shape. When viewed in spatial dimension only, it is simply a patterned collection of sample points in the two-dimensional planar surface. With the addition of the time dimension as shown in Fig. 26.1, it turns into a spatiotemporal trajectory of a pen. Those points are captured in a strict sequence; if we link them in order, the handwriting becomes a spatiotemporal curve characterizing the shape, writing order, and speed. One straightforward form of description that completely characterizes the curve would be  $(x, y, t), t = 1, \dots$ , or more conveniently,  $(x_t, y_t)$  with the same  $t$  values. This description is called a time series. Online handwriting is a sequential data that has dynamic information about the order of writing.

The dynamic information is valuable for character recognition since it provides us with the number and order of strokes. These are extremely valuable since they can render the task a lot easier by giving a good starting point: stroke. With the ordered and separate strokes given, we can bypass the difficult step of stroke identification or

segmentation and go directly on to concentrate on analyzing the shape of individual strokes and their relations.

Unfortunately, however, the blessing ends there. In English or in many other Western scripts, it continues. But for characters like Chinese, it is half a curse. There are tens of thousand characters in Chinese, and many of them have a lot of strokes, up to two or three dozens. In many of those characters, the order of writing strokes is often not obvious and thus tends to vary. This complicates recognition with additional variations not apparent in offline static images. It is obvious that the temporal information can be used to advantage. But too great a dependence on it may more than nullify the advantage. In this sense the dynamic information of handwriting is a mixed blessing for the designers of character recognizers.

There are times when the violation of the temporal order is unavoidable. In cursive handwritings like Arabic and Western words, a sequence of several strokes usually get connected to form a long stroke followed by zero or more delayed strokes like dots, bars, or even diacritical marks. These are largely script dependent. However, the knowledge of delayed stroke handling techniques or even stroke order-free methods in one script may be worth the effort since one can benefit therefrom for modeling his or her own script.

By convention optical character recognition refers to the offline recognition of character images. It takes an image of words to analyze their two-dimensional shape. An online handwriting data is distinguished from an offline image in that it is a sequential data corresponding to a curve in the spatiotemporal space. Therefore, when the data representation or the recognition technique is concerned, online handwriting is a different problem with entirely different kind of patterns.

But the final outcome of both tasks is the same: text. Hence, it is natural for system developers to think that one can benefit from the other. Since online handwriting has been more successful in both technology development and commercialization than the offline counterpart, some initial efforts were launched to recover dynamic information about the pen trajectory from offline images [10, 11]. But intrinsic ambiguities abound, so the effort was not very successful to date. An integrated approach of tracking pen trajectories based on robust online shape models could be one solution that needs to be tested [12].

## Script Classification

Most scripts on Earth today are linear or curved stroke-based system so that we can write characters using a pen. In essence online handwriting recognition is about shape analysis of those strokes given in sequence. The analysis usually proceeds from preprocessing of the input signal and then moves on to extracting effective features in a compact form suitable for shape classification and linguistic decoding. Most of these steps depend greatly on the type of script among others. Cursive scripts need sophisticated ways of modeling curves, while syllable-based

characters are free from character segmentation but often suffer from stroke order variations like Chinese. Hence, it will be helpful to know the differences and similarities among scripts and across languages.

Writing systems or scripts can be classified in many ways. But from the perspective of online handwriting, we can try a very simple classification based on the alphabets and the method of composing characters and words. Scripts as well as languages have evolved gradually, sometimes with histories dating back to many millennia ago. Today there are many alphabet-based scripts around the world. Among them, Western alphabets like Latin, Cyrillic, and Greek and Korean Hangul are considered pure alphabets with a large number of users. In Western scripts, the natural unit of handwriting is a word which is written spatially separated from others. It is a linear concatenation of letters from the alphabet and written in the corresponding order left to right, often with delayed strokes. The structure of most handwriting recognizers often reflects the geometrical structure. In fact, historically, most of the initial research efforts have been around those scripts. Therefore, any new online handwriting research should start with the literature thereof.

Korean Hangul is a syllable-based writing system where people write a word as composed of several characters written left to right or, at times, top to bottom. Each character is a composition of two or three letters called graphemes: consonant(C) + vowel(V) or C + V + final consonant(C). This composite character represents a syllable. A big difference from the Western word is that the letters are arranged two-dimensionally inside a virtual square. The spatial relation among component graphemes is very important, if not critical, for readers. Therefore, spatial relation modeling has been an important issue in Hangul recognition. The good news here is that the writing order of graphemes is highly natural that there are few exceptions; C + V or C + V + C.

Another script that is written usually horizontally like the Western scripts is Arabic, an abjad. Although written right to left, an Arabic word is a horizontal sequence of subwords, each consisting of a few Arabic alphabets with varying shapes depending on their position. Just like Hangul, the basic modeling unit of recognition could be such a subword representing a syllable. The overall recognition architecture can be similar to Western words but with many more delayed strokes.

If we move a little east from Southwest Asia or the Middle East, we come across a huge population using another syllabic script called Devanagari. Linguistically speaking, it is an abugida, a similarly consonant-based alphabet but with vowels explicitly indicated with diacritics or systematic graphic modification to the letters. Many other scripts in the Southeast Asia are also abugidas. All of them are believed to have descended from the ancient Brahmi script. Therefore, researchers in those scripts could benefit a lot from the knowledge of the others.

Unlike the previous scripts, Japanese kanas are syllabaries where each character represents a complete syllable with a different vowel. There is no composition of new sounds like the above alphabets; hence, they are not classified as an alphabet. Anyway, as the number of distinct characters is small, kana character recognition appears to be relatively easy. Hence, most Japanese scholars devote their effort to the

recognition of kanji, a set of Chinese characters used in Japan [3]. The techniques developed for kanji can be readily applied to Chinese character recognition and vice versa.

The Chinese character is quite distinct from others in the pattern recognition perspective. First of all it is a logosyllabic script. Each character has a meaning just like a word. But like many other syllable-based scripts, each character is written separately within a virtual bounding square. At first glance, the biggest feature would be that there are so many strokes. In many cases some of the strokes are grouped to form a stereotyped cluster called a radical. There are hundreds of distinct radicals in Chinese, and most of them are legal character in their own right. They can be combined to form a more complex character, leading to a hierarchical organization within a character. Apart from the complexity, the stroke order variation in handwriting stands out from among the daunting issues in Chinese character recognition. Other issues include modeling increasingly cursive shapes and complex spatial relations among the numerous strokes and radicals. It is a formidable task. Let us keep a close watch on how our Chinese and Japanese colleagues are doing.

## **Organization of This Chapter**

The organization of this chapter is made simple. First, section “[Preprocessing](#)” presents several tools of preprocessing. Then section “[Feature Extraction](#)” gives a brief description of feature extraction. The next section comprises the largest proportion of this chapter. It is subdivided into four subsections detailing aspects of difficulties in online recognition in general. The remaining sections describe example systems known in the literature and market and then conclude this chapter.

---

## **Preprocessing**

Online handwriting contains a range of sensor noise originated from the limitation of resolution grid and digitization process, mechanical switch of the pen, as well as the erratic hand motion [1]. These are often harmful to the recognition process. Hence, it is desirable and usually necessary to remove those factors before starting the actual computation. Additionally for practical reasons, we try to isolate units of recognition, which can be a character, an alphabetic letter, a stroke, or even a word (with multiple strokes) based on any shape features. The shape features are usually domain specific and require the knowledge about the target pattern class. But this kind of segmentation is sometimes applied before the recognition. Thus, a brief discussion is apt here.

## Segmentation

Segmentation in online handwriting refers to the isolation of scripts into recognition units such as words or characters, or even strokes. If this is done without knowledge of the target class, it is called external segmentation. Segmentation requiring recognition is called internal segmentation [1].

The simplest means of external segmentation is one done by the user. One natural method is giving a spatial clue where the user writes the next character well separated spatially from the last character of the previous word. Alternatively it is also possible that the user stops writing and then, after a time-out, the system starts to analyze the shape. This can be termed temporal clue for word segmentation, unavailable in offline character recognition.

In pure syllable-based writing systems like Chinese, Hangul, and Devanagari, each character corresponds to a syllable and written separately, thus removing the need for character segmentation. In this case the temporal clue is convenient, while the spatial clue is not obvious. This is especially the case in sequences of regular block characters where there are many vertical strokes and short strokes. Therefore, external segmentation in a character string is commonly performed by dynamic programming-based lattice search [3, 4].

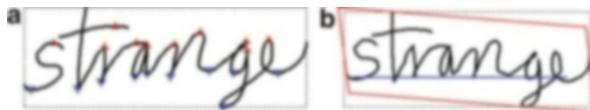
## Noise Filtering

Noise filtering involves various signal processing techniques that range from removing obvious random noise to reducing various redundancies. Typical random noise in online handwriting includes wild dots causing random jumps in otherwise smooth strokes. They are caused by hardware problems. Using a simple heuristic about physical limitation in abrupt hand motion, we can remove points of high acceleration and sudden change of direction.

Perhaps the opposite form of wild point includes dot loss causing brief loss of pen trajectory or extraneous pen lifts resulting in broken strokes. In the former case, we can introduce a number of intermediate points by interpolation depending on the size of the gap. A typical gap filter uses the Bezier curve [13]. In the latter case when the gap between the end points of the two strokes is small relative to the size of the character, we can simply connect the two strokes into one.

A related filtering technique is data reduction which refers to reducing redundancies in the input signal. Depending on the speed of writing, a sequence of points are often redundant implying no or little hand motion. Data reduction or dot reduction refers to discarding all of them except one.

A more common filtering technique is smoothing where a point in a stroke is replaced by a weighted average of its neighbors. This helps reduce small random noise and jitters embedded in the input signal. This type of filtering is useful when the input handwriting is small relative to the resolution of the digitizer.



**Fig. 26.2** (a) Local maxima and minima are marked by *upright* and *inverted triangles*, respectively, (b) the baseline aligned and characters deskewed by rotation and shearing transformation

There is an additional form of noise called a hook. A hook is an unintended part of a stroke at the start or the end of an intended stroke body. It is caused by an inaccurate pen-down or lifts and usually occurs at a sharp angle to the body stroke, thus called a “hook.” The hook can be interpreted as part of the pen-up motion between two strokes which happened to be registered by earlier pen-down or late pen-up motion. The hook can be considered as a kind of noise and removed. But it can also be modeled as a legal pattern of a ligature in natural cursive handwriting [14].

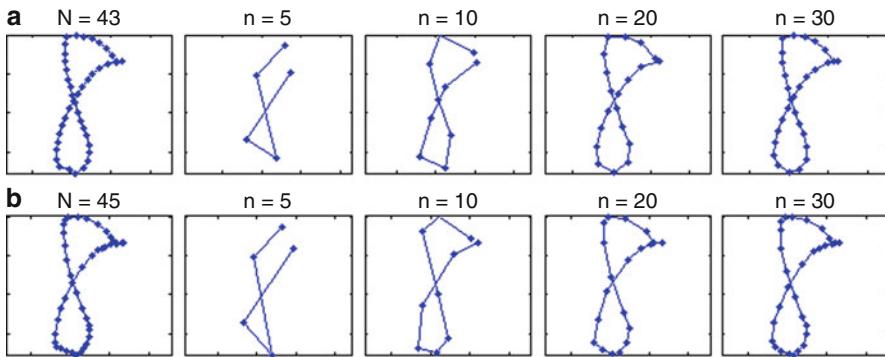
## Normalization

Handwriting normalization refers to the conversion of input handwriting to a canonical form that is free from the influence of sensor devices, writing habits, and environments. The primary goal of normalization is reducing shape and other geometrical variances in order to derive invariant features which are relevant to recognition.

The first and most obvious form of normalization is a linear transformation which includes scaling while preserving the aspect ratio, and rotation and shearing. Rotation is often called slant correction where the drifting baseline of the input handwriting is made straight and aligned to the horizontal line. The key is finding the baseline and midline. A typical technique of detecting a baseline is finding the local minima and then applying linear regression or Hough transform [13]. In the case of Latin-based words or Arabic, we can obtain both the baseline and the midline which are more reliable by constraining the slope equal to the midline aligned to the local maxima. Unfortunately, however, this does not work well for short words and often go awry in the presence of spurious extrema.

Individual characters are also written skewed. A typical deskewing method uses a heuristic of shearing the shape back to an upright form by noting the statistics of skewed region of strokes. See Fig. 26.2. This step is critical for assigning delayed strokes to the letter in the correction position.

Another normalization technique is resampling feature points each of which lie at a uniform distance from the immediately preceding point. This tries to remove the variation in writing speed throughout the entire handwriting and possibly across different handwritings of different scale in different environment. In handwriting the shape of the script is largely determined by curve bends, particularly sharp corners called cusps. So those points are usually retained in the transformed samples.



**Fig. 26.3** (a) Resampling in space for equidistant points, (b) resampling in time for equal travel time between points, obtained by Fourier transformation

Resampling is still a very common technique employed regardless of language [1, 3, 4]. It is useful as a means of data reduction, too. But the problem is that the loss of points is great around cusps and corners. Therefore, the spatial resampling could mean a loss of information in the temporal dimension. Hence, it is not always recommended to apply resampling to natural handwriting. Figure 26.3 shows examples of resampling in space and time by varying the number of resampling points.

## Feature Extraction

Feature extraction refers to the process of converting a raw data into a set of measurements in a form as required by the recognition algorithm. Despite intensive research during the past couple of decades, there has been no consensus yet about common or universal features that can be applied to any scripts. But it is not that bad because we are left with a rich menu of general, intuitive features that can be useful for diverse scripts.

Perhaps the most obvious kinds of features are local geometrical features like point coordinates, pen-down/up states, and pressure depending on the device. Some of the slightly more advanced and informative features are called “delta features” that include tangential direction or angle at sampled points. They can be described by direction code or by a continuous density function like von Mises or wrapped Gaussian [15, 16]. In addition, the local curvature and the writing speed can be expressed in the feature vector. These features are often robust and reliable in pattern description.

If a more meaningful and intuitive features are desired, we can derive high-level shape features like cusps, t-crossings, loops, and, in the case of Latin and Arabic words, ascenders and descenders. These are powerful by themselves telling us a lot

about the overall shape, but are difficult to detect and include into the sequential computation structure. They are by nature more of offline features than online.

Since local features generally do not capture the bigger picture of the handwriting, some form of segmental or regional features have been tried in most recognition systems. They include delayed strokes and related methods, Fourier descriptors describing an oscillating cursive stroke, regional configuration around a sample point often expressed as a small bitmap called a contextmap [13], and spatial relations to neighbor strokes [17, 18]. These features are usually language dependent, and thus, their use can be quite limited. Furthermore it is not obvious how to represent and evaluate them effectively.

A stroke in cursive online handwriting is often a connection of several strokes followed by shape distortion into a smooth winding curve. Depending on the modeling unit in the recognition stage, it is often necessary to segment the stroke into a sequence of basic sub-strokes. Common techniques of stroke decomposition involve the detection of feature points in a stroke, such as vertical or horizontal maxima and minima or simply cusps and points of great curvature.

Still another method dealing with a stroke is to take the whole stroke as it is and label it as an independent entity [4]. This idea has occasionally been tried in many studies. In one extreme, the label could be a simple code in some codebook, and a handwriting script can be described as a sequence of stroke codes. This idea makes the system very simple and works well for syllable-based regular scripts. But the problem is that it is always possible to encounter a new stroke belonging to none of the codewords. One popular technique is to use the stroke data as a template and then use the elastic matching algorithm for stroke or character classification. It is effective for small alphabet or vocabulary recognition tasks. The best example is Graffiti symbol recognition to be discussed in the next section. Note that the discussion has now crossed the boundary from feature extraction to recognition.

---

## Recognition Methods

Given a set of features, the recognition method in online handwriting is more or less determined, or vice versa. Here the method refers to the type and structure of pattern models and the associated classification algorithm. In this section, we assume an online handwriting is given as a time series, a sequence of local vectors. Each feature vector can be as simple as an  $(x, y)$ -coordinates.

If the online handwriting signal were a pure time series with only local information, the recognition model would be very simple. It is because some sequence of local decisions would exactly correspond to the global decision. Note, however, that a time series signal with strictly local features is often just a boring random noise of no one's interest.

In many real online handwriting signals, the feature vectors are largely correlated. Hence, most recognizers today try to incorporate regional or global features which are nonsequential and thus considered as static or offline features. The amount and extent of correlation depend on the type of patterns and scripts. In order to avoid

getting bogged down in details, let us again assume that the information is captured and represented in the sequence of feature vectors. The information so captured could be incomplete but should not be missed entirely.

A recognition model is a principled representation of the temporal information in real signals. And the associated recognition method should provide the computation of the model given the input data. However, there is no panacea for all scripts. Languages have different character sets or alphabets with different spelling or writing rules. Still, however, most scripts share many linguistic features; hence, it is not surprising that we can learn a lot from other languages. In fact all scripts around the world are neither completely random nor truly independent, but rather they are the products of human minds that have coevolved over a few millennia.

Let us first consider a very simple case of digit recognition before digging deeper into the issues of online handwriting recognition. The following discussion is highly educational providing an insight into the nature of handwriting recognition in general. In addition we will assume the HMM as the basic modeling tool.

## Simple String Recognition

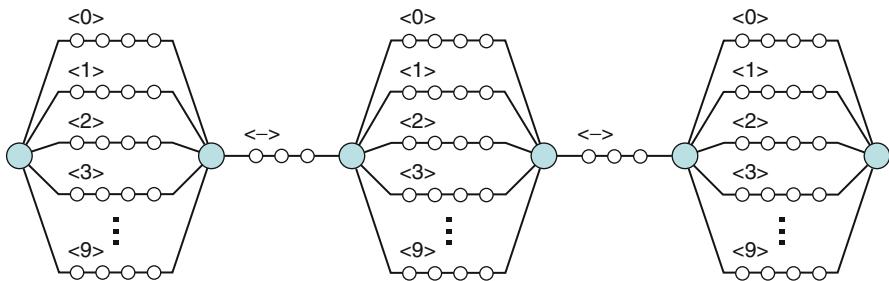
One archetypal example is the recognition of a numeral written as a digit string as shown in Fig. 26.4. In general digit strings, one digit does not affect the occurrence of other digits. Hence, the digits are essentially independent. For instance, let us consider the problem of evaluating a model of digit string  $W = W_1 W_2 \dots W_K$  given an input sequence  $X = X_1 X_2 \dots X_T$ . In such a case, we can replace the overall problem of evaluating the pair  $\text{score}(W, X)$  by a sequence of local computations  $\text{score}(W_k, X(k))$ ,  $k = 1, \dots, K$ , where  $X(k)$  denotes a subsequence mapped to  $k$ -th digit model. When the probabilistic function is used for the evaluation, we can rewrite it as

$$P(W, X) = P(W) \prod_{k=1}^K P(X(k)|W_k)$$

Now we can devote all our efforts to a sequence of smaller problems of computing simpler functions  $P(X(k)|W_k)$  representing the conditional density of the sequence  $X(k)$  given the model  $W_k$ . This deceptively simple function requires three kinds of missing information: the number of digits  $K$ , the identity of models  $W_k$ , and the boundaries of  $X(k)$ ,  $k = 1, \dots, K$ . The solution to this task comprises the core of digit string recognition.

A numeral or the digit string thereof consists of a sequence of random digits. See Fig. 26.4. Each digit can be any of the ten possibilities. Then the string is nothing more than a sequence of digit choices. The model architecture for this type of tasks is shown in Fig. 26.5, which is well-known and often employed as a baseline model in the literature [14, 19].

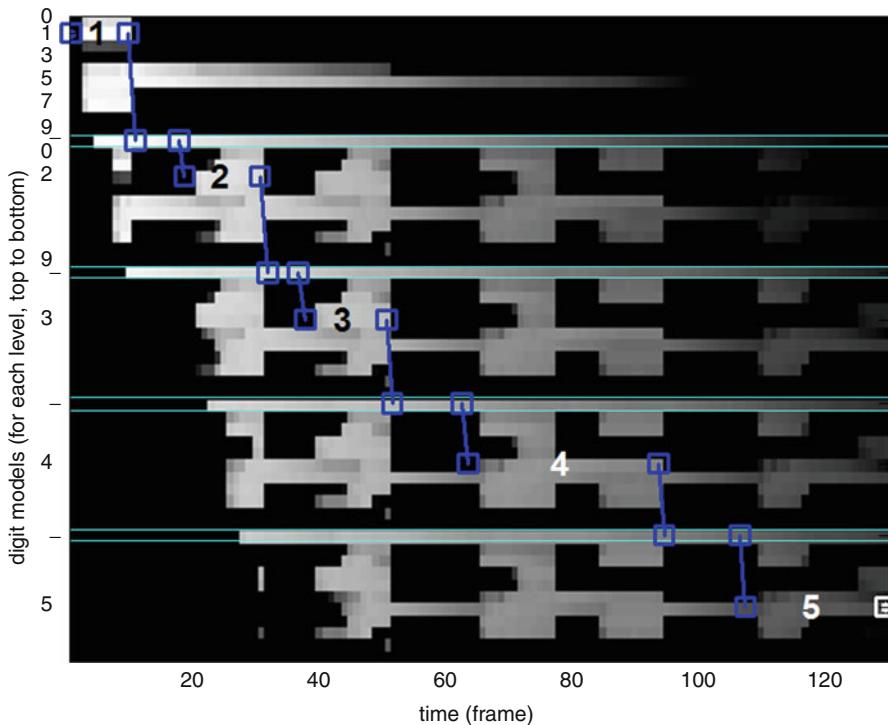
It is a direct translation of digit choices into a concatenated sequence of model choices. We can duplicate the choice an arbitrary number of times for arbitrary numeral recognition. Here we choose HMM as the model for individual digits.

**Fig. 26.4** A digit string**Fig. 26.5** Digit string net for string length

Then the resulting model becomes a network of HMMs. This way of modeling complex patterns by a combination of simple units is so general and useful that it can be applied to a wide range of dynamic signals when the locality conditions are met [14].

Given the network and an input observation sequence  $X$ , we are given a problem of optimization over the triples: the digit sequence  $W = W_1 W_2 \cdots W_K$  of unknown length  $K$  with the segmentation of input sequence  $S(X) = X(1) X(2) \cdots X(K)$ . The problem of finding the optimal value of the triple  $(K^*, W^*, S^*)$  is often called model decoding. Fortunately there are already several algorithms available [20–22]. Among them the level-building algorithm has been chosen here to obtain the optimal digit string that can be recovered by backtracking the forward likelihood computation chart as shown in Fig. 26.6.

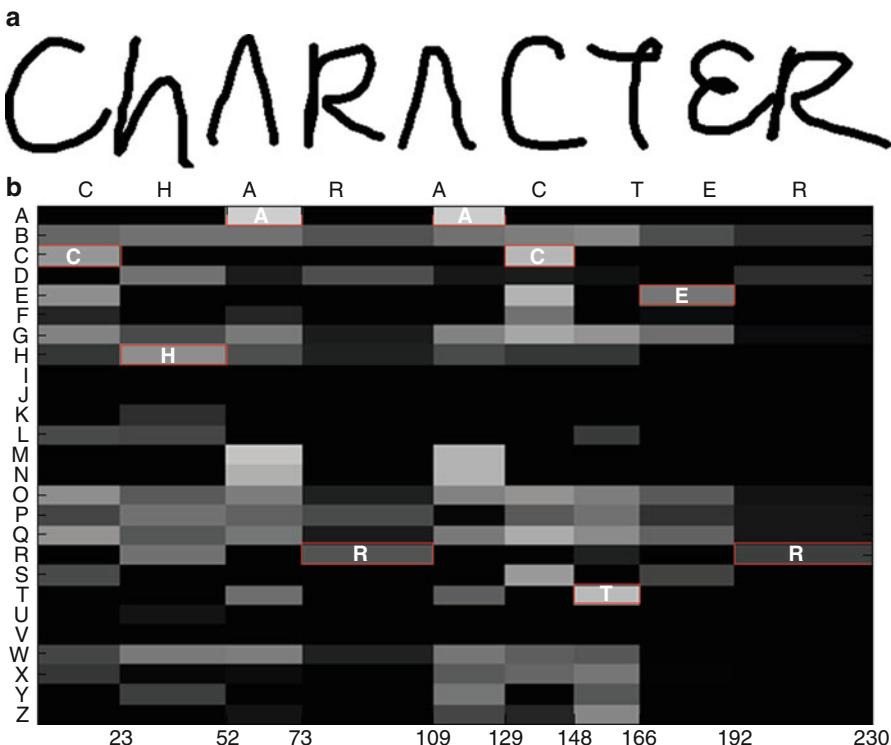
Each horizontal band of ten or one scan lines corresponds to a level, starting from the top. So the curve of downward staircase tells us that the first pattern is digit one. It is followed by a between-digit pattern called a ligature in the thin band, which is an imaginary inkless stroke. Here the intensity of each small block of pixels corresponds to the likelihood score of the final state of a digit or ligature HMM. The brighter, the greater score. So each single horizontal line represents the history of input sequence annotated by the corresponding model at the corresponding level. It would rise and fall depending on the current features and the preceding digit. The score must be the maximum in the sequential context to be selected as an output candidate.



**Fig. 26.6** Level-building chart for the input of Fig. 26.4. Each of the nine wide or narrow bands represents a level in sequence from *top* to *bottom*. There are ten digit models (from 0 to 9) in the wide bands, while only one ligature model in the narrow bands. The best model and the best (and brightest) segmentations in each band are marked inside the lattice by *numbers* and *square markers*. The sequence of digits “12345” as shown in the chart is the best candidate

With  $M$  HMMs and given a  $T$ -long sequence, the amount of computation involved is  $O(MN^2 \times TK)$  where up to  $K$  digits are expected. Accordingly, the complexity is already high. Fortunately there are heuristic measures to cut it down to a practical level, such as beam search, maximum string length, and slope constraints in the chart. In Fig. 26.6, we assumed that  $K \leq 5$  for the sake of illustration. The best number of levels should be determined simultaneously in the same computational framework.

In digit string recognition, the amount of computation can be reduced by the factor of  $K$  if the number of digits and digit boundaries are known. Graffiti (see section “Commercial Recognizers”) is a typical example where every character is written in a single stroke and there is no connecting successive strokes. Figure 26.7a shows a sample word written in Graffiti. In this case the number of strokes is the number of characters, and the end points of each stroke are its boundaries. Furthermore, we need not care about the between-character pen-up movements, and we can even “overwrite” the characters over the previous ones.



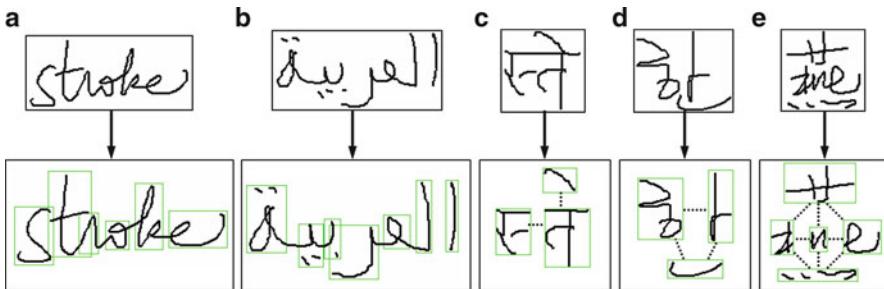
**Fig. 26.7** (a) a Graffiti sample and (b) a computation lattice

Note that Graffiti word recognition is in fact an isolated character recognition problem. The classifier is merely applied to each stroke separately. Each column of Fig. 26.7b shows the matching score of the models given a stroke, where brighter cells represent a greater matching score for the corresponding model. There is only one classifier, which can be realized by any of the known methods, such as template matching, and neural network, in addition to the HMM.

### Handwriting Recognition Techniques

It is well known that the HMM is a very useful tool for modeling temporal variability of dynamic signals. So the HMM with the dynamic programming search algorithms developed for the HMM could be the tools of choice. But the problem is not so simple when it comes to online recognition of most scripts beyond the “boring” digits. Hence, the rationale for all our efforts even today.

Here is a list of some major sources of difficulty in online handwriting that includes:



**Fig. 26.8** Handwriting structure analyzed (a) an English word; (b) an Arabic word; (c) a Hindi character or part of some word; (d) a Korean Hangul character, also part of a word; and (e) a Chinese character, also part of a word

- Characters have a spatial structure, and there is a global shape which is not easy to capture with simple local feature measurements.
- Stroke order variations and delayed strokes which complicates the sequential processing of signals.
- Stroke connection in fluent and fast script, and subsequent shape distortion and ambiguity.
- Large vocabulary or character set that can strain even powerful multi-core processors.

Let us explore each of these issues in turn in a wide perspective.

### Modeling Spatial Structure

A character, be it machine printed or handprinted, is by nature a spatial pattern that has a certain shape on a two-dimensional plane. In online handwriting, characters are “drawn” sequentially stroke by stroke, represented as a time sequence of point coordinates, and then expressed as a sequence of feature vectors. The problem is that these feature vectors represent only local features of the strokes and fail to capture the overall shape of the characters. Even though the global information is contained in the sequence implicitly, it is not captured well in most character or shape pattern models due to their basic modeling assumptions and structural reasons. For example, the well-known first-order Markov assumption makes it impossible to tell what happened before a single clock unit. But character strokes or their features are correlated far and wide. This is known as “the long-range correlation” problem that has challenged numerous researchers around the globe. How do we solve this problem? Let us look at Fig. 26.8.

Western and Arabic words. Although there are differences, most Western words and Arabic scripts are similar in that they are written linearly in one direction, left to right or right to left. Frequently there are delayed strokes which complicate sequential processing. Arabic scripts just have more delayed dots above or below the body strokes. Characters differ depending on the presence and position of delayed dots. There are optional diacritics for marking vowels, especially in print

documents. Since, however, Arabic describes a sound by composing a few letters, one may benefit from the knowledge of syllable-based alphabetic writing systems like Hangul.

Devanagari characters. Basically it is a syllable-based character writing system. There are some 48 alphabets mostly for consonants. In addition there are diacritical marks to denote vowels and rules of composing two or more characters. Figure 26.8c shows a composite character that combines two basic Devanagari alphabets. There are three components; the two lower components represent the combination of sounds s(a) + ta into “sta,” while the diacritic in the upper block specifies the vowel change to “e.” Thus, the resulting character represents “ste.” What is noteworthy is the appearance of two-dimensional structure unlike Western words. Unlike Arabic the diacritical marks are mandatory.

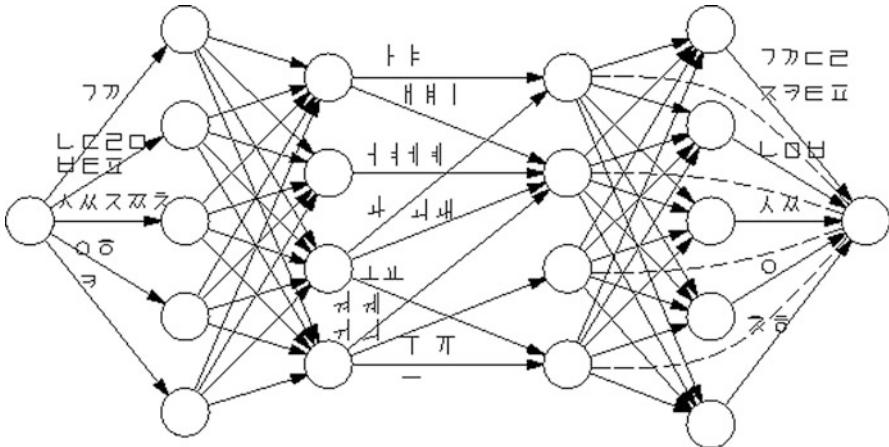
Korean Hangul is a phonetic writing system based on a pure alphabet with 24 letters including separate vowel symbols. Its basic writing unit is a character which represents a syllable. Each character combines two or three letters arranged inside a virtual square as in Fig. 26.8d. These features have been incorporated into the structure of the cursive character model [14].

Also known as a featural alphabet, Hangul has a very simple character composition and writing rule allowing us to spell any sound as it is heard. Figure 26.8d shows a sample character with three letters **C**(consonant, left) + **V**(vowel) + **C**(below), combined into a box (*/h/+/a/+/n/ = /han/* where */a/* is said like “a” in “palm”). There are only six different arrangements determined by the shape of the vowels. The **CV** or **CVC** order is so natural that there are few exceptions in Hangul handwriting. Stroke order variations within a letter are not rare, though. So the HMM-based method, aka BongNet, has been very successful for cursive Hangul character recognition [14]. In this method, the ligature modeling has first been tried to represent ligature parts in cursive strokes and the direction of movement or the spatial relation between strokes and letters. According to this method, the character in Fig. 26.8d is described and to be decoded as

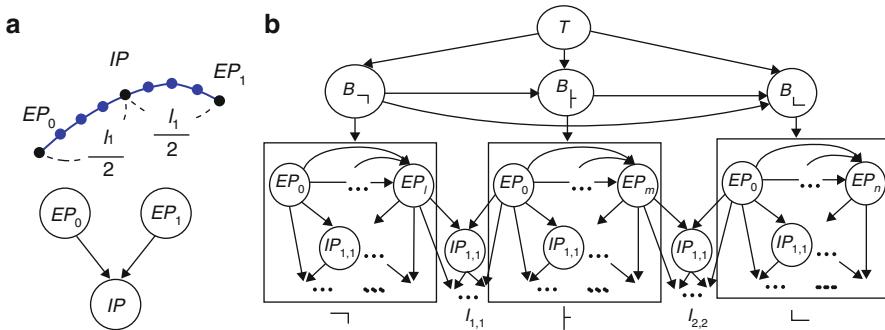
$$/h/ + / - /^{5,7} + /a/ + / - /^{11,16} + /n/$$

where the superscripts denote the ligature identifiers as used in Fig. 26.9 with an appropriate node numbering. BongNet models the spatial relation implicitly using a set of separate HMMs and applies a frame-synchronous version of Viterbi algorithm which builds levels while making node transitions to the right [21, 22].

The spatial structure of a character is very important, if not critical, in Hangul. Then we can even choose to focus only on modeling spatial relations between various components in a Hangul character [17]. Given a sequence of strokes, we can divide each stroke into halves like Fig. 26.10a. The two segments have a certain spatial relation called a within-stroke relation (WSR), which can be modeled by a small Bayesian network as shown to the right with three nodes. The stroke can further be divided recursively until some covariance condition is met resulting in a proportionately complex network as shown within each of the square blocks of Fig. 26.10b. On top of these, the relationships between strokes (ISR or inter-stroke



**Fig. 26.9** Online Hangul character model BongNet. There are six columns of grammar nodes including the start node on the *left* and the end node on the *right*. They are numbered in order left to right and top to bottom. The leftmost five arcs represent consonant types, and each arc is labeled by a group of consonant letter HMMs. The mesh of arcs between the second and third columns of nodes is labeled by an HMM for a particular ligature shape



**Fig. 26.10** Bayesian network-based modeling of (a) feature point relation in a stroke, and (b) inter-letter and inter-stroke relation in Hangul character /gan/

relation) and between graphemes (IGR or inter-grapheme relation) can similarly be modeled by a Bayesian network over graphemes. This method could be most effective for regular scripts where strokes are written discrete in scripts like Hangul and Chinese characters.

Chinese character and its variants like Japanese Kanji and Korean Hanja go to the extreme of exploiting the spatial relations among character components called radicals [4]. There are numerous radicals, as many as 300, in the first place. The sample character in Fig. 26.8e has five radicals. Each radical or its variant is an independent character in its own right. We can create a character by combining two or more radicals and arranging them inside a 2D rectangular block with an

appropriate scaling. Unlike Hangul character, the spatial composition in Chinese character is somewhat arbitrary if not completely random.

One of the most distinguishing features of Chinese character is the hierarchical composition of radicals, with a complex multi-radical character at the top and a number of individual strokes at the bottom. Those components at any level are arranged spatially and can comprise a different character with different arrangement. In online handwriting, this kind of spatial information is no more local than the shape information is. It is not easy to represent it in sequential feature vectors. For this and other reasons, many researchers have turned away from the HMM for more static and structural models.

The structural information in general implies the global shape where, typically, some feature points are separated apart in time but are close to each other geometrically and thus related. This kind of long-range correlation is not easy to capture with sequential feature vectors. Besides, the next issue of writing order variation has led us to almost forgetting the dynamic information altogether and rather relying on static offline features. For this reason, Chinese character is special as far as its recognition method is concerned. Despite numerous research efforts, there is no consensus as yet about established features and recognition methodology. Investigations are still ongoing. For instance, we can extract primitives at the level stroke or sub-stroke and evaluate them using a distance function or DP matching or even HMMs. Then we can employ an attributed relational graph (ARG) or a decision tree to analyze the hierarchical structure and between-component relationships [23]. In Chinese character recognition, the problem is further complicated by a substantial amount of variations in writing order.

Finally, if we extend the target to a sentence or a sequence of characters, we have to deal with detecting character boundaries. Surprisingly enough, even in character-based writing, this is not so simple because character boundaries often become ambiguous as the number of strokes increases. A popular solution is to apply the dynamic programming search again that evaluates all possible stroke combinations and generates the best while consulting a dictionary and other linguistic information.

### **Modeling Stroke Order Variation**

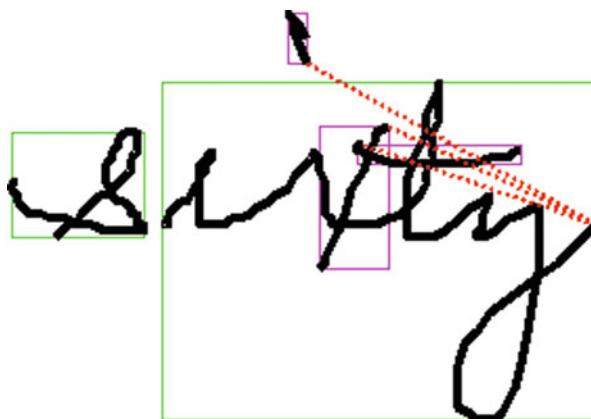
The dynamic information of online handwriting has been perceived to be useful by default, often leading to the view that the recognition problem would be easy without challenging issues. But an appropriate response out of experience would be “It depends” or “Yes or no.”

As remarked earlier, the dynamic information of online handwriting can be a double-edged sword. First of all, it gives the number and the order of strokes, and the direction of stroke direction. Definitely these are valuable and can be used to advantage, as long as they do not depart from the norm. But there are always exceptions and variations, which come in two forms.

The first one is delayed writing of strokes. Examples are found in i-dots, t-bars, and x-crosses in Latin alphabet-based scripts. See Fig. 26.11. They can be detected using simple heuristics like their relative size and position with respect to the last

**Fig. 26.11** Delayed strokes.

They are written to the left, opposite to the writing direction, with respect to the end point of the body stroke



body stroke and its end point. Once they are identified as delayed strokes, they are removed from the normal input sequence of body strokes. Here we introduce three kinds of letter models for “i,” “j,” “t,” and “x” with three possible writing orders. For example, the body stroke of “i” can be written after a dot, or immediately before a dot, or long before a delayed dot. The first two cases pose no problem. Only for the third case do we have to design a dot-free model and include a postprocessing step that follows the forward search like the string recognition of section “[Simple String Recognition](#).” When a dot-free “i” pattern is detected from a string search, then we reevaluate the segment including any nearby dot that has been left out previously.

One final remark about delayed stroke processing is that, although cumbersome, you can eventually jump clear the hurdle quite successfully in the case of Western and quite possibly Arabic scripts. There are some important similarities in Arabic scripts. Although written right to left with many delayed dots above or below, we can apply a similar method, mutatis mutandis, to recognize Arabic handwriting.

The second form of stroke order variation concerns a direct challenge to the basic assumption in online handwriting recognition. This is best exemplified by the stroke variations in Chinese characters. There are general rules in the order of writing strokes in a character. Children do learn and practice them. But over time they form their own style while forgetting or breaking the rules. The great number of strokes in many characters and their complex shape may be contributing to the degradation.

A Chinese character in general has a lot of strokes which are arranged in a complex way within a bounding block. Naturally people find different ways and orders of writing and gradually harden it into their own writing style. The order variation is arguably the greatest issue that has long harassed Japanese and Chinese researchers. In fact, because of the frequency and variety, many have given up using the HMM framework, opting for more classical tools like template matching and nearest classifier [3, 4].

Three approaches are possible to resolve order variation in Chinese characters. The first method is to limit the variation only at the level of radicals [3]. Radicals are found at any levels in the hierarchy of character structure, and their size and shape can vary depending on the position in the hierarchy. By applying a suitable affine transformation to each of the canonical radical models, a whole character model can be built by consulting a character dictionary. Note that a character is viewed as a two-dimensional layout of several radicals scaled, distorted, and translated into a square. This is compared to a Hangul character or even a Western word, only with such a big alphabet. Here one observation or, more precisely, an assumption is that the number and the complex organization of radicals in a character frequently lead to order variations among those radicals, but not within individual radicals. Although not without exceptions, this assumption seems reasonable that it is considered to be worth trying [3].

The second method is going to the extreme of simply forgetting the stroke order [4]. Still, however, individual strokes are identified and thus can be used to advantage, much greater than in offline characters. The problem can be approached in two steps: stroke shape analysis and stroke relation analysis. Stroke analysis involves analyzing the structure of a stroke and identifying its type. Any of the traditional methods can be employed including template matching and neural network. Based on the result, we can analyze the spatial relations between strokes and radicals, with one option being the attributed relational graph as referred to in section “[Modeling Spatial Structure](#).” This second step could be similar to analyzing spatial relation after stroke extraction. Unfortunately, however, it is rather (doubtful) if there is much to gain from the conversion to offline recognition problem. Although offline task has been studied a lot, we have yet to find a breakthrough there. Therefore, it would be more desirable to stay and play within online recognition framework for now.

Still another method is searching the best stroke correspondence to reference models. There is an interesting effort named cube search that tries to find the best correspondence between strokes in input to references [24]. It will most likely be useful for discrete regular scripts.

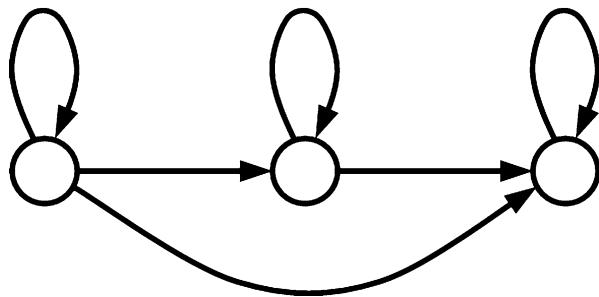
The basic assumption in stroke order resolution is that people write regular or less cursive scripts. In normal and natural handwriting of characters or words in most languages, people often write cursively connecting strokes and subsequently distorting the overall shape. This is the third source of difficulty in handwriting recognition.

### **Modeling Cursive Strokes**

The third point of difficulty in online handwriting recognition is the great variability in cursive writing. Everyone has his or her own writing style developed gradually over time since the first grade at school. The style is the result of optimizing the motor skill of the hand and arm [25].

As one masters handwriting, strokes become smoother and rounder, entailing considerable shape distortion insofar as the characters do not get ambiguous. At first, two or more strokes written in succession get connected. Simply without pen-lifts

**Fig. 26.12** Ligature model topology



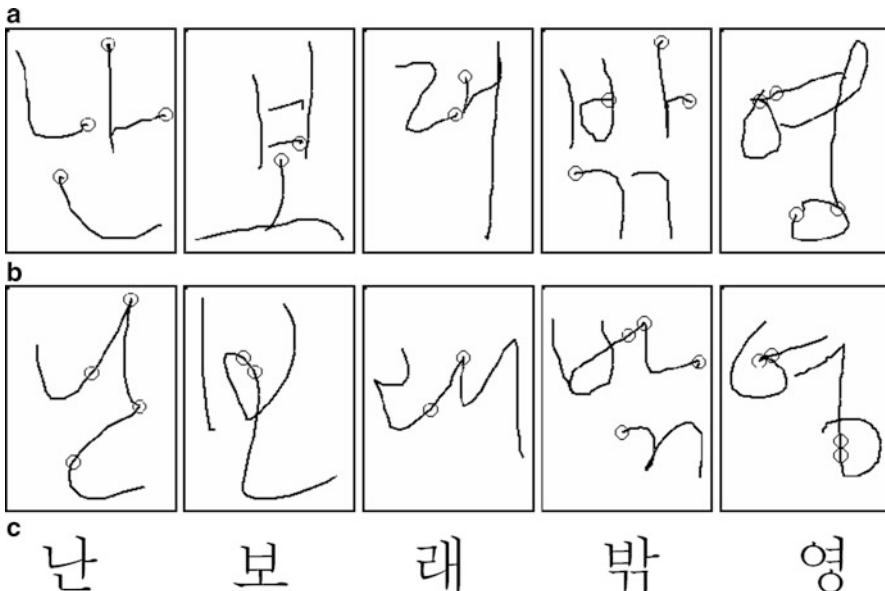
or with pen-down drags, the strokes are connected into one. In the latter case, the added part of the stroke is referred to as a ligature [14]. In either case, once strokes are linked, the pen motion is constrained physically and psychologically and causes substantial shape change. Except for the originally cursive scripts like Arabic and cursive Latin, the amount of distortion is highly dependent on the connection and the degree of writing skill. This being the case, it is necessary to model the ligature patterns explicitly.

For instance, it is strongly desired to introduce a ligature model if a linking pattern between two letters or characters occurs frequently. Rejecting them or discounting them as mere nuisance or simply a variable part of a character means avoiding the reality and consequently makes the pattern modeling more difficult.

One method of modeling ligatures is adapting the idea of triphone which is popular for continuous speech recognition [26]. It is in fact a context-dependent model where each phoneme is assigned multiple models each of which is distinguished by its left and right context. Of course, the number of models will grow geometrically and render training a lot more difficult. But we can reduce the number down to a practical level by clustering the two end contexts. The resulting model with context implies a ligature in online handwriting. The difference is that the script ligature model is not part of a character but a separate entity representing a combination of the right and the left contexts of two successive letters [14].

We can define a number of distinct ligatures based on the context and shape. Those ligature patterns appear linear or almost linear with characteristic direction. Each type of ligatures can be modeled using an HMM. One big advantage of using HMM is that both pen-down patterns and pen-up virtual patterns can be modeled in one homogeneous framework. Figures 26.5 and 26.9 show two examples of handwriting pattern model with a set of ligature models embedded in alternate levels. Considering the simple shape but variable length of ligatures, we can design an HMM with a very simple topology as shown in Fig. 26.12. The number of states can be adjusted, if desired, depending on the pattern length statistics.

Let us look at the picture in Fig. 26.6 where the even numbered levels are assigned a single ligature HMM. In the digit string of Fig. 26.4a, there is no pen-down ligature stroke, and all the pen-up ligature motions are not observed in the input except digits 4 and 5. Thus, the activity of digit HMMs in the ligature section is almost nonexistent (of course except digits 4 and 5).



**Fig. 26.13** Letter segmentation examples of five characters, discrete (*upper*) and cursive (*lower*), and labels at the bottom. Note that many segmentation points are on the smooth regions of strokes. It is almost impossible to detect them by heuristic methods only due to lack of prominent features around them and shape variability in free, cursive handwriting. But they have been found by the HMM-based recognizer of Fig. 26.9 computing simultaneous recognition and segmentation. There are many ligatures which are usually short, some very short, and linear

When it comes to cursive handwriting, we can distinguish some scripts which are not developed by personal skill but evolved out of history. The pure cursive script in English and the grass script in Chinese are two examples. They are so different from regular block characters that it would be better to regard them as a different script with distinct alphabets or character sets. To recognize them, developing certain forms of functions for handwriting distortion transformation could be an idea but it would not be successful in covering all kinds of nonlinear shape variations. In fact we have no special reason to object to studying hand motor and writing behaviors [25]. But the right and more effective way of doing research in online handwriting recognition will be directly modeling the shape features and their variability from a given dataset.

### Character Set and Large Vocabulary

As a means of communication, most modern languages have tens or even hundreds of thousand words in their vocabulary as well as grammatical rules, for naming or describing physical objects, abstract concept, relations, phenomena, and actions.

These words are the basic targets of handwriting recognition. In some cases like logographic Chinese, they could be characters. For practical real-time recognition of these words or characters, it is imperative to reduce the computation and space.

The large character set of Chinese poses some practical difficulty for character recognition. Although it may not be as hard as that of stroke order variation and shape variability, it has long been an important topic in optical character recognition [4]. One common approach to large set character recognition is to reduce the candidates down to a manageable level, say 100 or less. This can be achieved in two ways: one is coarse preclassification of the character set into clusters and/or building a decision tree using some predefined set of features, such as the number of strokes. Once done, the result can be used repeatedly for all future input characters.

Another method of candidate reduction is to apply some dynamic evaluation function using some of the input features and generate a small set of candidate characters. This is applied to each input character resulting in a different set. In both methods, a detailed classification should follow the initial coarse classification. Chinese characters are complex with as much as three dozen strokes; therefore, the resulting models could be proportionately complex both in structure and computation. Hence, it is essential to introduce any type of coarse classification for reducing candidates.

Large vocabulary recognition requires a dictionary [27]. Although written as a sequential combination of letters or syllabic characters, the basic unit of writing and communication is word. Simple string recognition could return an arbitrary sequence of letters. Thus, we have to filter out those illegal, nonsensical words in order to build a practical system. This can be done by incorporating a priori linguistic knowledge, often in the form postprocessing after shape-based classification.

Another popular method is incorporating a language model, either a word lexicon or a statistical language model, into the recognition engine. Screening illegal words using a word lexicon can be embedded into the sequential computational structure of the recognizer [13]. For this, it is most appropriate to organize the lexicon into a trie that can help search only legal candidates. Dictionary-based method, however, suffers from the problem of rejecting many proper nouns like a friend's name. The statistical language model comes to the rescue here. Also called an N-gram or a Markov source model, it uses the statistical information about substrings of certain lengths; it is called a bigram if  $N = 2$  and a trigram if  $N = 3$ . These are popular partly because these models fit the Markovian property of HMM and the optimality principle of dynamic programming.

Still another method of incorporating linguistic knowledge is to configure the handwriting pattern model like Figs. 26.5 and 26.9 dynamically based on bigram or trigram [28]. As the forward computation of the Viterbi algorithm proceeds, the model network is made to expand level by level. In this case it is natural to use

context-dependent models like triphone equivalents in handwriting recognition. In fact this is a well-known approach in continuous speech recognition.

---

## Example Systems and Softwares

Although online handwriting recognition still abounds with problems and issues, we can say that the technology of the field has largely matured with a number of high-performance systems and commercial products. In this section, let us briefly review some of them.

### Research Systems

Most, if not all, research efforts and papers in the literature in online handwriting recognition present certain forms of recognizers or, at least, methods and algorithms. Therefore, it is impossible to enumerate all of them here. Hence, just a few of them which are relevant to this chapter are named here.

#### BongNet/UniRec

BongNet is online cursive Hangul character recognition model for Korean Hangul characters [12, 14, 29]. Developed at Korea Advanced Institute of Science and Technology (KAIST), this system was the first to introduce explicit modeling of ligatures as separate entities. Along with a frame-synchronous version of level-building algorithm, the model, as shown in Fig. 26.9, solves the problem of mixed cursive handwriting recognition while determining the optimal boundaries of letters and ligatures simultaneously. The performance recorded 93.7 % using HMMs only and ultimately reached upwards of 95 % on their Hangul databases using heuristic structural constraints. The idea has been successfully extended to recognizing English words and ultimately to UniRec [11], the system accepting sentences of mixed languages based on a big circular network connecting several script recognizers.

#### NPenn++

NPenn++ was developed at University of Karlsruhe, Germany, and Carnegie Mellon University, USA, using a multistate time delay neural network [13]. It is actually a hybrid method combining features of neural network and HMM as well. For word recognition, a tree-structured dictionary is used along with a pruning technique for reducing the search space. For sentence recognition, there is another structure, a search tree that expands each time a letter is to be explored. Each node in the tree is assigned a letter HMM. The terminal nodes are linked back to the root node representing between-word space, thus modeling a sentence as a sequence of words separated by a space. It is reported that the system achieved 91~92 % hits for 20k vocabulary recognition tasks using their cursive datasets.

## Frog on Hand

Frog on hand has been developed using a number of modeling tools in pattern recognition [30]. Besides HMMs, this system employs a clustering technique and a dynamic time-warping algorithm. The developers highlighted a holistic combination of cluster generative statistical DTW (CSDTW) and the HMM-based method to treat handwriting variations. The recognition rate reached around 90 % accuracy using allegedly difficult UNIPEN dataset.

There are numerous other systems worth mentioning but omitted here due to limited space. In particular there are many good systems dealing with Chinese character recognition. But to our relief, there is a wonderful list of systems in the survey paper by Liu et al. [4] and Jaeger et al. [3].

With any of the known pattern recognition techniques, one can start from scratch to build a full-blown system with a complete set of functionalities. But, today, it would often be more desirable to use free or open sources that provide some or most of those functionalities mentioned here. Although often not guaranteed to work without bugs, you can, at least, get some help from any of the softwares listed below.

- **CellWriter** (<http://risujin.org/cellwriter/>)  
An open source program, designed to be writer-dependent, writing in cells
- **LipiTk** (<http://lipitk.sourceforge.net>)  
A general toolkit for online handwriting recognizer
- **Rosetta** (<http://www.handhelds.org/project/rosetta/>)  
A multistroke and full-word handwriting recognition software for X Window System

## Commercial Recognizers

There are several well-known products.

- **Graffiti** (<http://en.softonic.com/palm/text-entry-graffiti>)  
Graffiti is a single-stroke shorthand handwriting recognition system. It has been adopted as an input interface for PDAs based on the Palm OS. It has long been an object of a lawsuit from Xerox. Several variants are available.
- **Calligrapher** (ParaGraph, acquired by PhatWare, <http://www.PhatWare.com>)  
This software provides a multilingual support for Western European languages, available for Windows Mobile. It accepts all handwriting styles, print, cursive or mixed.
- **MyScript Notes** (Vision Objects, <http://www.visionobjects.com>)  
This software recognizes over 80 languages, including Russian, Chinese, Korean Hangul, Japanese, Cyrillic, and Arabic. It has been trained through millions of handwritten samples from native writers.
- **riteScript** (<http://www.ritescript.com>)  
It was developed by Evernote, a successor of Parascript's division. riteScript accepts both cursive and block letters and supports most of the functionalities such as vocabulary and non-vocabulary words and baseline detection in sentences.

## Conclusion

This chapter discussed many of the issues that need to be considered to develop a practical system for natural cursive handwriting recognition. Albeit brief, the discussion has been made broad enough to provide quick and helpful tips for designing diverse script recognizers with different characteristics. It is based on the belief that designers of different language script recognizers can benefit a lot from the techniques and solutions of other languages.

Signal processing often constitutes the core of many practical system development. Indeed the success of this step is a key to successful recognition. Nevertheless, there is no great theory other than traditional “good old” techniques, and heuristics and task-dependent insights abound here. Thus, the discussion has carefully been limited to a number of generic techniques. The extraction of features largely depends on the type of recognition algorithm employed. Thus, only a few generic issues are addressed including dynamic information and high-level shape features.

The lions’ share of this chapter has been devoted to the recognition methods. And this is the place where most ideas and theories have been developed. The discussion is around an online handwriting recognition method based on the HMM and explores the issues of modeling data variabilities and incorporating linguistic information. The use of the HMM is highly appropriate in that it is one of the most successful tools to date, and it often is a designer’s choice as it is simple to develop and easy to extend for complex patterns.

Finally this chapter cited a few research and commercial softwares, the list of which is far from being complete or comprehensive. In fact there are numerous other successful softwares claiming a large market share today. But only a handful of selected systems with historical importance and technical significance have been listed here. System developers may benefit a lot from case studies on those systems before designing their own recognizers.

A final remark is that the technology is highly mature and experiences are many. And the task that remained is a better and deeper understanding of the problems at hand – language and script characteristics, writing behaviors and styles, the type of applications, and so on – and a more extensive exploitation of a priori knowledge.

---

## Executive Summary

This chapter introduced roughly the techniques in online handwriting recognition. Those techniques are language and script-dependent. Therefore, a mere chapter would never be successful in digging out all the details. Instead this chapter has focused on understanding the differences and similarities across different scripts and then hard issues in developing real-world solutions. One script could be similar to some others linguistically or in pattern recognition perspective. So we can learn a lot by watching what other people are doing.

Since the introduction and following the rapid expansion of smartphones as a more attractive, personal device, the touch interface has become popular. The main interface, however, is still the keyboard, only just soft. But this does not mean that there is no place in those or other devices for a pen. In fact the pen is more convenient in many cases, such as large alphabet or large character set-based scripts, correcting errors or mistakes in texts, and in tasks extending hours or requiring more than touch.

During the past decades, the recognition algorithm has improved a lot. When typing errors and corrections are taken into account, pen-based text input could be faster or more accurate than the keyboard. The problem, if any, could be the interface. Most smartphone users are accustomed to keyboard, but not to stylus even though they have lived with lots of pens since childhood. Therefore, the real issue surrounding online handwriting is making the interface more intuitive and familiar.

Finally, the recognition engine technology has matured, if not finished, a lot after decades of research and development. Here again a real and more promising solution would be designing a “recombinant” system – combining existing techniques in an integrated framework while incorporating script-specific features and knowledge available.

---

## Cross-References

- ▶ [Asian Character Recognition](#)
- ▶ [Datasets and Annotations for Document Analysis and Recognition](#)
- ▶ [Handprinted Character and Word Recognition](#)
- ▶ [Middle Eastern Character Recognition](#)
- ▶ [Online Signature Verification](#)

---

## References

1. Tappert CC, Suen CY, Wakahara T (1990) The state of the art in on-line handwriting recognition. *IEEE Trans Pattern Anal Mach Intell* 12(8):787–808
2. Plamondon R, Srihari SN (2000) On-line and off-line handwriting recognition: a comprehensive survey. *IEEE Trans Pattern Anal Mach Intell* 22(1):63–82
3. Jaeger S et al (2003) The state of the art in Japanese on-line handwriting recognition compared to techniques in western handwriting recognition. *Int J Doc Anal Recognit* 6(2):75–88
4. Liu C-L et al (2004) On-line recognition of Chinese characters: the state of the art. *IEEE Trans Pattern Anal Mach Intell* 26(2):198–213
5. Al Emami S, Usher M (1990) On-line recognition of handwritten Arabic characters. *IEEE Trans Pattern Anal Mach Intell* 12(7):704–710
6. Lorigo LM, Govindaraju V (2006) Offline Arabic handwriting recognition: a survey. *IEEE Trans Pattern Anal Mach Intell* 28(5):712–724
7. Mezghani N, Mitiche A, Cheriet M (2008) Bayes classification of online Arabic characters by Gibbs modeling of class conditional densities. *IEEE Trans Pattern Anal Mach Intell* 30(7):1121–1131
8. Pal U, Chaudhuri BB (2004) Indian script character recognition: a survey. *Pattern Recognit* 37(9):1887–1899

9. Dongre V, Mankar V (2010) A review of research on Devanagari character recognition. *Int J Comput Appl* 12(2):8–15
10. Boccignone G, Chianese A, Cordella LP, Marcelli A (1993) Recovering dynamic information from static handwriting. *Pattern Recognit* 26(3):409–418
11. Jaeger S (1998) Recovering dynamic information from static, handwritten word images. PhD thesis, University of Freiburg, Foelbach
12. Sin B-K, Kim JH (1998) Network-based approach to Korean handwriting analysis. *Int J Pattern Recognit Artif Intell* 12(2):233–249
13. Jaeger S, Manke S, Reichert J, Waibel A (2001) Online handwriting recognition: the NPen++ recognizer. *Int J Doc Anal Recognit* 3:169–180
14. Sin B-K, Kim JH (1997) Ligature modeling for online cursive script recognition. *IEEE Trans Pattern Anal Mach Intell* 19(6):623–633
15. Bishop C (2006) Pattern recognition and machine learning. Springer, New York, p 740
16. Bahlmann C (2006) Directional features in online handwriting recognition. *Pattern Recognit* 39:115–125
17. Cho S-J, Kim JH (2004) Bayesian network modeling of strokes and their relationships for on-line handwriting recognition. *Pattern Recognit* 37:253–264
18. Chan K-F, Yeung D-Y (2000) Mathematical expression recognition: a survey. *Int J Doc Anal Recognit* 3(1):3–15
19. Meyers CS, Rabiner LR (1981) Connected digit recognition using a level-building DTW algorithm. *IEEE Trans Acoust Speech Signal Process ASSP-29*:351–363
20. Sakoe H (1979) Two-Level DP-matching – a dynamic programming-based pattern matching algorithm for connected word recognition. *IEEE Trans Acoust Speech Signal Process ASSP-27(6)*:588–595
21. Ney H (1984) The use of a one-stage dynamic programming algorithm for connected word recognition. *IEEE Trans Acoust Speech Signal Process ASSP-32*:263–271
22. Lee C-H, Rabiner LR (1989) A frame-synchronous network search algorithm for connected word recognition. *IEEE Trans Acoust Speech Signal Process* 7(11):1649–1658
23. Chen JW, Lee SY (1997) On-line Chinese character recognition via a representation of spatial relationships between strokes. *Int J Pattern Recognit Artif Intell* 11(3):329–357
24. Shin J-P (2002) Optimal stroke-correspondence search method for on-line character recognition. *Pattern Recognit Lett* 23:601–608
25. Plamondon R, Guerfali W (1998) The generation of handwriting with delta-lognormal synergies. *Biol Cybern* 78:119–132
26. Lee KF (1989) Automatic speech recognition: the development of the SPHINX system. Kluwer Academic, Boston
27. Seni G, Srihari RK, Nasrabadi N (1996) Large vocabulary recognition of on-line handwritten cursive words. *IEEE Trans Pattern Anal Mach Intell* 18(6):757–762
28. Hu J, Lim SG, Brown MK (2000) Writer independent on-line handwriting recognition using an HMM approach. *Pattern Recognit* 33:133–147
29. Lee J, Kim J (1997) A unified network-based approach for online recognition of multilingual cursive handwritings. In: Progress in handwriting recognition. World Scientific Publishing, pp 81–86
30. Bahlmann C, Burkhardt H (2004) The writer independent online handwriting recognition system frog on hand and cluster generative statistical dynamic warping. *IEEE Trans Pattern Anal Mach Intell* 26(3):299–310
31. Lee S-W (ed) (1999) Advances in handwriting recognition. Series in machine perception and artificial intelligence, vol 34. World Scientific, Singapore/River Edge, p 587
32. Liu Z-Q, Cai J-H, Buse R (2003) Handwriting recognition, soft computing and probabilistic approaches. Studies in fuzziness and soft computing, vol 133. Springer, New York, p 230
33. Impedovo S (2012) Fundamentals in handwriting recognition. Springer-Verlag, p 496
34. Doermann D, Jaeger S (eds) (2006) Arabic and Chinese handwriting recognition. In: SACH 2006, Summit, College Park. LNCS. Springer-Verlag, New York

- 
- 35. Su T (2013) Chinese handwriting recognition: an algorithmic perspective. Springer, Berlin/New York, p 139
  - 36. Mondal T (2010) On-line handwriting recognition of Indian scripts – the first benchmark. In: Proceedings of the international conference on frontiers in handwriting recognition, Kolkata, pp 200–205, Nov 2010

## Further Reading

There is a great body of literature on online handwriting recognition in numerous conference and workshop proceedings. And some of important technologies and results are well documented usually in the form of handbooks, such as those edited by Lee [31] and Liu et al. [32], while the most recent book on handwriting recognition methods will be the work by Impedovo [33].

The last decades saw a surge of research in Chinese and Arabic scripts. Interested readers may start their intellectual exploration with handbooks by Doerman and Jaeger [34] and Su [35]. On the other hand, many Indian scripts in the South Asian subcontinent seem to have thus far been defined as an offline character recognition problem. However, enthusiastic students and serious system designers may refer to any journal and conference papers such as the one on a benchmark test (2010) [36] to get a picture of the technology landscape in the continent.

Réjean Plamondon, Giuseppe Pirlo, and Donato Impedovo

## Contents

Introduction.....	918
Overview.....	919
Modeling and Description.....	920
Neuromuscular Representation.....	921
Data Acquisition and Preprocessing.....	922
Feature Extraction.....	924
Verification.....	926
Evaluation.....	931
Online Signature Verification: A Comparative Analysis.....	934
Conclusion.....	940
Description of Consolidated Software and/or Reference Datasets.....	942
Cross-References.....	942
References.....	942
Further Reading.....	947

---

## Abstract

Along with the exponential rise in the use of the Internet and of mobile devices is a rapidly growing need for personal identity confirmation. As a result, online signature verification is being considered with renewed interest.

This chapter presents an overview of online signature verification and a discussion of the most important theoretical aspects of signature generation and

---

R. Plamondon (✉)

Département de génie électrique, Ecole Polytechnique de Montréal, Montréal, QC, Canada

e-mail: [rejean.plamondon@polymtl.ca](mailto:rejean.plamondon@polymtl.ca)

G. Pirlo

Dipartimento di Informatica, Università degli Studi di Bari, Bari, Italy

e-mail: [pirlo@di.uniba.it](mailto:pirlo@di.uniba.it)

D. Impedovo

Politecnico di Bari, Bari, Italy

e-mail: [impedovo@gmail.com](mailto:impedovo@gmail.com)

modeling. The principal issues related to the development of online signature verification systems are addressed, and the most commonly used approaches proposed in the literature are presented and compared.

Finally, throughout the chapter, based on the state-of-the-art techniques available, current challenges are identified and promising directions for further research are highlighted.

### Keywords

Biometrics • Kinematic theory • Logical and physical access control • Log-normal models • Neuromuscular systems • Online signature verification • Personal identification • Security • Signature generation • System evaluation

## Introduction

Biometrics is an emerging field of research and technology that involves the recognition of individuals through their physical or behavioral traits. Examples of physical attributes are fingerprints, facial features, the iris, and DNA. Some behavioral characteristics are the signature, the voice, and keystroke dynamics, among others.

Handwritten signatures occupy a very special place in biometrics. A signature is a biometric trait generated by a complex process originating in the signer's brain as a motor control "program," instantiated through the neuromuscular system and left on the writing surface by a handwriting device.

The net result is that automatic signature verification is a multidisciplinary research field involving aspects of disciplines ranging from human anatomy to engineering and from neuroscience to computer and system sciences. The fundamental research issues in biometrics are described in two comprehensive surveys that report the progress in automatic signature verification specifically: up to 1989 in one [78] and up to 1993 in the other [47]. In 1994, a special journal issue and a book summarizing the most relevant research activities were published [74]. The increasing number of research efforts in this field, up to 2008, has also been summarized in a recent survey [37].

Today, in the era of the networked society, the automatic confirmation of personal identity is a fundamental requirement, and the need to integrate signature verification technologies into other standard equipment for a wide range of applications is increasing rapidly. In fact, the verification of a person's identity by signature analysis does not involve an invasive measurement procedure, and people have long agreed to the use of signatures in their daily lives. Furthermore, handwritten signatures are a long been established means of personal identification, and their use is widespread and well recognized by administrative and financial institutions. Consequently, several national associations and international bodies have recently supported the standardization of signature data interchange formats, and specific rules and regulations on signature-based personal identity verification have been approved in many countries [2, 39].

This chapter, which provides an overview of the field of online signature verification, is organized as follows: section “[Overview](#)” presents an outline of the signing process, along with the main aspects of signature verification. Section “[Modeling and Description](#)” deals with the modeling and description of signatures. A model of the neuromuscular system that underlies the generation of signatures is introduced, and then the problems related to signature description for verification purposes are addressed, focusing on both the signature acquisition and feature extraction steps. Section “[Verification](#)” describes the main verification techniques. Certain aspects of verification system evaluation are discussed in section “[Evaluation](#),” and a comparative analysis of the state-of-the-art systems for online signature verification is presented in section “[Online Signature Verification: A Comparative Analysis](#). ” A brief summary of the key issues raised in the chapter is provided in the conclusion, as well as an indication of the most valuable research directions for the field in the future.

---

## Overview

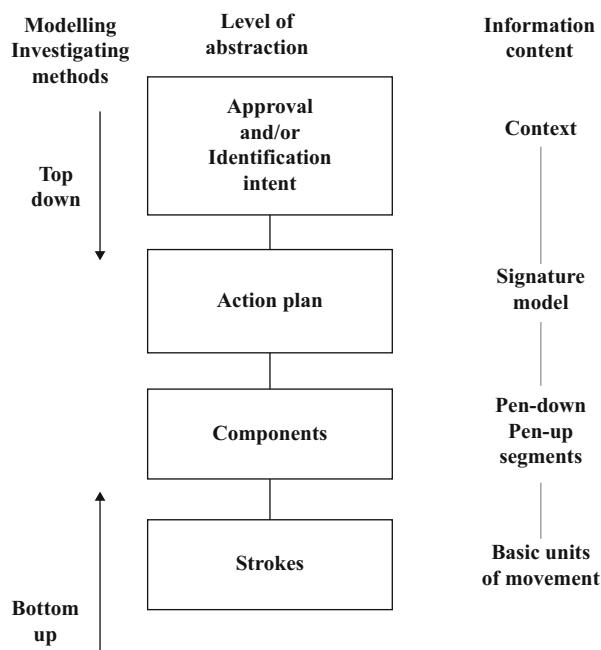
The signature is quite different from the other identification modalities that can be used as potential candidates for biometric applications because it is a behavior-based identifier. A signature is a rapid movement that has been defined, learned, and practiced over a period of years, in literate populations, to become a person’s characteristic identifying pattern. The distinct imprint that is left on a surface is the outcome of a complex generation process that occurred while that person was engaged in the signing process. To design an efficient online verification system, several aspects have to be taken into account. Figure 27.1 schematizes the entire verification process.

The first aspect to take into account is the context: What is the intention of the subject producing a signature? The usual context is an approval procedure, but this might not always be the case. Depending on the signer’s intentions, a signature instance will be either a genuine gesture or a forgery. In the first case, it can be the result of an honest action in response to an identity challenge to gain access or get an authorization, a compelled response to a threat or a gesture that the author is planning to deny later on. In the second, the false specimen can indicate an attempt to deceive or a random error. The initial challenge is to design a system capable of operating under these various conditions and ideally of recognizing them.

Whatever the motivation for the signature, the signer has to access an action plan, which is a spatiotemporal map that has been memorized and stored in the motor cortex of the brain. This plan is an activation sequence that represents a signature model. It is instantiated by signature “components,” which represent portions of a trajectory between pen-up and pendown movements. Each component is itself made up of a sequence of strokes that are considered as the basic units of human handwriting movements.

In other words, every time a person writes his signature, whatever the context of his action, he has to recall a map of the overall gesture from his motor memory,

**Fig. 27.1** A model of the signature generation process in terms of the information content and the level of abstraction that have to be taken into account in the design of a verification system



which is a premeditated image. Depending on the complexity of this map, which can be considered as a “motor program,” this image will be instantiated in a fairly large number of pieces, separated by pen-up and pen-down links. Each of these pieces, or “chunks,” is itself the outcome of activating a sequence of strokes.

The overall process can be modeled and studied using a top-down or a bottom-up approach. In the former case, the gesture is first analyzed as a whole to extract global features and then on smaller portions of the movement to extract local features, at various levels of representation. In the latter case, the analysis starts with local features and a more global representation is built up by combining the initial representations. These two methodologies lead to various algorithms and techniques aimed at designing a successful verification system, as shown below.

## Modeling and Description

As explained above, a handwritten signature is the result of a complex generation process, and suitable schemes and strategies must be defined to model and describe signatures for the design of an online signature verification system. This section describes the Sigma-Lognormal model, which is the general model underlying the signature generation process. The problem of signature acquisition is addressed step by step. Finally, some important strategies related to signature description are presented.

## Neuromuscular Representation

To better illustrate the signature generation process, Fig. 27.2 schematically shows the production of a typical specimen, according to the Sigma-Lognormal model [64]. This model is invariant with respect to cultural or language differences and can be applied to any type of signature: Asian, Arabic, European, or American. This illustration shows that the action plan is made up of virtual targets, each pair of them linked to an arc of a circle. This map represents a sequence of discontinuous strokes. The plan involves activating a motor command generator which produces a series of impulses that activates a neuromuscular system which is itself characterized by its lognormal impulse response [76]. For each impulse, a lognormal velocity profile is generated at the pen tip, and the temporal superimposition of these strokes results in a smooth well-controlled signature. According to this model, strokes are hidden in the signal, which makes segmentation a very difficult and often unreliable process.

Specifically, the Sigma-Lognormal model technically considers the velocity of the pen tip as the result of an action of the neuromuscular system, as described by a vectorial summation of lognormal primitives:

$$\vec{v}(t) = \sum_{i=1}^N \vec{v}_i(t; t_{0i}, \mu_i, \sigma_i^2) = \sum_{i=1}^N D_i \begin{bmatrix} \cos(\theta_i(t)) \\ \sin(\theta_i(t)) \end{bmatrix} \Lambda_i((t; t_{0i}, \mu_i, \sigma_i^2); N \geq 2) \quad (27.1)$$

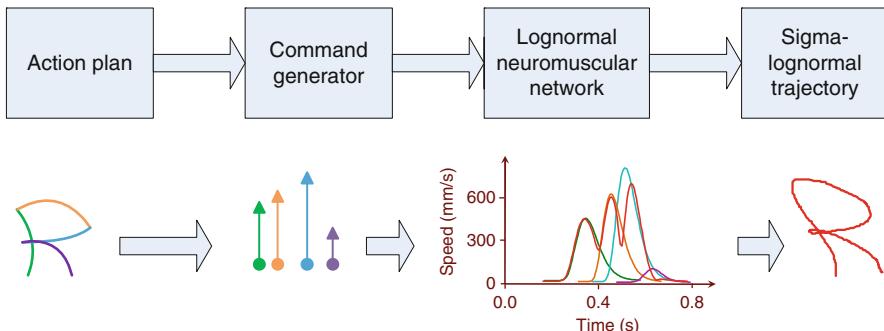
As expressed in this equation, each lognormal defines a stroke which is scaled by a command parameter ( $D$ ) and time shifted by the time occurrence of this command ( $t_0$ ), the overall stroke pattern being described by a lognormal time function:

$$\Lambda(t; t_0, \mu_i, \sigma_i^2) = \frac{1}{\sigma_i \sqrt{2\pi} (t - t_0)} \exp \left( \frac{-[\ln(t - t_0) - \mu_i]^2}{2\sigma_i^2} \right) \quad (27.2)$$

Each of these strokes also occurs along a pivot, and the angular position of the pen tip can be calculated using an error function (erf) (27.3):

$$\theta_i(t) = \theta_{bi} + \frac{(\theta_{ei} - \theta_{bi})}{2} \left[ 1 + \text{erf} \left( \frac{\ln(t - t_{0i}) - \mu_i}{\sigma_i \sqrt{2}} \right) \right] \quad (27.3)$$

where  $\theta_{bi}$  and  $\theta_{ei}$  refer to the beginning and ending angular direction of each stroke, respectively. In these equations,  $\mu_i$  and  $\sigma_i$  represent the log time delay and the log response time of the neuromuscular system, respectively, as it reacts to the  $i$ th command [77]. Finally, the synergy emerging from the interaction and coupling of many of these neuromuscular systems results in the sequential generation of a complex signature pattern as depicted in Fig. 27.2.



**Fig. 27.2** The production of a signature according to the Sigma-Lognormal model: illustrative sketches

Below, the main features of this model are used as a framework to present the various sections of our survey and to point out the major difficulties that have to be overcome by researchers in designing a reliable online signature verification system.

## Data Acquisition and Preprocessing

Data acquisition devices for online signatures generate a spatiotemporal representation of the signature. A wide variety of data acquisition devices is available, such as electronic pens capable of detecting position, velocity, acceleration, pressure, pen inclination, and writing forces and digitizing tablets with digital ink technologies providing immediate feedback to the writer [90]. Other approaches do not require the use of a special stylus, using instead a video camera focused on a user writing on a piece of paper with an ordinary pen [57].

Other methods use computer vision techniques to capture handwriting. For instance, a special stylus containing a small CCD camera that captures a series of snapshots of the writing process has recently been proposed. This system recovers the whole handwritten trace by analyzing the sequence of successive snapshots. The stylus is also equipped with a stress sensor for detecting the pressure applied on the ballpoint and determining the pen-up/pen-down information [59].

In addition, a wired-glove device for virtual reality applications has been used for online signature verification. This device can provide data on both the dynamics of the pen motion during signing and the individual's hand shape [41].

Of course, interoperability problems arise when signature verifiers are required to work with specimens acquired from a multitude of diverse input devices, ranging from a mouse to input devices for mobile computing [36]. In particular, signature

verification on handheld devices is assuming ever-greater practical relevance, since PDAs and smartphones are increasingly pervasive in our daily lives, as they allow access to a multitude of new applications and services. Of course, it is worth noting that signature verification on handheld devices raises specific issues that do not apply in other scenarios. In particular, handheld devices have a very small pen input area and generally provide poor sampling quality. In addition, the input device captures only position information, and the signer has to use a touch screen instead of paper and a stylus instead of an ordinary pen [56]. Finally, the basic underlying hypothesis of considering a signature as a well-learned movement might not be fully appreciated from the start with these new devices. Degradation in the verification performance in the first tests could then be expected, as well as the need to update more frequently the reference signatures, as a user learning curve evolves.

After data acquisition, the input signals can be enhanced using standard filtering and noise reduction algorithms, based on the Fourier transform, as well as normalization techniques to standardize signatures in terms of position, size, orientation, and duration. This is followed by signature segmentation, which is a critical task, since it strongly determines the elements of the signature model and influences all the successive phases of signature verification [37, 43, 102, 104].

Some segmentation techniques derive from specific characteristics of handwritten signatures and reflect specific handwriting models. For instance, a signature can be considered as a sequence of writing units, delimited by abrupt interruptions. Writing units are the “regular” parts of the signature, while interruptions are the “singularities” of the signature. Thus, as explained in section “[Overview](#),” pen-up/pen-down signals can be used to segment a signature into components, as can the analysis of curvilinear and angular velocity signals [37, 76, 78]. Another way to segment a signature is to use its perceptually important points. The importance of a point depends on the extent of change in the writing angle between the selected point and its neighbors [6]. A modified version of this technique considers the end points of pen-down strokes as significant splitting points. Other approaches use perceptually important points for segmenting signatures while considering the evolutionary distance measure, based on arc-length distance, for segment association [37, 84].

In some cases, segmentation techniques are based on splitting strategies supporting specific signature verification approaches. For example, with Dynamic Time Warping (DTW), two or more signatures can be segmented into the same number of segments that correspond more or less perfectly. After the first signature has been split, based on uniform spatial criteria or on the position of geometric extremes, DTW is applied to determine the corresponding set of points on other specimens [12, 49]. A model-guided segmentation technique has also been proposed, which uses DTW to segment an input signature based on its correspondence to the reference model [37]. Again, all these techniques provide “operational strokes” that constitute a fairly reliable estimation of the real strokes that are hidden in the signal (see Fig. 27.2).

**Table 27.1** Function features

Functions	References
Position	Y. Hongo et al. [29], Y. Komiya et al. [46], J. Ortega-Garcia et al. [68], R. Plamondon [75], and Q.-Z. Wu et al. [96]
Velocity	V. Di Lecce et al. [10, 11], M. Fuentes et al. [18], K. Huang and H. Yan [32], A. K. Jain et al. [40], J. Ortega-Garcia et al. [68], R. Plamondon [75], T. Qu et al. [81], C. Schmidt and K.-F. Kraiss [83], and Q.-Z. Wu et al. [95]
Acceleration	C. Schmidt and K.-F. Kraiss [83]
Pressure	Y. Hongo et al. [29], K. Huang and H. Yan [32], Y. Komiya et al. [46], J. Ortega-Garcia et al. [68], T. Qu et al. [81], and C. Schmidt and K.-F. Kraiss [83]
Force	R. Martens and L. Claesen [55] and D. Wang et al. [90]
Direction of pen movement	M. Fuentes et al. [18] and J. J. Igarza et al. [34]
Pen inclination	J. J. Igarza et al. [34], Y. Komiya et al. [46], R. Martens and L. Claesen [55], and J. Ortega-Garcia et al. [68]

## Feature Extraction

Function and parameter features can be used for online signature verification. When function features are used, the signature is characterized by a time function, the values of which constitute the feature set. When parameter features are used, the signature is characterized as a vector of parameters, each of which represents the value of one feature [78].

As Table 27.1 shows, position, velocity, and acceleration functions are widely used for online signature verification. The position function is conveyed directly by the acquisition device, whereas the velocity and acceleration functions can be provided by the acquisition device and numerically derived from position [10, 96]. In recent years, pressure and force functions have frequently been used, and specific devices have been developed to capture these functions directly during the signing process [90]. The directions of the pen movements and the pen inclinations have been also considered [34, 68]. In fact, a comparative study of the consistency of features for online signature verification has demonstrated that the position, velocity, and pen inclination functions can be considered to be among the most consistent, when a distance-based consistency model is applied. This model starts by considering that the characteristics of a feature must be estimated by using the distance measure associated with the feature itself [50]. More recently, personal entropy has been used to assess the robustness to short-term and long-term variability of the position, pressure, and pen inclination functions. The results show that position is the most robust to time variability. Pressure is not recommended in the long-term context, although it may give better performance results in the short-term context. This latter result is consistent with those of a study dealing with pressure control in handwriting [84] where it is shown that subjects can be classified into three groups with regard to the control of pressure: those who are consistent controllers throughout the script, those who are consistent during some specific time windows,

**Table 27.2** Parameter features

Parameters	References
Total signature duration	R. S. Kashi et al. [43], J. Lee et al. [49], L. L. Lee et al. [48], W. Nelson et al. [63], R. Plamondon [75], and T. Qu et al. [81],
Pen-down time ratio	R. S. Kashi et al. [43] and W. Nelson et al. [63].
Number of pen-ups/pen-downs	J. Lee et al. [49], L. L. Lee et al. [48], and T. Qu et al. [81]
Direction-based	R. S. Kashi et al. [43], J. Lee et al. [49], W. Nelson et al. [63], T. Qu et al. [86], and M. Zou et al. [104]
Curvature-based	A. K. Jain et al. [40]
Moment-based	R. S. Kashi et al. [43]
AVE/RMS/MAX/MIN of posit., displ., speed, accel.	M. Fuentes et al. [18], R. S. Kashi et al. [43], M. A. Khan et al. [44], J. Lee et al. [49], L. L. Lee et al. [48], W. Nelson et al. [63], and T. Qu et al. [81].
Duration of positive/negative posit., displ., speed, accel.	R.S. Kashi et al. [43], J. Lee et al. [49], L.L. Lee et al. [48], and W. Nelson et al. [63].
X-Y correlation of posit., displ., speed, accel.	M. Fuentes et al. [18], R. S. Kashi et al. [43], W. Nelson et al. [63], and R. Plamondon [75]
Fourier transform	G. Dimauro et al. [13] and Q. Z. Wu et al. [95, 97].
Wavelet transform	D. Letjman and S. George [51], R. Martens and L. Claesen [55] and I. Nakanishi et al. [60].

and those who apparently have no reliable control. This kind of investigation has found that pen inclination is a very unstable feature [30], contrary to the results reported in [50]. In fact, some of these results are consistent with the bottom-line conclusion in Fig. 27.2 to the effect that the pen tip velocity is the basic control variable exploited by the central nervous system for the generation of a signature.

Table 27.2 lists some major parameter features for online signature verification. Total signature duration [43, 48, 81], the pen-down/pen-down time ratio [43, 63], and the number of pen lifts (pen-down, pen-up) [48, 49, 81] are some of the parameters most often used for signature verification. Other parameters are derived from the analysis of direction, curvature, and moments of the signature trace. Parameters can also be numerically derived from the representative time functions of a signature, like the average (AVE), root mean square (RMS), maximum (MAX), and minimum (MIN) values of position, displacement, speed, and acceleration [48, 63, 81]. Parameters derived from correlations and time-dependent relations between function features can also be considered [43, 48, 49, 63], as can coefficients derived from Fourier [13, 95, 97] and Wavelet [51, 55, 60] transforms. Concerning the variability of parameter features, a recent study shows that the long-term temporal variability of online signatures is very high although no typical evolution occurs over time. Furthermore, since inter-session variability is very high, systems can give less accurate results in real-time applications if they are trained using data from only one acquisition session [36, 78].

It is worth noting that depending on the signature model, features can be derived at the global or the local level. The global features reflect the holistic characteristics of the signature action plan, and the local ones highlight some very specific and personal patterns in the instantiation of this plan. When function features are considered, they can be defined for the whole signature or for individual signature segments. Similarly, two kinds of parameter features are considered in the literature: global and local. Typical global parameters are the total duration, number of pen lifts, number of components, and global orientation of the signature. Local parameters are related to features derived from specific parts of the signature, so that direction-based, curvature-based, and moment-based parameters are generally considered at the local level. Even when coefficients obtained by mathematical transforms are considered as parameter features, they can be extracted at the global or local level, depending on the signature model that has been adopted [24, 80].

Of course, the use of a universally applied feature set is not effective for signature verification, since signatures from different writers generally contain very few common characteristics. The knowledge that an individual's signature is unique has led many researchers to devote special attention to the selection of the most suitable features to use in signature validation, for genuine signatures as well as for random and skilled forgeries. The basis for this approach is the evidence that some features are well suited to distinguishing skilled forgeries from genuine signatures, while others are better at distinguishing random forgeries. Genetic Algorithms have also been used for parameter selection, and in some cases the feature set is personalized by assigning a different weight to each feature, as well as selecting the optimal prototype function. Sometimes, the most valuable function feature for verification is selected for each signature segment based on the specific characteristics of that segment. For instance, information on the local stability of a signature can be derived from different representation domains and can be used for selecting the best function feature [38].

---

## Verification

In the verification process, the authenticity of the test signature is evaluated by matching its features against those stored in the knowledge base developed during the enrolment stage. The result for the authenticity of a signature is provided as a Boolean value. In some systems, uncertainty is incorporated into the decision process. When this occurs, the signer is required to sign again, and the system generally increases its acceptance thresholds, to make the success of this second attempt more difficult to achieve.

As Table 27.3 shows, different techniques can be considered for signature comparison. In terms of distance-based verification techniques, Dynamic Time Warping (DTW) has been used extensively with function features, since this allows the time axis of two time sequences that represent a signature to be compressed or expanded locally, in order to obtain the minimum of a given distance value [37, 72]. The fact that a comparison between the complete time sequences results in higher

**Table 27.3** Comparison techniques

Technique		References
Dynamic Programming		J. Lee et al. [49] and I. Nakanishi et al. [60]
Dynamic Time Warping (DTW)	Continuous	L. Bovino et al. [4], V. Di Lecce et al. [10, 11], G. Dimauro et al. [12, 14], K. Huang and H. Yan [32], M. E. Munich and P. Perona [57], and R. Plamondon [75]
	GA-based	M. Wirotius et al. [92]
	PCA-based	A. Kholmatov and B. Yanikoglu [45] and B. Li et al. [52]
	MCA-based	B. Li et al. [52]
	PA-based	M. Wirotius et al. [92]
	EP-based	H. Feng and C.C. Wah [17]
	Random-based	M. Wirotius et al. [92]
	Asymmetric	R. Martens and L. Claesen [55]
Correlation		M. Parizeau and R. Plamondon [72] and R. Plamondon [75]
Split and merge		Q. Z. Wu et al. [96]
String/graph/tree matching		Y. Chen and X. Ding [7], A. K. Jain et al. [40], and M. Parizeau and R. Plamondon [72]
Structural description graph		G. Dimauro et al. [13] and K. Huang and H. Yan [32]
Euclidean distance		M. A. Khan et al. [44].
Mahalanobis distance		R. Martens and L. Claesen [55] and K. Zhang et al. [103]
Membership functions		T. Qu et al. [81]
Dynamic similarity measure		Q. Z. Wu et al. [97]
Support vector machine (SVM)		M. Fuentes et al. [18], M. T. Ibrahim et al. [33], and A. Kholmatov and B. Yanikoglu [45]
Hidden Markov Models (HMM)	Left-to-Right topology	J. G. A. Dolfing et al. [16], M. Fuentes et al. [18], J. Galbally et al. [19, 20], J. J. Igarza et al. [34, 35], R. S. Kashi et al. [43], M. Martinez-Diaz et al. [56], J. Ortega-Garcia et al. [68], M. M. Shafeei and H. R. Rabiee [85], B. Van et al. [87], T. Wessels and C. W. Omlin [91], H. S. Yoon et al. [101], and M. Zou et al. [104]
	Ergodic topology	T. Wessels and C. W. Omlin [91]
Neural network (NN)	Multilayer perceptrons (MLP)	M. Fuentes et al. [18] and K. Huang et al. [31]
	Backpropagation network (BPN)	D. Z. Letjman and S. E. George [51]
	Self-organizing map	T. Wessels and C. W. Omlin [91]

computational load has led to the development of data reduction techniques based on Genetic Algorithms (GA) Principal Component Analysis (PCA), Minor Component Analysis (MCA), and Linear Regression (LR), among others [37]. Asymmetric DTW has also been defined to avoid deformation of the reference signatures when they are matched against test specimens [55]. Similarity measures [97], split-and-merge strategies [96], and string matching [7] have also been considered for signature comparison. Other techniques use the Mahalanobis distance [55, 103], the Euclidean distance [44], or membership functions [81] for matching the target specimen to the parameter-based model of the declared signer's signatures. Support vector machines (SVMs) are another effective approach for online signature verification, since they can map input vectors to a higher dimensional space, in which clusters may be determined by a maximal separation hyperplane [18, 45].

Of the model-based techniques, Hidden Markov Models (HMM) have been found to be well suited to signature modeling. They are highly adaptable to personal variability [18, 56, 87] and yield superior results to other signature modeling techniques [32]. Although both Left-to-Right (LR) and Ergodic (E) topologies have been used for online signature verification, the former is considered best suited to signature modeling [34, 91, 104]. Neural networks have also been considered for signature verification, since they have demonstrated good generalization capabilities. In this group, multilayer perceptrons (MLP), time-delay neural networks, backpropagation networks, self-organizing map, and radial basis functions (RBF) have been used [18, 31, 51, 91]. Of course, the use of neural networks for online signature verification is strongly limited by the fact that they generally require large amounts of learning data, which are not available in many current applications [37, 47].

Whatever technique is adopted, the need to improve verification performance has led researchers to investigate multi-expert approaches for combining sets of verifiers based on global and local strategies [61, 62] and through boosting methods [29]. So far, several multi-expert systems have been proposed in the literature, based on abstract-level, ranked-level, and measurement-level combination methods as well as on serial, parallel, and hybrid topologies. In particular, multi-expert approaches have been found to be well suited for implementing top-down and bottom-up signature verification strategies [80]. Concerning bottom-up strategies, it is worth noting that multi-expert approaches allow effective implementation of a stroke-based signature verification approach, in which a signature is verified starting with an analysis of its basic elements. Furthermore, this approach can lead to lower error rates, compared to global approaches, since a large amount of personal information is conveyed in specific parts of the signature and cannot be detected when the signature is viewed as a whole [6, 12, 13, 83]. Furthermore, verification at stroke level can be performed by DTW, also considering multiple function features for stroke representation (like position, velocity, and acceleration), in order to verify both the shape and dynamics of each part of the signature [4, 11].

Concerning the use of the multi-expert approach in top-down verification strategies, the hybrid topologies are capable of achieving the performance advantages of serial approaches in quickly rejecting very poor forgeries while retaining

the reliability of parallel combination schemes. This is the case for multilevel verification systems: for example, those first verify the structural organization of a target signature and then analyze in detail the authenticity of its individual elements [13, 32].

Along with the matching techniques, attention has been paid to knowledge-based structures in relation to signature modeling techniques. Two approaches can be considered for signature verification: writer dependent and writer independent. When writer-dependent approaches are used, a specialized classifier is trained for each writer, using only genuine specimens. In this case, one approach uses a single prototype of a genuine signature for each writer. Several techniques have been proposed for the development of the optimal average prototype for a signer, including shape and dynamic feature combination, time- and position-based averaging, and selecting the genuine specimen with the smallest average difference when compared to the other true signatures available. Once the prototype has been determined, the decision threshold is generally defined on the basis of the distances between genuine signatures [37]. In another approach, a set of genuine signatures is used for reference. In this case, the selection of the optimal subset of reference signatures from among the specimens available is performed either on the basis of an analysis of variance within samples [26, 27] or by considering the high-stability regions in the signatures [9, 10]. The selection of the best subset of reference signatures can be avoided but at the expense of using multiple models for signature verification. The writer-independent approach uses a single classifier for all writers, which is trained using genuine and forged specimens of the entire population of individuals considered for training. The writer-independent approach is generally considered superior to the writer-dependent approach, when a limited number of reference signatures is available for each author. This is because the writer-independent classifier can be trained from previously collected specimens of other individuals [37].

It is important to note that knowledge base development and management requires handling multifaceted aspects related to signature type, complexity, and stability. For instance, short signatures could convey less information than long signatures, resulting in less accurate verification results [5]. Similarly, people with common names could be more likely to share similar signatures with other individuals – at least in terms of shape characteristics. In both cases, the system should be capable of adapting to the characteristics of the individuals enrolled. Furthermore, the complexity of a signature has been quantified by estimating the difficulty of imitating it, which corresponds to the estimated difficulty of perceiving, preparing, and executing each stroke of the signature itself [5]. Concerning signature stability, a local stability function can be obtained using DTW to match a genuine signature against other authentic specimens [8, 14, 32]. In this case, every match attempt is used to identify the *direct matching points* (DMPs), which are the unambiguously matched points of the genuine signature that indicate the presence of a small, stable region of the signature, in other words chunks of the action plan that are the best learned. The local stability function of a point of a genuine signature is then computed as the average number of times the point is a DMP, when an attempt

is made to match the signature against other genuine specimens. Furthermore, the use of an analysis of local stability to measure short-term modifications – which depend on the psychological condition of the writer and on the writing conditions allows selection of the best subset of reference signatures and the most effective feature functions for verification objectives while providing useful information to weight the verification decision obtained at the stroke level [10, 32]. Long-term modifications depend on the alteration of the physical writing system of the signer at the effector level, finger, hand and arm as globally reflected by the logtime delays and logresponse times, as well as on the modification of the motor program in his or her brain (as expressed in terms of the five Sigma-lognormal command parameters). Overall, signature variability is affected more by fluctuations of the parameters associated with the central neural coding than the peripheral parameters reflecting the timing properties of the muscular system activated by the action plan [15].

When these modifications are evaluated, useful information can be gathered for updating the reference signature model by including additional information from new signatures as it becomes available. Other types of approaches estimate the stability of a set of common features and the physical characteristics of signatures to which they are most closely related, in order to obtain global information on signature repeatability, which can be used to improve a verification system [25, 26]. In general, these approaches have shown that there is a set of features that remains stable over long periods, while there are other features that change significantly with time, as a function of signer age. This is the case with features related to total execution time, velocity, and acceleration [25]. Since intersession variability is one of the most significant causes of verification performance deterioration, specific parameter-updating approaches have been considered [37]. A client-entropy measure has also been proposed to group signatures in categories depending on signature complexity and variability. Also, the measure based on local density estimation by an HMM can be used to ascertain whether or not a signature contains enough information to be successfully processed by a verification system [30].

Of course, although the general model underlying the signature generation process is invariant in terms of cultural habits and language differences among signers, the enormous diversity in the signatures of people from different countries has suggested the development of specifically designed solutions. For instance, Western-style signatures generally consist of signs that could form concatenated text combined with pictorial strokes. In some countries, the habit is to sign with a readable written name, whereas in other countries signatures are not always legible. Many more differences can be expected when considering signatures written by people from non-Western countries. To address these differences, specific approaches have been proposed in the literature for Chinese and Japanese signatures, which can consist of independent symbols, and for Arabic and Persian signatures, which are cursive sketches that are usually independent of the person's name. In general, as the need for cross-cultural applications increases, it is becoming more and more important to evaluate both the extent to which personal background affects signature characteristics and the accuracy of the verification process. For this reason, a set of metadata, sometimes called "soft biometrics," has been also considered.

---

Metadata are related to various aspects of a writer's background, such as nationality, script language, age, gender, and handedness. Some metadata can be estimated by statistically analyzing human handwriting, which means that it is possible to adapt signature verification algorithms to a particular metadata context in order to improve verification performance [94].

Security and privacy issues play a fundamental role in knowledge base development and management, as in all phases of the online signature verification process. In fact, identity theft can occur if biometric traits are stolen by a cyber-attacker, and sensitive information about a person's personality and health can be revealed. One advantage of the signature in this dramatic context is that it can be changed by an individual by simply changing the pattern of his or her signature – and changed altogether by that individual in the case of identity theft. It must be remembered however that more practiced movements lead to better performances. Also full-length signatures generally outperform shorter versions, like initials or handwritten passwords, although a password can be kept secret and changed regularly [71].

To summarize, the security and privacy of biometric data is a multifaceted field of growing interest. Valuable results have already been achieved in the field of cryptography in cryptographic key generation, for example [86], and in cancellable biometrics, where template protection has been accomplished by applying an intentional and repeatable modification to the original signature data, which should make it impossible, or computationally unfeasible, to obtain the original raw biometric data from the stored secure template [54].

---

## Evaluation

Online signature verification systems are usually evaluated by analyzing their ability to accept genuine signatures and to reject forgeries. For this purpose, two types of error rates are considered: the type I error rate, which concerns the false rejection of genuine signatures (FRR – false rejection rate), and the type II error rate, which concerns the false acceptance of forged signatures (FAR – false acceptance rate). The equal error Rate (EER) has been widely considered as a measure of the overall error of a system. It is defined as the system error rate when  $FRR = FAR$ . More recently, the receiver operating characteristic (ROC) curve has also been considered for evaluating signature verification systems. Indeed, the ROC curve, which plots the FRR vs. the FAR, has several useful properties. The most important is the area under the curve (AUC) of the ROC, which can be used to estimate system performance by using a single value, since the AUC provides the probability that the classifier will rank a randomly chosen positive sample higher than a randomly chosen negative sample [47, 78].

However, it is worth noting that notwithstanding the numerous measures that have been proposed so far, performance evaluation still remains a very critical and complex task [78, 79]. In particular, FAR evaluation is difficult and generally imprecise, since the existence of skilled forgeries for a given signature is not certain, nor is the possibility of collecting good-quality forgery samples for the test. In fact,

forgery quality strongly depends on the type and amount of information provided to the forger, as well as his or her training, effort, and ability [3]. In order to address this problem, different classes of forgeries have been considered [78]: *random* forgeries, in which the forger uses his/her own signature instead of the signature to be tested; *simple* forgeries, in which the forger makes no attempt to simulate or trace a genuine signature; and *freehand* or *skilled* forgeries, in which the forger tries and practices imitating as closely as possible the static and dynamic information of a genuine signature. Another attempt to grade forgery quality considers the following four categories [37]: an *accidental* forgery, which involves the use of arbitrary, non-authentic writing samples against some other reference; a *blind* attacker, who is a forger who only has a textual knowledge about the writing content; a *low-force* forgery, which occurs when the forger is in possession of an off-line representation of the signature image; and a *brute-force* attacker, who is a forger who has had the opportunity to observe the dynamics of the writing process of the victim. Of course, as demonstrated in a recent experiment, a significant degradation in the verification of online signatures can be observed as the quality of the forgeries that are used for the test increases [1].

In addition, the comparative assessment of the approaches proposed in the literature needs large, public signature databases and widely accepted protocols for experimental tests [70, 73]. Concerning public databases, it is worth noting that the development of a benchmark signature database is a time-consuming and expensive process. It involves not only scientific and technical issues, like those related to the statistical relevance of the population of individuals involved, as well as acquisition devices and protocols but also legal aspects related to data privacy and intellectual property rights. In contrast, since the development of benchmark databases is rightly recognized as a key aspect of the success and expanded usage of signature-based verification systems, specific efforts have recently been made to develop both unimodal benchmark databases (i.e., that contain only a single biometric trait of an individual) and multimodal ones (i.e., that contain two or more biometric traits of an individual). The Philips database [16, 73] contains 1,530 genuine signatures from 51 individuals and 3,200 forgeries. In this case, the acquisition process was carried out at a sampling rate of up to 200 Hz, and information on position, pressure, and “pen tilt” in the x and y directions were captured. This database divides forgeries into the following three categories: “over the shoulder,” i.e., forgeries captured by the forger after seeing the genuine signature being written; “home improved,” i.e., forgeries captured by the forger after having the opportunity of practicing the signature in private; and “professional,” i.e., high-quality forgeries produced by experts. The BIOMET multimodal database [24] contains 2,201 signatures from 84 individuals: 1,252 genuine specimens and 949 forgeries. These signatures were acquired using an ink pen on a digitizing tablet at a sampling rate of 100 Hz. Information on position, pressure, and pen orientation (azimuth and altitude) are provided. Data acquisition occurred in two separate sessions. The MCYT database [69] is a bimodal database containing 16,500 online signatures, 8,250 of them genuine (produced by 330 individuals) and 8,250 skilled forgeries. Data acquisition was performed by a digitizing tablet at 100 Hz. The tablet captured the following information on the

acquired signatures: position, pressure, and pen orientation (azimuth and altitude). The SVC2004 database [100] contains data from 1,600 signatures, 800 of them genuine specimens (produced by 40 individuals) and 800 forgeries. A 100 Hz sampling rate tablet was used for data acquisition. In order to help the forgers practice the signatures before producing the forgeries, a computer program was provided to them to enable them to visualize the writing sequence of the signature to be forged on a monitor. Information on position, pressure, pen orientation (azimuth and altitude), and pen-up/pen-down signals were recorded for each specimen. The BioSecure multimodal database [70] has two relevant datasets of online signatures: sub-corpus 2 (DS2) and sub-corpus 3 (DS3). DS2 contains data from 667 individuals acquired in two separate sessions using an ink pen on a digitizing tablet at a sampling rate of 100 Hz. The information obtained from the tablet included position, pressure, and pen orientation (azimuth and altitude). For each individual, 25 signatures (15 of them genuine and 10 forgeries) were collected in each session. DS3 contains the signatures of about 713 individuals. Signatures were acquired using a PDA at a frequency of 100 Hz. Each individual signs his/her name or another name using a touch pen while standing and keeping the PDA in his hand. For each signature, the acquisition device provides the position function and the elapsed time between the acquisitions of two successive points. For each individual and each session, 25 signatures (15 of them genuine and 10 forgeries) were collected. The Caltech database contains signatures acquired by a camera. About 25–30 genuine signatures and 10 forgeries from 105 individuals were collected [57].

Recently, in order to overcome the lack of online signatures, well-defined strategies for the generation of synthetic signatures have been considered. Two in particular have been proposed so far [22, 23]: (1) *synthetic sample generation*, in which well-defined transformations are used to generate synthetic samples from the real ones of a given individual – distortion-based techniques using elastic matching procedures [82] and variability estimation [18] have been proposed in the literature for this purpose – and (2) *synthetic individual generation*, in which a model of the signature produced by a population of individuals is created and new synthetic individuals can be generated by sampling the model – models based on spectral analysis [18] and Sigma-Lognormal parameters [21] have been considered for this purpose. Synthetic signatures have also been used for improving the enrolment procedure for a signature verification system [18] and for evaluating system resistance to brute-force attacks [21].

Finally, it is worth emphasizing that, to date, the characteristics of unimodal biometrics are not always considered adequate for large-scale deployment or for security critical applications, no matter which biometric trait is used. As a result, specific efforts are being made to incorporate multimodal biometrics into these applications. This addresses the problem of non-universality and is expected to achieve better performance than the unimodal approaches [22].

Concerning the definition of standards for experimental tests, international competitions are very important references for advancements in the field, since they provide common benchmark databases and testing protocols [100]. For instance, the Signature Verification Competition of 2004 (SVC2004), which considered

signatures acquired by a digitizing tablet, demonstrated that signature verification systems are no less accurate than other biometric systems, like those based on facial features or fingerprints [89]. In addition, the competitions are designed to allow researchers and practitioners to systematically evaluate the performance of online signature verification systems not only for error rates but also for other parameters, like interoperability, processing speed, and security of data. In fact, the feasibility of a particular system in a specific operating environment should be determined by analyzing all these parameters. For instance, concerning system interoperability, in the Signature Evaluation Campaign of 2009 (BSEC'2009), two different databases were considered, each containing data from the same individuals that were acquired by a digitizing tablet and a PDA, in order to measure the impact of a mobile device on the performance of signature verification systems [70].

---

## Online Signature Verification: A Comparative Analysis

Table 27.4 lists some online signature verification systems using distance-based classification techniques. In the approach proposed by Nakanishi et al. [60], the position signals of an online signature are decomposed into sub-band signals using the discrete wavelet transform (DWT), and Dynamic Programming was used for signature matching. Yeung et al. [100] reported the results of the First International Signature Verification Competition (SVC2004), in which teams from all over the world participated. SVC2004 considered two separate signature verification tasks using two different signature databases. The signature data for the first task contained position information only. The signature data for the second task contained position, pen inclination, and pressure information. In both cases, DTW-based approaches provided the best results. DTW was also used by Bovino et al. [4], who presented a multi-expert system based on a stroke-oriented description of signatures. Each stroke was analyzed in the position, velocity, and acceleration domain, and then a two-level scheme for decision combination was applied. For each stroke, soft- and hard-combination rules were used at the first level to combine decisions obtained by DTW from different representation domains. Simple and weighted averaging was used at the second level to combine decisions from different parts of the signature. Di Lecce et al. [11] performed signature verification by combining the efforts of three experts. The first expert used shape-based features and performed signature verification by means of global analysis. The second and third experts used speed-based features and a regional analysis. The expert decisions were combined by a majority voting strategy. Guru and Prakash [28] represented online signatures by interval-valued symbolic features. They used parameter-based features derived by a global analysis of signatures and achieved the best verification results when a writer-dependent threshold was adopted for distance-based classification. The system presented by Zhang et al. [103] used global, local, and function features. In the first verification stage, a parameter-based method was implemented, in which the Mahalanobis distance was used as a measure of dissimilarity between the signatures. The second verification stage

**Table 27.4** System performances: distance-based techniques full database (*FD*), signature (*S*), genuine signatures (*G*), forgeries (*F*), random forgeries (*RF*), simple forgeries (*SF*), skilled forgeries (*SK*), and number of authors (*A*)

Matching technique	Main features	Database	Results	Reference
Dynamic Programming	Wavelet transform	(Training) 20(G) from(4(A)) (Test) 98 (G) (from 4(A)), 200(F) (from 5(A))	EER: 4 %	I. Nakanishi et al. [60]
DTW (best result)	<b>Task 1:</b> position <b>Task 2:</b> position, pen inclination (azimuth), pressure, etc.	(Training) 800(G)(20(G)x40(A)), 800(SK)(20(SK)x40(A)) (Test 1) 600(G)(10(G)x60(A)), 1200(SK)(20(SK)x60(A)) (Test 2) 600(G)(10(G)x60(A)), 1200(RF)(20(RF)x60(A))	(Test 1) EER: 2.84 % (Task 1) EER: 2.89 % (Task 2) (Test 2) EER: 2.79 % (Task 1) EER: 2.51 % (Task 2)	D.-Y. Yeung et al. [100] (1st Signature Verification Competition)
DTW (ME by simple averaging)	Position, velocity, acceleration	(Training) 45(G) (3(G) x 15(A)) (Test) 750(G) (50(G)x15(A)), 750 (F) (50(F)x15(A))	EER: 0.4 %	L. Bovino et al. [4]
DTW (ME by majority voting)	Shape-based features (segmentation dependent), velocity	(Training) 45(G) (3(G) x 15(A)) (Test) 750 (G) (50(G)x15(A)), 750 (F) (50(F)x15(A))	FRR: 3.2 % FAR: 0.55 %	V. Di Lecce et al. [11]
DTW	Velocity, pressure	(FD) 4600 (S)	EER: 4 %	K. Huang and H. Yan [32]
DTW (PCA, MCA)	Position, velocity	(Training) 405 (G) (5(G) x 81(A)) (Test) 405 (G) (5(G) x 81(A)), 405(F) (5(F) x 81(A))	EER: 5.00 %	B. Li et al. [52]
Distance-based	Total signature duration, number of pen-ups, STD velocity and acceleration in x and y direction, number of local maxima in x direction, etc.	(FD1) (Training1) 2000 (G) (20(G) x 100(A)) (Test1) 500 (G) (5(G) x 100 (A)), 9900 (RF) (99 (RF) x 100(A)), 500 (SK) (5(SK) x 100(A)) (FD2) (Training2) 6600(G) (20(G) x 330(A))	(FD1) EER: 3.80 % (Test 1 – SK) EER: 1.65 % (Test 1 – RF) (FD2) EER: 4.70 % (Test 2 – SK) EER: 1.67 % (Test 2 – RF)	D. S. Guru and H. N. Prakash [28]

(continued)

**Table 27.4** (continued)

Matching technique	Main features	Database	Results	Reference
		(Test2) 1650 (G) (5(G) x 330(A)), 75570 (RF) (229(RF) x 330(A)), 8250(SK) (25(SK) x 330(A))		
Euclidean distance, Mahalanobis distance, DTW	Geometric-based, curvature-based	(FD) 306 (G), 302 (F)	<b>FRR:</b> 5.8 % <b>FAR:</b> 0 %	K. Zhang et al. [103]
Membership function	Total signature time, AVE/RMS speed, pressure, direction-based, number of pen-ups/pen-downs, etc.	(Test) 60 (G), 60 (F)	<b>FRR:</b> 6.67 % <b>FAR:</b> 1.67 %	T. Qu et al. [81]
Membership function	Speed, pressure, direction-based, Fourier transform	(FD) 1000 (G), 10000 (F)	<b>FRR:</b> 11.30 % <b>FAR:</b> 2.00 %	M. Zou et al. [104]
Dynamic similarity measure	Fourier transform (cepstrum coefficients)	(Training) 270(G) (from 27(A)) (Test) 560 (G) (from 27(A)), 650 (F)	<b>FRR:</b> 1.4 % <b>FAR:</b> 2.8 %	Q. Z. Wu et al. [97]
String matching	Velocity, curvature based	(FD) 1232 (S) (from 102 (A))	<b>FRR:</b> 3.3 % <b>FAR:</b> 2.7 % (common threshold) <b>FRR:</b> 2.8 % <b>FAR:</b> 1.6 % (writer-dependent threshold)	A. K. Jain et al. [40]

involved corner extraction and corner matching, as well as signature segmentation. In the third verification stage, an elastic matching algorithm was used, and a point-to-point correspondence was established between the compared signatures. By combining the three different types of verification, a high security level was reached. Zou et al. [104] used local shape analysis for online signature verification. Specifically, FTT was used to derive spectral and tremor features from well-defined segments of the signature. A weighted distance was finally considered, in order to combine the similarity values derived from the various feature sets. Fourier analysis

was applied by Wu et al. [97] for online signature verification, as they extracted and used cepstrum coefficients for verification, according to a dynamic similarity measure. Jain et al. [40] used a set of local parameters which described both spatial and temporal information. In the verification process, string matching was used to compare the test signature to all the signatures in the reference set. Three methods were investigated to combine the individual dissimilarity values into one value: the minimum, average, and maximum of all the dissimilarity values. Common and personalized (signer-dependent) thresholds were also considered. The best results were achieved by considering the minimum of all the dissimilarity values combined with personalized threshold values.

Table 27.5 reports some online signature verification systems using model-based classification techniques. M. Martinez-Diaz et al. [56] presented a Left-to-Right HMM-based signature verification system for handheld devices. Signatures were captured by a PDA and described by the position function only. The best results were achieved by user-dependent HMM [56]. Igarza et al. [34] also used a Left-to-Right HMM for online signature verification and demonstrated its superiority over Ergodic HMMs. The superiority of Principal Component Analysis and Minor Component Analysis for online signature verification over DTW and Euclidean-based verification was also investigated and demonstrated by Igarza et al. [35]. The online signature verification system proposed by Kashi et al. [42] used a Fourier transform-based normalization technique and both global and local features for signature modeling. The global features captured the spatial and temporal information of the signature, and the local features, extracted by a Left-to-Right HMM, captured the dynamics of the signature production process. The verification result was achieved by combining the information derived by both global and local features. Muramatsu and Matsumoto [58] used HMM-LR to incorporate signature trajectories for online signature verification. Individual features were extracted as high frequency signals in sub-bands. The total decision for verification was carried out by averaging the verification results achieved at each sub-band. Ortega-Garcia et al. [68] presented an investigation of the ability of HMM-LR to model the signing process, based on a set of 24 function features (8 basic function features and their first and second derivatives). In Shafiei and Rabiee's system [85], each signature was segmented using its perceptually important points. For every segment, four dynamic and three static parameters were extracted, which are scale and displacement invariant. HMM was used for signature verification. Wessels and Omlin [91] combined a Kohonen self-organizing feature map and HMM. Both Left-to-Right and Ergodic models were considered. Yang et al. [99] used directional features along with several HMM topologies for signature modeling. The results demonstrated that HMM-LR is superior to other topologies in capturing the individual features of the signatures while at the same time accepting variability in signing. A polar coordinate system was considered for signature representation by Yoon et al. [101], whose aim was to reduce normalization error and computing time. Signature modeling and verification were performed by HMMs, which demonstrated their ability to capture the local characteristics in the time sequence data, as well as their flexibility in terms of modeling signature variability. Lee et al. [49] performed signature verification by

**Table 27.5** System performances: model-based techniques full database (*FD*), signature (*S*), genuine signatures (*G*), forgeries (*F*), random forgeries (*RF*), simple forgeries (*SF*), skilled forgeries (*SK*), and number of authors (*A*)

Matching technique	Main features	Database	Results	Reference
HMM	Position	( <b>FD</b> ) 1000(G) / 20(G) x 50(A)), 1000(SK) / 20(SK) x 50(A)), ( <b>Training</b> ) 250(G) / 5(G) x 50(A)) ( <b>Test</b> ) 750(G) (15(G)x50(A)), 2450 (RF) (49(RF) x 50(A)), 1000 (SK) / 20(SK) x 50(A))	EER: 5.2 % (with <b>RF</b> ) EER: 15.8 % (with <b>SF</b> )	M. Martinez-Diaz et al. [56]
HMM	Direction of pen movement, etc.	( <b>FD</b> ) 3750 (G) (25(G)x150(A)), 3750 (F)	EER: 9.253 %	J. J. Igarza et al. [34]
HMM	Total signature time duration, X-Y (speed) correlation, RMS speed, moment-based, direction-based, etc.	( <b>Test</b> ) 542 (G), 325 (F)	EER: 2.5 %	R. S. Kashi et al. [42]
HMM	Direction of pen movements	( <b>Training</b> ) 165 (G) ( <b>Test</b> ) 1683 (G), 3170 (SK)	EER: 2.78 %	D. Muramatsu and T. Matsumoto [58]

HMM	Position, velocity, pressure, pen inclination (azimuth), direction of pen movement, etc.	(Training) 300(G) (from 50(A)) (Test) (450 (G) (from 50(A)), 750 (SK) (from 50(A)))	<b>EER:</b> 1.21 % (global threshold) <b>EER:</b> 0.35 % (user-specific threshold)	J. Ortega-Garcia et al. [68]
HMM	AVE speed, acceleration, pressure, direction of pen movement, etc.	(FD) 622 (G) (from 69(A)), 1010 (SK)	<b>FRR:</b> 12 % <b>FAR:</b> 4 %	M. M. Shafiei and H. R. Rabiee [85]
HMM	Position, pressure, direction of pen movements, pen inclination	(Training) 750 (G) (15(G) x 50(A)) (Test) 750 (G) (15(G) x 50(A))	<b>FAR:</b> 13 %	T. Wessels and C. W. Omlint [91]
HMM	Direction of pen movements	(FD) 496 (S) (from 31 (A))	<b>FRR:</b> 1.75 % <b>FAR:</b> 4.44 %	L. Yang et al. [99]
HMM	Position	(Training) 1500 (S) (15 (S) x 100 (A)) (Test) 500(S) (5 (S) x 100 (A))	<b>EER:</b> 2.2 %	H. S. Yoon et al. [101]
NN (with DP)	Position (geometric extrema), AVE velocity, number of pen-ups, time duration of neg/pos, velocity, total signing time, direction-based, etc.	(FD) 6790 (S) (from 271(A))	<b>EER:</b> 0.98 %	J. Lee et al. [49]

means of a backpropagation neural network, which integrated verification results at segment level, using a bottom-up strategy. They performed signature segmentation by means of a DP technique based on geometric extrema. Segment matching was performed by global features and a DP approach.

As Tables 27.4 and 27.5 show, experimental results achieved with small-to-medium-sized [42] and large [68] datasets demonstrate the superiority of model-based approaches over distance-based approaches. HMMs in particular are highly adaptable to personal variability and are very effective compared to other techniques for online signature verification [87]. It is worth noting that system performances are generally overestimated in the literature, since they were obtained from laboratory tests, which usually involve very controlled writing conditions and poor forgeries produced by researchers [72]. In addition, laboratory tests do not generally consider long-term variability, which can significantly reduce the estimated performance of online signature verification systems [50].

---

## Conclusion

Today, interest in online signature verification continues to grow, along with the increasingly stringent security requirements of a networked society. In fact, the handwritten signature is still one of the most widely accepted of the biometric traits, and automatic signature verification is generally welcomed as a noninvasive and nonthreatening process.

This chapter has presented an overview of online signature verification. In it, the fundamental aspects of the neuromuscular model underlying the signing process were addressed, as well as the main issues related to the design of an automatic signature verification system. Some of the most relevant state-of-the-art research was discussed and the most valuable results were summarized. In addition, a few of the most promising directions for research in this field were highlighted. In the near future, signature verification systems will be expected to operate in new working scenarios with specific constraints and challenging objectives. The net result will be that new research problems and challenges will emerge, and traditional and fundamental points of view will take on new meanings.

For instance, as the number of input devices and techniques for handwriting acquisition increases, device interoperability will become a more relevant issue and require investigation. Signature capture by both fixed and mobile devices will become feasible in many everyday environments, and automatic signature verification will be used in even more applications.

Analysis of the individual characteristics of handwriting remains an interesting research area and should encompass not only the features produced by people with normal abilities but also those generated by people with disabilities and illnesses that constrain their handwriting abilities. Investigation of the human mechanisms involved in handwriting production is therefore deserving of greater attention,

as well as studies on the feature selection techniques and signature modeling methods that produce the best possible description of the personal characteristics involved in signing. Techniques for analyzing signature complexity and stability can offer new insights into the selection of the most useful signature fragments and features for various kinds of applications and also to better understand time-based variations in signing.

Even though very effective model-based and distance-based matching techniques have been proposed, signature verification accuracy still needs to be improved significantly. This will require further effort in terms of the adaptability and personalization of verification processes. Multi-expert systems can also play an important role, since they can combine decisions obtained through top-down and bottom-up strategies, using matching algorithms at both the global and the local levels. The multi-expert approach also represents an important area for further research, with a view to using multimodal biometrics in signature verification, as their performances are expected to be superior to those of the unimodal approaches.

In addition, it will be important to develop this field of research in the direction of real applications, strictly in terms of the need to have benchmark databases available to a wide research community, in order to enable comparative evaluation of signature verification systems under different application scenarios. Such efforts are also necessary to support the development of fully interoperable systems, which take into consideration signature variability over time, as well as to build large databases for multicultural and multi-language signature verification systems which consider metadata information.

Finally, in order to advance the integration of automatic signature verification into a wide range of applications as soon as possible, specific attention should be paid to the issues of personal privacy and the protection of personal signature data, along with a clear definition of legal and regulatory norms for the identification of individuals by means of their handwritten signature.

Many of these new challenges will require to go out of the beaten tracks. On the one hand, it is expected that new investigations will be based on various pattern recognition paradigms that have not been fully exploited yet: fuzzy logics, evolutionary algorithms, particle swarm systems, and immune systems, to name a few, as well as their hybrid combinations, for example, fuzzy particle swarm optimization, and neuro-swarm systems. It is far from being clear for the moment how these techniques could be adapted to the various problems and challenges described in the present chapter, neither which of these will be practically efficient in designing new systems. On the other hand, it is also expected that as biometric security systems using the handwritten signatures will become ubiquitous, the biometric analysis of signatures for developing identification systems that can discriminate the health conditions rather than the identity of a subject might become a brand new research field. So far such studies have been focused on the analysis of handwriting [53, 65, 66, 88, 93, 98], but a preliminary study [67] dealing with the use of signatures to detect brain stroke risk factors is under way and looks very promising. As for many other challenges addressed in this handbook, the best has yet to come.

## Description of Consolidated Software and/or Reference Datasets

The following is a brief list of reference databases containing online signatures:

- BIOMET [24]
- BioSecure [70]
- Caltech [57]
- MCYT [69]
- Philips [73]
- SVC2004 [100]

---

## Cross-References

- ▶ [Datasets and Annotations for Document Analysis and Recognition](#)
- ▶ [Online Handwriting Recognition](#)
- ▶ [Sketching Interfaces](#)

---

## References

1. Alonso-Fernandez F, Fierrez J, Gilperez A, Galbally J, Ortega-Garcia J (2010) Robustness of signature verification systems to imitators with increasing skills. In: Proceedings of the 20th ICPR, Istanbul, 23–26 Aug 2010, pp 728–732
2. ANSI (2011) Data Format for the Interchange of Fingerprint, Facial & Other Biometric Information, NIST Special Publication, 500–290
3. Ballard L, Lopresti D, Monroe F (2007) Forgery quality and its implication for behavioural biometric security. IEEE Trans Syst Man Cybern Part B 37(5):1107–1118
4. Bovino L, Impedovo S, Pirlo G, Sarcinella L (2003) Multi-expert verification of hand-written signatures. In: 7th international conference on document analysis and recognition (ICDAR-7), Edinburgh, Aug 2003. IEEE Computer Society, pp 932–936
5. Brault J-J, Plamondon R (1993) A complexity measure of handwritten curves: modeling of dynamic signature forgery. IEEE Trans Syst Man Cybern (T-SMC) 23(2):400–413
6. Brault J-J, Plamondon R (1993) Segmenting handwritten signatures at their perceptually important points. IEEE Trans Pattern Anal Mach Intell (T-PAMI) 15(9):953–957
7. Chen Y, Ding X (2002) On-line signature verification using direction sequence string matching. Proc SPIE 4875:744–749
8. Congedo G, Dimauro G, Impedovo S, Pirlo G (1994) A new methodology for the measurement of local stability in dynamical signatures. In: 4th international workshop on frontiers in handwriting recognition (IWFHR-4), Taipei, Dec 1994, pp 135–144
9. Congedo G, Dimauro G, Forte AM, Impedovo S, Pirlo G (1995) Selecting reference signatures for on-line signature verification. In: Braccini C, De Floriani L, Vernazza G (eds) 8th international conference on image analysis and processing (ICIAP-8), San Remo, Sept 1995. LNCS 974. Springer, Berlin/Heidelberg, pp 521–526
10. Di Lecce V, Dimauro G, Guerriero A, Impedovo S, Pirlo G, Salzo A, Sarcinella L (1999) Selection of reference signatures for automatic signature verification. In: 5th international conference on document analysis and recognition (ICDAR-5), Bangalore, 20–22 Sept 1999, pp 597–600
11. Di Lecce V, Dimauro G, Guerriero A, Impedovo S, Pirlo G, Salzo A (2000) A multi-expert system for dynamic signature verification. In: Kittler J, Roli F (eds) 1st international workshop on multiple classifier systems (MCS 2000), Cagliari, June 2000. LNCS 1857. Springer, Berlin/Heidelberg, pp 320–329

12. Dimauro G, Impedovo S, Pirlo G (1993) On-line signature verification by a dynamic segmentation technique. In: 3rd international workshop on frontiers in handwriting recognition (IWFHR-3), Buffalo, May 1993, pp 262–271
13. Dimauro G, Impedovo S, Pirlo G (1994) Component-oriented algorithms for signature verification. *Int J Pattern Recognit Artif Intell (IJPRAI)* 8(3):771–794
14. Dimauro G, Impedovo S, Modugno R, Pirlo G, Sarcinella L (2002) Analysis of stability in hand-written dynamic signatures. In: 8th international workshop on frontiers in handwriting recognition (IWFHR-8), Niagara-on-the-Lake, Aug 2002, pp 259–263
15. Djouia M, Plamondon R (2009) Studying the variability of handwriting patterns using the kinematic theory. *Hum Mov Sci* 28(5):588–601
16. Dolfling JGA, Aarts EHL, Van Oosterhout JJGM (1998) On-line Verification signature with Hidden Markov Models. In: Proceedings of the 14th international conference on pattern recognition (ICPR-14), Brisbane, Aug 1998, pp 1309–1312
17. Feng H, Wah CC (2003) Online signature verification using a new extreme points warping technique. *Pattern Recognit Lett* 24(16):2943–2951. Elsevier Science Direct
18. Fuentes M, Garcia-Salicetti S, Dorizzi B (2002) On-line signature verification: fusion of a Hidden Markov Model and a neural network via a support vector machine. In: 8th international workshop on frontiers in handwriting recognition (IWFHR-8), Niagara-on-the-Lake, Aug 2002, pp 253–258
19. Galbally J, Fierrez J, Martinez-Diaz M, Ortega-Garcia J (2010) Improving the enrollment in dynamic signature verification with synthetic samples. In: Proceedings of the 20th ICPR, Istanbul, 23–26 Aug 2010, pp 1295–1299
20. Galbally J, Fierrez J, Martinez-Diaz M, Ortega-Garcia J (2010) Evaluation of brute-force attack to dynamic signature verification using synthetic samples. In: Proceedings of the 20th ICPR, Istanbul, 23–26 Aug 2010, pp 131–135
21. Galbally J, Fierrez J, Martinez-Diaz M, Ortega-Garcia J, Plamondon R, O'Reilly C (2010) Kinematical analysis of synthetic dynamic signatures using the sigma-lognormal model. In: Proceedings of the 12th ICFHR, Kolkata, 16–18 Nov 2010, pp 113–118
22. Galbally J, Plamondon R, Fierrez J, Ortega-Garcia J (2012) Synthetic on-line signature generation. Part I: methodology and algorithms. *Pattern Recognit* 45(7):2610–2621
23. Galbally J, Fierrez J, Ortega-Garcia J, Plamondon R (2012) Synthetic on-line signature generation. Part II: experimental validation. *Pattern Recognit* 45(7):2622–2632
24. Garcia-Salicetti S, Beumier C, Chollet G, Dorizzi B, Leroux les Jardins J, Lunter J, Ni Y, Petrovska-Delacrta D (2003) Biomet: a multimodal person authentication database including face, voice, fingerprint, hand and signature modalities. In: Kittler J, Nixon MS (eds) Audio-and video-based biometric person authentication. LNCS 2688. Springer, Berlin/Heidelberg, pp 845–853
25. Guest RM (2004) The repeatability of signatures. In: 9th international workshop on frontiers in handwriting recognition (IWFHR-9), Kichijoji, Oct 2004, pp 492–497
26. Guest R (2006) Age dependency in handwritten dynamic signature verification systems. *Pattern Recognit Lett* 27(10):1098–1104
27. Guest R, Fairhurst M (2006) Sample selection for optimising signature enrolment. In: Proceedings of the 10th international workshop on frontiers in handwriting recognition (IWFHR 10), La Baule, Oct 2006
28. Guru DS, Prakash HN (2009) Online signature verification and recognition: an approach based on symbolic representation. *IEEE Trans Pattern Anal Mach Intell (T-PAMI)* 31(6):1059–1073
29. Hongo Y, Muramatsu D, Matsumoto T (2005) AdaBoost-based on-line signature verifier. In: Biometric technology for human identification II, Orlando. Proceedings of the SPIE, Orlando, Florida, USA, vol 5779, pp 373–380
30. Houmani N, Garcia-Salicetti S, Dorizzi B (2008) A novel personal entropy measure confronted with online signature verification systems' performance. In: Proceedings of the IEEE 2nd international conference biometrics: theory, applications and systems (BTAS 2008), Washington, DC, Sept 2008, pp 1–6

31. Huang K, Yan H (1995) On-line signature verification based on dynamic segmentation and global and local matching. *Opt Eng* 34(12):3480–3487
32. Huang K, Yan H (2003) Stability and style-variation modeling for on-line signature verification. *Pattern Recognit* 36(10):2253–2270
33. Ibrahim MT, Kyan M, Khan MA, Alimgeer KS, Guan L (2010) On-line signature verification using 1-D velocity-based directional analysis. In: Proceedings of the 20th ICPR, Istanbul, 23–26 Aug 2010, pp 3830–3833
34. Igarza JJ, Goirizelaia I, Espinosa K, Hernández I, Méndez R, Sánchez J (2003) Online handwritten signature verification using Hidden Markov Models. In: Sanfelix A, Ruiz-Shulcloper J (eds) CIARP 2003, Havana, Cuba. LNCS 2905. Springer, Berlin/Heidelberg, Havana, Cuba, pp 391–399
35. Igarza JJ, Gómez L, Hernández I, Goirizelaia I (2004) Searching for an optimal reference system for on-line signature verification based on (x, y) alignment. In: Zhang D, Jain AK (eds) ICBA 2004, Hong Kong. LNCS 3072. Springer, Berlin/Heidelberg, Hong Kong, pp 519–525
36. Impedovo S, Pirlo G (2007) Verification of handwritten signatures: an overview. In: Proceedings of the 14th international conference on image analysis and processing (ICIAP 2007), Modena, 11–13 Sept 2007, pp 191–196
37. Impedovo D, Pirlo G (2008) Automatic signature verification – state of the art. *IEEE Trans Syst Man Cybern Part C Appl Rev* 38(5):609–635
38. Impedovo D, Pirlo G (2010) On-line signature verification by stroke-dependent representation domains. In: Proceedings of the 12th ICFHR, Kolkata, 16–18 Nov 2010, pp 623–627
39. ISO (2013) Information technology – Biometric data interchange formats – Part 11: Signature/sign processed dynamic data ISO/ IEC 19794-11
40. Jain AK, Griess FD, Connell SD (2002) On-line signature verification. *Pattern Recognit* 35(12):2963–2972
41. Kamel NS, Sayeed S, Ellis GA (2006) Glove-based approach to on-line signature verification. *IEEE Trans Pattern Anal Mach Intell (T-PAMI)* 30(6):1109–1113
42. Kashi RS, Hu J, Nelson WL, Turin W (1997) On-line handwritten signature verification using Hidden Markov Model features. In: 4th international conference on document analysis and recognition (ICDAR-4), Ulm, Aug 1997, vol I. IEEE Computer Society, pp 253–257
43. Kashi RS, Hu J, Nelson WL, Turin WL (1998) A Hidden Markov Model approach to online handwritten signature verification. *Int J Doc Anal Recognit (IJDAR)* 1(2):102–109
44. Khan MK, Khan MA, Khan MAU, Ahmad I (2006) On-line signature verification by exploiting inter-feature dependencies. In: Proceedings of the 18th international conference on pattern recognition (ICPR'06), Hong Kong, Aug 2006, pp 796–799
45. Kholmatov A, Yanikoglu B (2005) Identity authentication using improved online signature verification method. *Pattern Recognit Lett* 26:2400–2408
46. Komiya Y, Ohishi T, Matsumoto T (2001) A pen input on-line signature verifier integrating position, pressure and inclination trajectories. *IEICE Trans Inf Syst E84-D(7):833–838*
47. Leclerc F, Plamondon R (1994) Automatic signature verification: the state of the art – 1989–1993. *Int J Pattern Recognit Artif Intell (IJPRAI)* 8(3):643–660. In: Plamondon R (ed) (1994) Special issue on progress in automatic signature verification, Series in: MPAI, World Scientific, pp 3–20
48. Lee LL, Berger T, Aviczer E (1996) Reliable on-line human signature verification systems. *IEEE Trans Pattern Anal Mach Intell (T-PAMI)* 18(6):643–647
49. Lee J, Yoon H-S, Soh J, Chun BT, Chung YK (2004) Using geometric extrema for segment-to-segment characteristic comparison in online signature verification. *Pattern Recognit* 37(1):93–103
50. Lei H, Govindaraju V (2005) A comparative study on the consistency of features in on-line signature verification. *Pattern Recognit Lett* 26:2483–2489
51. Lejtman DZ, George SE (2001) On-line handwritten signature verification using wavelets and back-propagation neural networks. In: 6th international conference on document analysis and recognition (ICDAR-6), Seattle, Sept 2001, pp 992–996
52. Li B, Wang K, Zhang D (2004) On-line signature verification based on PCA (principal component analysis) and MCA (minor component analysis). In: Zhang D, Jain AK (eds)

- ICBA 2004, Hong Kong. LNCS 3072. Springer, Berlin/Heidelberg, Hong Kong, pp 540–546
- 53. Longstaff MG, Heath RA (2006) Spiral drawing performance as an indicator of fine motor function in people with multiple sclerosis. *Hum Mov Sci* 25:474–491
  - 54. Maiorana E, Campisi P, Fierrez J, Ortega-Garcia J, Neri A (2010) Cancelable templates for sequence-based biometrics with application to on-line signature recognition. *IEEE Trans Syst Man Cybern Part A Syst Hum* 40(3):525–538
  - 55. Martens R, Claesen L (1998) Incorporating local consistency information into the online signature verification process. *Int J Doc Anal Recognit (IJDAR)* 1(2):110–115
  - 56. Martinez-Diaz M, Fierrez J, Ortega-Garcia J (2008) Incorporating signature verification on handheld devices with user-dependent Hidden Markov Models. In: Proceedings of the ICFHR 2008, 19–21 Aug 2008, Montreal, pp 326–331
  - 57. Munich ME, Perona P (2003) Visual identification by signature tracking. *IEEE Trans Pattern Anal Mach Intell (T-PAMI)* 25(2):200–217
  - 58. Muramatsu D, Matsumoto T (2003) An HMM On-line signature verification algorithm. In: Kittler J, Nixon MS (eds) AVBPA 2003, Guildford. LNCS 2688, Springer, Berlin/Heidelberg, Guildford, UK, pp 233–241
  - 59. Nabeshima S, Yamamoto S, Agusa K, Taguchi T (1995) MemoPen: a new input device. In: International conference human factors in computer systems (CHI), Denver, Colorado, USA, pp 256–257
  - 60. Nakanishi I, Nishiguchi N, Itoh Y, Fukui Y (2004) On-line signature verification based on discrete wavelet domain adaptive signal processing. In: Zhang D, Jain AK (eds) ICBA 2004, Hong Kong. LNCS 3072, Springer, Berlin/Heidelberg, Hong Kong, pp 584–591
  - 61. Nanni L (2006) An advanced multi-matcher method for on-line signature verification featuring global features and tokenised random numbers. *Neurocomputing* 69(16–18):2402–2406
  - 62. Nanni L, Maiorana E, Lumini A, Campisi P (2010) Combining local, regional and global matchers for a template protected on-line signature verification system. *Expert Syst Appl* 37(5):3676–3684
  - 63. Nelson W, Turin W, Hastie T (1994) Statistical methods for on-line signature verification. *Int J Pattern Recognit Artif Intell (IJPRAI)* 8(3):749–770
  - 64. O'Reilly C, Plamondon R (2009) Development of a sigma-lognormal representation for on-line signatures. *Pattern Recognit Spec Issue Front Handwrit Recognit* 42:3324–3337
  - 65. O'Reilly C, Plamondon R (2011) Impact of stroke risk factors on human movements. *Hum Mov Sci* 30:792–806
  - 66. O'Reilly C, Plamondon R (2012) Design of a neuromuscular disorders diagnostic system using human movement analysis. In: Proceedings of the 11th international conference on information sciences, signal processing and their applications, Montreal, July 2012, pp 787–792
  - 67. O'Reilly C, Plamondon R (2012) Looking for the brain stroke signature. In: Proceedings of the 21st international conference on pattern recognition, Tsukuba, Nov 2012, pp 1811–1814
  - 68. Ortega-Garcia J, Fierrez-Aguilar J, Martin-Rello J, Gonzalez-Rodriguez J (2003) Complete signal modeling and score normalization for function-based dynamic signature verification. In: Kittler J, Nixon MS (eds) AVBPA'03, Guildford. LNCS 2688. Springer, Berlin/Heidelberg, pp 658–667
  - 69. Ortega-Garcia J, Fierrez-Aguilar J, Simon D, Gonzalez J, Faundez M, Espinosa V, Satue A, Hernaez I, Igarza J-J, Vivaracho C, Escudero D, Moro Q-I (2003) MCYT baseline corpus: a bimodal biometric database. *IEE Proc Vis Image Signal Process Spec Issue Biom Internet* 150(6):395–401
  - 70. Ortega-Garcia J, Fierrez J, Alonso-Fernandez F, Galbally J, Freire MR, Gonzalez-Rodriguez J, Garcia-Mateo C, Alba-Castro JL, González-Agulla E, Otero Muras E, Garcia-Salicetti S, Allano L, Ly V-B, Dorizzi B, Kittler J, Bourlai T, Poh N, Deravi F, Ng MWR, Fairhurst MC, Hennebert J, Humm A, Tistarelli M, Brodo L, Richiardi J, Drygajlo A, Ganster H, Sukno F, Pavani S-K, Frangi AF, Akarun L, Savran A (2010) The multiscenario multienvironment BioSecure Multimodal Database (BMDB). *IEEE Trans Pattern Anal Mach Intell (T-PAMI)* 32(6):1097–1111

71. Parizeau M, Plamondon R (1989) What types of scripts can be used for personal identity verification? In: Plamondon R, Suen CY, Simner M (eds) Computer recognition and human production of handwriting. World Scientific, Singapore/New Jersey/London/Hong Kong, pp 77–90
72. Parizeau M, Plamondon R (1990) A comparative analysis of regional correlation, dynamic time warping, and skeletal tree matching for signature verification. *IEEE Trans Pattern Anal Mach Intell (T-PAMI)* 12(7):710–717
73. Philips PJ, Martin A, Wilson CL, Przybocki M (2000) An introduction evaluating biometric systems. *Computer* 33(2):56–63
74. Plamondon R (ed) (1994) Progress in automatic signature verification. World Scientific, Singapore
75. Plamondon R (1994) The design of an on-line signature verification system: from theory to practice. *Int J Pattern Recognit Artif Intell (IJPRAI) Spec Issue Signat Verif* 8(3):795–811
76. Plamondon R (1995) A kinematic theory of rapid human movements: part I – movement representation and generation. *Biol Cybern* 72(4):295–307
77. Plamondon R, Djouia M (2006) A multi-level representation paradigm for handwriting stroke generation. *Hum Mov Sci* 25(4–5):586–607
78. Plamondon R, Lorette G (1989) Automatic signature verification and writer identification – the state of the art. *Pattern Recognit* 22(2):107–131
79. Plamondon R, Srihari SN (2000) On-line and off-line handwriting recognition: a comprehensive survey. *IEEE Trans Pattern Anal Mach Intell (T-PAMI)* 22(1):63–84
80. Pirlo G (1994) Algorithms for signature verification. In: Impedovo S (ed) Fundamentals in handwriting recognition. Proceedings of the NATO-ASI series. Springer, Berlin, pp 433–454
81. Qu T, El Saddik A, Adler A (2003) Dynamic signature verification system using stroke-based features. In: IEEE international workshop on haptic, audio and visual environments and their applications (HAVE 2003), Ottawa, Sept 2003, pp 83–88
82. Rabasse C, Guest RM, Fairhurst MC (2007) A method for the synthesis of dynamic biometric signature data. In: Proceedings of the 9th ICDAR, Curitiba, Brasil, 23–26 Sept 2007, pp 168–172
83. Schmidt C, Kraiss K-F (1997) Establishment of personalized templates for automatic signature verification. In: 4th international conference on document analysis and recognition (ICDAR-4), Ulm, Aug 1997, vol I. IEEE Computer Society, pp 263–267
84. Schomaker LRB, Plamondon R (1990) The relation between axial pen force and pen-point kinematics in handwriting. *Biol Cybern* 63:277–289
85. Shafiei MM, Rabiee HR (2003) A new on-line signature verification algorithm using variable length segmentation and Hidden Markov Models. In: 7th international conference on document analysis and recognition (ICDAR-7), Edinburgh, Aug 2003, vol I. IEEE Computer Society, pp 443–446
86. Uludag U, Pankanti S, Prabhakar S, Jain AK (2004) Biometric cryptosystems: issues and challenges. *Proc IEEE* 92(6):948–960
87. Van B, Garcia-Salicetti S, Dorizzi B (2007) On using the Viterbi path along with HMM likelihood information for online signature verification. *IEEE Trans Syst Man Cybern Part B* 37(5):1237–1247
88. Van Gemert W, Adler CH, Stelmach GE (2003) Parkinson's disease patients undershoot target size in handwriting and similar tasks. *J Neurol Neurosurg Psychiatry* 74:1502–1508
89. Vielhauer C (2005) A behavioural biometrics. *Public Serv Rev Eur Union* 20(9):113–115
90. Wang D, Zhang Y, Yao C, Wu J, Jiao H, Liu M (2010) Toward force-based signature verification: a pen-type sensor and preliminary validation. *IEEE Trans Instrum Meas* 59(4):752–762
91. Wessels T, Omlin CW (2000) A hybrid system for signature verification. In: International joint conference on neural networks (IJCNN 2000), Como, July 2000, vol 5, pp 509–514
92. Wirotius M, Ramel J-Y, Vincent N (2005) Comparison of point selection for characterizing on-line signature. In: Jain AK, Ratha NK (eds) Biometric technology for human identification II, Orlando, Mar 2005. Proceedings of the SPIE, vol 5779, Orlando, Florida, USA, pp 307–313

93. Woch A, Plamondon R, O'Reilly C (2011) Kinematic characteristics of successful movement primitives in young and older subjects: a delta-lognormal comparison. *Hum Mov Sci* 30:1–17
94. Wolf F, Basu TK, Dutta PK, Vielhauer C, Oermann A, Yegnanarayana B (2005) A cross-cultural evaluation framework for behavioral biometric user authentication. In: Spiliopoulou et al (eds) From data and information analysis to knowledge engineering. Springer, Berlin/Heidelberg, pp 654–661
95. Wu Q-Z, Jou I-C, Lee S-Y (1997) Online signature verification using LPC cepstrum and neural networks. *IEEE Trans Syst Man Cybern Part B Cybern* 27(1):148–153
96. Wu Q-Z, Lee S-Y, Jou I-C (1997) On-line signature verification based on split-and-merge matching mechanism. *Pattern Recognit Lett* 18(7):665–673
97. Wu Q-Z, Lee S-Y, Jou I-C (1998) On-line signature verification based on logarithmic spectrum. *Pattern Recognit* 31(12):1865–1871. Pergamon
98. Yan JH, Rountree S, Massman P, Doody RS, Li H (2008) Alzheimer's disease and mild cognitive impairment deteriorate fine movement control. *J Psychiatr Res* 42:1203–1212
99. Yang L, Widjaja BK, Prasad R (1995) Application of Hidden Markov Models for signature verification. *Pattern Recognit* 28(2):161–170
100. Yeung D-Y, Chang H, Xiong Y, George S, Kashi R, Matsumoto T, Rigoll G (2004) SVC2004: first international signature verification competition. In: Zhang D, Jain AK (eds) ICBA 2004, Hong Kong. LNCS 3072. Springer, Berlin/Heidelberg, Hong Kong, pp 16–22
101. Yoon HS, Lee JY, Yang HS (2002) An on-line signature verification system using Hidden Markov Model in polar space. In: 8th international workshop on frontiers in handwriting recognition (IWFHR-8), Ontario, Aug 2002. IEEE Computer Society, pp 329–333
102. Yue KW, Wijesoma WS (2000) Improved segmentation and segment association for on-line signature verification. *IEEE Int Conf Syst Man Cybern* 4:2752–2756
103. Zhang K, Nyssen E, Sahli H (2002) A multi-stage online signature verification system. *Pattern Anal Appl* 5:288–295
104. Zou M, Tong J, Liu C, Lou Z (2003) On-line signature verification using local shape analysis. In: 7th international conference on document analysis and recognition (ICDAR-7), Edinburgh, Aug 2003, vol I. IEEE Computer Society, pp 314–318

## Further Reading

- Proceeding01. Proceedings of the IAPR International Conferences on Biometry: ICB 2009 (Alghero, Italy), ICB 2011 (Washington DC, USA), ICB 2012 (New Delhi, India), ICB (2013, Madrid, Spain). <http://ieeexplore.ieee.org/xpl/conhome.jsp?punumber=1801205>
- Proceedings02. Proceedings of the IEEE International Conference on Biometrics: Theory, Applications and Systems: ICBTAS 2009, 2010, 2011, 2012 (Washington DC, USA). <http://ieeexplore.ieee.org/xpl/conhome.jsp?punumber=1001496>
- Proceedings03. Proceedings of the International Conferences on Frontiers in Handwriting Recognition: ICFHR 2008 (Montréal, Canada), 2010 (Kolkata, India), 2012 (Bari, Italy). <http://ieeexplore.ieee.org/xpl/conhome.jsp?punumber=1000298>
- Proceedings04. Proceedings of the Biennial Conferences of the International Graphonomics Society: IGS 2009 (Dijon, France), IGS 2011 (Cancun, Mexico), IGS 2013 (Nara, Japan). <http://www.graphonomics.org/publications.php>
- Proceedings05. Proceedings of the IAPR International Conference on Document Analysis and Recognition: ICDAR 2009 (Barcelona, Spain), 2011 (Beijing, China), 2013 (Washington DC, USA). <http://ieeexplore.ieee.org/xpl/conhome.jsp?punumber=1000219>
- Proceedings06. Special Issues of Human Movement Science on Handwriting: Vol.30, No.4 (2011) <http://www.sciencedirect.com/science/journal/01679457/30/4>
- Proceedings07. Special Issues of Human Movement Science on Handwriting: Vol.32, (2013). <http://www.sciencedirect.com/science/journal/01679457>
- Proceedings08. Special Issue of Motor Control Vol.14, No 1, (2010). <http://journals.human kinetics.com/mc-back-issues/MCVolume14Issue1January>

Tong Lu and Liu Wenyin

## Contents

Introduction.....	950
Sketching System Overview.....	951
Preprocessing.....	952
Denoising.....	954
Stroke Segmentation.....	954
Primitive Shape Fitting.....	954
Gesture Recognition.....	955
Methods for Distinguishing Between Gestures and Content Inputs.....	955
Recognition Algorithms.....	959
Online Graphics Recognition.....	962
Partial Graphic Object Recognition.....	962
Complete Graphic Object Recognition.....	963
Graphic Document Understanding.....	964
Applications.....	966
Sketch-Based Annotation.....	966
Sketch-Based Diagramming.....	968
Sketch-Based Design and Modeling.....	971
Sketch-Based Recognition and Simulation.....	973
Multi-domain Applications.....	974
Conclusion.....	975
Description of Consolidated Software and/or Reference Datasets.....	976
Cross-References.....	977
References.....	977
Further Reading.....	980

---

T. Lu (✉)

Department of Computer Science and Technology, State Key Laboratory for Novel Software Technology at Nanjing University, Nanjing, China  
e-mail: [lutong@nju.edu.cn](mailto:lutong@nju.edu.cn)

L. Wenyin

College of Computer Science and Technology, Shanghai University of Electric Power, Shanghai, China  
e-mail: [liuwenyin@gmail.com](mailto:liuwenyin@gmail.com)

**Abstract**

This chapter introduces the problems and technologies involved for implementing the sketching interfaces. A number of typical applications of sketching interface are discussed. Conclusion is drawn by summarizing the current issues and further research perspectives.

**Keywords**

Annotation tool • Graphic document understanding • Online graphics recognition • Pen input • Sketch recognition • Sketching interface • Stroke segmentation • Symbol recognition

---

**Introduction**

Sketching on paper is a natural way of externalizing thought and is often used in early prototyping stages to express and record design ideas. Its popularity has not decayed even after computer-aided design becomes popular. The commoditization of increasingly numbered pen-based input devices indicates the trend against traditional WIMP (window, icon, menu, and pointer) paradigm. The recent multi-touch interactive displays also enable users to interact with complex models through multiple points of control and natural hand gestures. The models can be directly activated and tracked on the display. The multi-touch control provides not only a continuous freehand experience but also a more direct visual feedback. The traditional single-point click-drag operations are slowly being replaced by continuous strokes and freehand gestures. Touch Screen Palm PDA, Blackberry, iPhone, Amazon Kindle 3, Sony Reader PRS 700, and a variety of tablet and table PCs such as the Microsoft Coffee Table make accessing information faster, easier, and more natural to interact with. In such trend, sketching interfaces play an important role in human-computer interaction.

The input in such sketching interfaces is either the user's command or the graphical content. Both require interpretation and understanding by the computer. A command of the user is expressed by a sketching gesture and used to operate the computer. The content is the data the user wishes to input as part of the whole graphic document. Both gesture and content are composed of strokes, the basic format of sketch input. These strokes usually undergo a preprocessing procedure, including noise filtration and/or stroke segmentation, before further processing. Then, the computer should distinguish between gesture input and content input before applying specific recognition procedures. Such distinction can be done either with an explicit mode switch between the gesture mode and the content mode or with an automatic classification. Besides, the whole graphic document can be analyzed and understood after its completion or partial completion. In this chapter, we present related problems and technologies involved for implementing the sketching interfaces together with selected instances of sketching interface applications.

Online sketches, or more formally, online sketching documents, are a special kind of graphical documents. Their analysis usually undergoes a similar technical framework as other graphical documents, including offline graphical documents, CAD drawings/diagrams, and geographic maps. However, sketches have both online/dynamic features (with known trajectories and temporal information of the graphic objects) and sketchy features (in irregular/free-form shapes). In this chapter, we focus on these specialties. Readers interested in techniques for analysis and interpretation of other offline or vector-form graphical documents at a broader context are recommended to refer to ►Chap. 17 (Analysis and Interpretation of Graphical Documents) of this handbook for more details.

---

## Sketching System Overview

Sketching interfaces are usually supported by specific hardware systems with direct pointing devices. The typical devices and software systems which support sketching interface applications include personal digital assistants (PDAs, e.g., Apple Newton), graphic tablets (e.g., Wacom tablet), tablet PCs, and now the most popular iPads and iPhones. Such devices accept input using either special pens/styluses or just fingers.

Similar to other online recognition systems introduced in ►Chap. 26 (Online Handwriting Recognition), the fundamental input format of sketching interfaces is a so-called stroke, which is a temporal sequence of  $X - Y$  coordinates representing the pen trajectory, captured using a digitizer that senses the pen-tip position, speed, etc. Without loss of generality, let us first examine several common scenarios of applications of sketching interfaces.

1. While a user is drawing a stroke, it is continuously morphed to the predicted primitive shape as in Arvo and Novins's work [1]. We refer to this type of recognition as simultaneous recognition since the recognition is simultaneously and continuously done while the user is inputting. Limited by its computational cost, this type of recognition systems is suitable for simple graphics input. On the other stream, primitive shape recognition is performed on the completion of strokes. We refer to this type of recognition as immediate recognition since recognition is only done right after the stroke is completely drawn. These systems include QuickDiagram [2], and CALI [3]. Moreover, Fonseca et al. [3] also refer to this kind of interface for graphics input as calligraphic interfaces.
2. After a user draws a freehand stroke, the stroke is segmented into a few fragments. Each of them is fitted and/or recognized as a primitive geometric shape. This step is called preprocessing. In some systems, this step is omitted and the original strokes simply undergo certain subsequent procedures for recognition and understanding.
3. The primitive shape segments resulted from scenario 2 or even the original strokes are then categorized to either gesture input or content input. There are two genres of methods for this purpose: implicit mode detection and explicit mode switching. The explicit approach requires the user to specify the category

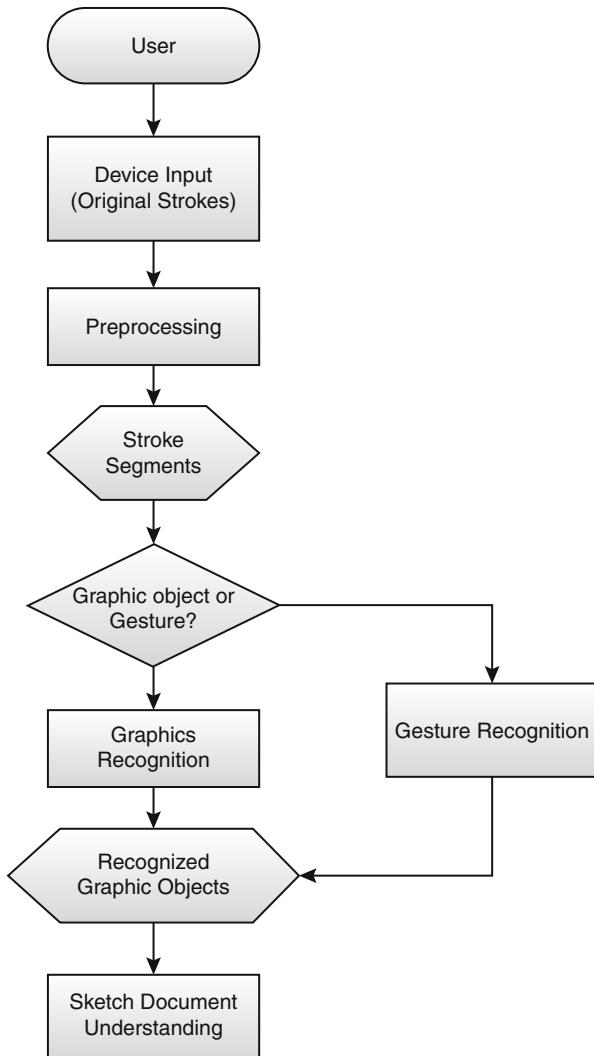
- of his/her next input by designated operations, such as pressing a certain button. The implicit approach does not require the user to specify his/her intended input category but automatically determines it.
4. If the input is determined as a part of the graphic object/document, the system proceeds to online graphics recognition step. Various features (either statistical or syntactic/structural or combined) of the strokes are usually extracted for classification. Graphic objects are usually recognized online, which means the temporal (or even others, e.g., pressure and speed) information of strokes is utilized. If a stroke is determined as part of a previously drawn graphic object, it is combined with the previously inputted components according to their spatial relationship to form a query of similar instances in the model database. The retrieved candidates are ranked according to their similarity to the query and displayed for the user to select. The object selected by the user then replaces the input strokes on the screen. In this way, the user does not need to finish the entire object and can therefore save some of his/her time. This technique is particularly useful for graphics input on small screen devices, and it is also a type of simultaneous recognition. On the other hand, it is also possible to recognize a composite object after its completion, as in immediate recognition.
  5. If the input is classified as a gesture, the system proceeds to gesture recognition procedure. Choice of features can be different from those of graphic object recognition. A typical example of gesture recognition is drawing an “X” gesture or some doodle lines (like using an eraser on a sheet of paper) on top of an existing graphic object which triggers delete operation of the graphic object. This is known as gesture recognition for editing. Many sketching systems, such as SILK [4] and CALI [3], embed such functions and gestures for common operations, namely, “delete,” “move,” and “copy.” A well-known application of graphical gesture recognition is a pen-based interface for interactive editing [5] on both text and graphics.
  6. After all the gestures and graphic objects are recognized, the system may perform understanding of the whole graphic documents in certain domain and output desired results or response.

As we can see from these scenarios, the most common, basic, and important application of sketching interfaces is inputting and editing graphic objects. Online graphics recognition is the key enabling technique for such kind of sketching interfaces. The overall workflow of the tasks in a sketching interface system is shown in Fig. 28.1. The remaining of this chapter focuses on introducing researches for each key procedure of the workflow. Then, we list the current advances of sketching interface applications. We conclude by delivering a thorough discussion on the current hindrances and perspectives of sketching interfaces.

---

## Preprocessing

In online sketching interfaces, the strokes are normally sampled as time-stamped points. There are methods, e.g., Rubine’s [6], which directly recognize the input



**Fig. 28.1** The framework of the procedures in a typical sketching interface system

stroke from the time-stamped points. These methods may work well for simple gesture recognition which will be discussed in section “[Direct Sample Point Matching](#).” However, direct recognition from time-stamped points is inefficient for composite online graphics, e.g., circuit diagrams. In this section, we will focus on existing methods for preprocessing input strokes into higher level primitive representations (i.e., denoising techniques, stroke segmentation techniques, and primitive fitting techniques) for further recognition and understanding processes.

## Denoising

In order to enhance the robustness of recognition, noncritical points, agglomerate points (caused by shaky operations), and end points need to be removed, filtered, and refined, respectively. These extra points or noise points are detected by analyzing the density of the stroke points or the distance from the points to the desired/intended line segment. Denoising is essential if the recognition method used relies on accurate stroke segmentation, for example, hidden Markov model (HMM) proposed by Sezgin and Davis [7].

## Stroke Segmentation

After noise reduction process introduced above, input strokes are normally segmented into sub-strokes and fitted into primitive shapes before conducting recognition. Existing methods of stroke segmentation can be categorized into local optimal approaches and global optimal approaches.

Local optimal approaches attempt to find breakpoints of a stroke, which is also referred to as splitting points. Most of these approaches first determine the support region for a stroke point. Then, within the support region, they calculate the significance of the point based on designated heuristics and use it as an indicator to identify splitting points. The heuristic for calculating the significance of a stroke point has been researched intensively. The local information, such as speed and curvature, of stroke points are utilized to detect splitting points more accurately [8]. The advantages of local optimal methods are that they are usually very fast and sometimes even parameter-free. However, their accuracy is limited due to the local optimality.

The second category is called globally optimal approach, which is also referred to as edge approximation. Subjective judgment criteria for perceptual significance, which are used to define a measure of perceptual error, are often introduced in this type of methods [9]. For every fragment of a stroke, a primitive shape with the minimal perceptual error is chosen to fit the original stroke. The advantage of this approach is that their accuracy is relatively high. However, almost all of them are not parameter-free; for example, some methods require the number of segments to be defined. Furthermore, the computational complexities of these methods are usually high.

## Primitive Shape Fitting

The variety of primitive elements used for fitting the user sketches is also a well-studied topic. Ray and Ray [10] fit strokes using only line segments. This approach is usually fast in computation but may produce a large number of segments for a cursive stroke. Therefore, higher order curves such as circular arcs are used.

They can produce a more accurate representation at a cost of increased computational complexity. With the development of ellipse fitting [11], elliptical arcs are used to produce a more accurate representation. Many other approaches finish the process of curve fitting using a combination of line segments and conic arcs.

---

## Gesture Recognition

In this section, we survey on the techniques used for distinguishing whether an input stroke(s) is a gesture (user command) or a part of the graphic document and what specific command it is for.

### Methods for Distinguishing Between Gestures and Content Inputs

The methods used for distinguishing between gestures and content inputs can be classified into two major classes, namely, explicit methods and implicit methods. Explicit methods require users to explicitly indicate the type of the next input stroke. In implicit methods, the distinguishing task is done by the computer. The reason for this classification is that the former simplifies the recognition complexity at cost of degrading user experience, while the latter attempts to improve user experience at cost of increasing computational complexity.

#### Explicit Methods

Sketching interfaces developed using explicit mode-switching methods usually support two distinct modes: gesture mode and content mode. In content mode, the strokes are taken as content input. In gesture mode, the strokes are taken as commands. There are many ways of implementing the mode switch, for instance, a button. However, users may have to frequently switch between these modes, and an inefficient mode-switching technique may become a major bottleneck for user experience of the interface. As a consequence, simple and effective explicit mode-switching techniques could provide users with consistent mechanisms that are applicable across a wide variety of applications. Here we list the most commonly used explicit mode-switching techniques:

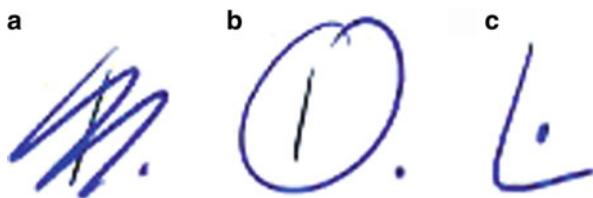
1. Barrel Button: This is the standard and frequently used technique in many existing pen-based applications [12]. Holding the barrel button (on the pen as shown in Fig. 28.2) down while drawing on the panel indicates a gesture input. Drawing without holding the barrel button indicates content input.
2. Press and Hold: This method requires a user to press the tip of the pen onto the tablet, hold it still until some mode change feedback appears, and then, while the mode change feedback still appears on the screen, the user can either lift the pen to bring up a pop-up menu or move the pen to draw a gesture. This technique leverages temporal information for switching modes, which is useful on devices where few input devices (including buttons) are available, e.g., a PDA or mobile phone. Once the pen touches the tablet, it enters the hold detection phase in which

**Fig. 28.2** A sample tablet

the bounding box size of the pen movement must be kept less than a pen travel threshold to be considered “still”. The system developed by Wenyin and Kong [2] includes this feature.

3. Non-preferred Hand: This method requires two-handed interactions. The preferred hand (e.g., the right hand) is used for drawing strokes, and the non-preferred hand (e.g., the left hand) switches modes by pressing the corresponding buttons (e.g., the button on the top left of the tablet shown in Fig. 28.2 can be used for switching modes using the left hand, while the right hand is used for holding the pen). This reduces the task completion time by simultaneously carrying out two subtasks. Many researches (e.g., [13]) have been done to examine the efficiency of two-handed interactions. Also, Ruiz et al. [14] have proposed a model for non-preferred hand model switching.
4. Pressure-Based Mode Switching: Pen pressure sensitive inputs are available on many tablet devices. The LEAN system proposed by Ramos et al. uses pressure as a feature for gesture recognition. Furthermore, Ramos et al. indicate that dividing the pressure space into six levels or less produces the best user performance [15]. Under the assumption that content input occupies most of the user’s time in a pen-based interface, they preserve the normal (middle) pressure space for content input and heavy spectrum for gesturing. They suggested a practical method for obtaining a suitable threshold value by using a small sample. They also suggested using personalized pressure setting to improve performance.
5. Using Pen with Eraser End: Some pens are designed with an eraser end (as shown in Fig. 28.2), and a user can use the eraser end of the pen for indicating gesture strokes [16].
6. Repeated Strokes: Since there are only two types of inputs, content and gesture, an alternative method for specifying input type is to repeat the strokes for one type of inputs. For example, the sketching interface proposed by Bae et al., EverybodyLovesSketch [17], uses the convention that repeated strokes are recognized as content input while otherwise a single stroke is recognized as a gesture. Alternatively, repeated strokes can be recognized as gestures for the interfaces where gesture inputs are less frequent.

**Fig. 28.3** Terminal punctuation. The *blue* strokes in (a), (b), and (c) correspond to two-stroke gestures (delete, select, and paste) (From [20])



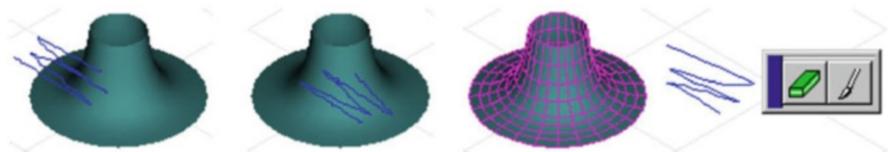
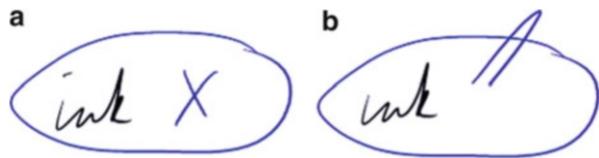
The experiment carried out by Li et al. [18] shows that non-preferred hand method offers the fastest performance. Press and hold is significantly slower than other methods and prone to error. Pressure-based mode switching did not perform as well as non-preferred hand method but can be improved using a personalized pressure space setting. Explicit methods are suitable for professional users, like designers or artists, since the working efficiency can be achieved heavily depends on the users' experience on the mode selection methods. The mode selection method using non-preferred hand is an optimal choice in this category for enhancing working efficiency. However, it has a limitation of requiring both hands to cooperate while working. Furthermore, explicit methods are usually hardware dependent as discussed above.

### Implicit Methods

Implicit methods do not require mode switching during user's input. The computer is responsible for determining whether an input pattern (can be formed by multiple strokes) is a gesture or a content input. There are still no standard methods for implementing implicit mode switching since they often require more complex recognition algorithms and architectural design. Furthermore, the effectiveness of implicit mode switching is limited due to the limited computational power a PC can provide. However, as the processing power of a PC is rapidly growing, implicit mode switching for generic sketching interfaces will eventually become feasible. The major aspects we need to consider for implementing implicit mode switching are:

1. Terminal punctuation gesture: Saund and Lank suggested to use terminal punctuation (additional strokes after an input pattern) to indicate a gesture input [19]. For example, as shown in Fig. 28.3, the dots are drawn after main gesture patterns to indicate gesture input. Terminal punctuation gestures can also be used to parameterize the main gesture. For example, drawing a cross sign inside a lasso immediately after drawing the lasso or drawing an acute angle crossing the drawn lasso indicates cut operation on the region selected by the lasso (as shown in Fig. 28.4). However, this method is not commonly used due to two major challenges: (a) how to distinguish punctuated gestures from regular content stroke without special hardware and (b) How to provide feedback of the state of the system given that until punctuation is specified, the preceding input may or may not be one of many gestures.
2. Gestural shortcuts: Zeleznik and Miller [20] suggest to use “/” like stroke followed by a mnemonic word or letter to perform meta-functions such as displaying a menu or widget. For example, drawing “/” with a letter “U”

**Fig. 28.4** Terminal punctuations indicating cut operation (From [20])



**Fig. 28.5** Context-based recognition example [21]

brings up an undo button on the screen. However, we face similar challenges as discussed in terminal punctuation gestures.

3. Context-based recognition: In order to use the gesture vocabulary efficiently, the same gesture can be interpreted differently under different circumstances. As shown in Fig. 28.5, a doodle stroke which crosses the edge of a screen object is interpreted as a delete command on the leftmost picture. In the middle, a stroke totally contained inside a screen object signifies recolor. On the right, a scratch gesture totally outside a screen object becomes ambiguous: a menu appears on screen asking the user which of the two meanings should be assumed.
4. Multi-touch gestures and uni-touch content input: This approach is rarely implemented or experimented. The key advantage of using multi-touch gestures for command input and uni-touch gestures for content input is that the distinction between command input and content input can be made at the very beginning of the gesture. Therefore, more useful functions can be achieved; for example, touching the screen using two fingers immediately triggers drag command and the content displayed on the screen moves along with the fingers. For instance, Kurihara et al. [22] developed a whiteboard system which applies similar concept. Although many multi-touch devices have been commoditized and a considerable amount of effort has been made for developing multi-touch gesture recognition technology, researches based on empirical evaluation and theoretical support are still rarely seen. There have been attempts of formulating the description of multi-touch gestures [23] and the interaction techniques using multi-touch gestures [24]. Furthermore, a few approaches, e.g., HMM [25] and neural network [26], have been proposed for recognizing multi-touch gestures. However, there is still space for further research studies.
5. Other information such as speed may also be used to distinguish a gesture from a graphic object input. For example, quickly drawing a line across a graphical object may be interpreted as deleting the graphical object. Whereas, drawing the line slowly may be interpreted as additional graphical input. However, not many researches have been done on this topic.

**Table 28.1** Comparison of the gesture recognition approaches

Type of approaches	Complexity of recognizable gestures	Training	Preprocessing	Relative recognition accuracy	Recognition speed
Hand-coded rules	Simple gestures	Not required	Not required	Acceptable for trivial cases	Quick for trivial cases but not acceptable for complex rules
Feature-based recognition	Dependent on the feature set	Required	Not required	Often high given enough samples	Dependent on the feature extraction and feature comparison time cost
Direct sample point matching	Simple uni-stroke gestures	Not required	Resizing and resampling of the input gestures	High for trivial cases	Quick
Probabilistic model	Accept complex multi-stroke gestures	Required and often computationally expensive	Stroke segmentation and primitive fitting are required	High given enough samples	Dependent on the complexity of the gestures

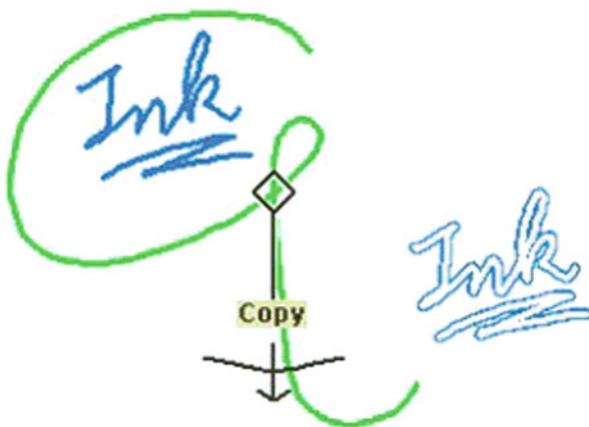
## Recognition Algorithms

In this section, the commonly used approaches for gesture recognition are discussed. Table 28.1 demonstrates a comparison among the recognition approaches with respect to the metrics: (1) complexity of recognizable gestures, (2) training, (3) preprocessing, (4) relative recognition accuracy, and (5) recognition speed.

### Hand-Coded Rules

Sketching interface systems recognize gestures using a great amount of different methods. Among these methods, rule-based ad hoc algorithms occupy the largest proportion of all the recognition methods at the early age. Those systems often empirically set several thresholds on the properties of the input gestures to perform simple hand-coded recognition. For example, in the work [27], the sketch system allows the user to draw a lasso surrounding a drawn object and then draw a “pigtail delimiter” to specify the desired operation among “cut,” “move,” “copy,” etc. As shown in Fig. 28.6, “pigtail delimiter” is detected by searching current pen stroke backwards over a 2,000 ms time window to look for an intersection line segment. If there is an intersection, the area and the perimeter of the polygon formed by the intersection of the stroke are calculated. Thresholds for those two parameters are empirically set to filter noises. Systems with gesture recognition method that lies in this category are usually complicated and difficult to implement, maintain, and modify.

**Fig. 28.6** A gesture called “pigtail delimiter” specifies “copy” command [27]



### Feature-Based Recognition

A better way of recognizing gestures other than hand-coded methods is to extract a set of features and train a gesture recognizer. The merit of a recognizer is that it could enable users or developers to extend the gesture commands easily. Unlike rule-based hand-coded recognizers, well-designed feature extraction recognizers are more stable and more error tolerating. It can also shift the burden of correcting sketch recognition from the user to the system.

Rubine developed a gesture manipulation interface called “GRANDMA,” which is short for “Gesture Recognizers Automated in a Novel Directed Manipulation Architecture” [6]. This is in literature the first attempt of developing a generic and extensible gesture management interface, enabling users to rapidly add gestures by training the single-stroke recognizer of the interface. The recognizer of the interface is feature-based single-stroke learner. The author decided to use 13 features extracted from the inputted training single-stroke gesture together with a linear discriminator for learning and classification. Normally, the recognizer requires 15–20 examples for training each gesture class. Then, given an input, the recognizer decides to which gesture class the input belongs. A reference implementation for Rubine’s recognizer is developed by Myers et al. [28]. There are several sketch systems, such as [29], which incorporate Rubine’s gesture recognizer.

Willems et al. [30] explored a large number of features for multi-stroke gesture recognition, and they presented a famous g-48 feature set. The detailed information of the feature set can be found in the appendix part of their work. With this feature set, the accuracy of tested classifiers (including multilayered perceptron, SVM, and DTW) can be significantly improved.

### Direct Sample Point Matching

Online gestures are often sampled as time-stamped points as discussed in the preprocessing section. There are methods which match an input gesture with a

defined gesture from point to point, for example, Protractor by Li [31]. This kind of methods first resamples the sample points of a raw gesture input as a fixed number, say  $N$ , of points which are equally spaced along the trajectory of the gesture. Then, the resampled gesture is rotated and scaled according to the designated heuristics of each specific method. The final step is to search for or compute from a closed form solution the optimal similarity between two gestures with respect to a defined similarity measure. This approach is easy to implement and often produce acceptable recognition accuracy for simple uni-stroke gestures. However, the computational cost grows rapidly as  $N$  increases which is inevitable for recognizing complex gestures.

### Probabilistic Model

If we make the basic assumption that the nature of pen-based gestures is a stochastic process, we can model and recognize them using the probabilistic modeling scheme. Unlike the linear discriminator used in GRANDMA, which only accepts single-stroke gestures, probabilistic model allows much more intelligent search from the space of all possible interpretations of sketches. However, this is at the cost of increasing the computational complexity since the size of the interpretations grows exponentially as the number of strokes increases. The most popular probabilistic model is HMM, which is largely used in speech recognition and symbol recognition. Its improved versions are also successfully applied in many domains, like variable duration Markov model (VDHMM) used in handwriting recognition which is discussed in ►Chap. 26 (Online Handwriting Recognition). An HMM assumes that the process being learned is composed of a finite sequence of discrete and stochastic hidden states. The states are assumed to be hidden but observable through features that the hidden states emit [32]. Anderson et al. [32] also state three classical problems associated with HMMs:

1. Classification problem: Given an observation sequence, which model is the most likely to have emitted particular sequence?
2. Most likely state sequence problem: What is the most likely sequence of hidden states which are believed to have emitted a given observation sequence?
3. Training or learning problem: Given one or more observation sequences, what are the model parameters associated which maximizing the likelihood of the observations?

There are quite a number of successful approaches for sketch gesture system using HMM or its variations as recognition scheme. Sezgin and Davis [7] presented semantic-based stroke recognizer modeled with HMMs. The recognizer views the process of sketching a gesture/symbol as an incremental, interactive process. By incremental it means strokes are put on the sketching surface of the sketch device one at a time. Its HMMs incorporate the sequential information of the strokes so that the sketching is highly stylized. For example, when sketching iconic people, most people tend to start from the head rather than other parts. This information is utilized and can drastically affect the interpretation of sketches.

## Online Graphics Recognition

Online graphics recognition is referred to as the entire procedure of recognizing the desired graphic object while sketching is taking place. It may also be called real-time or dynamic symbol recognition. The recognition approaches introduced for online gesture recognition can be adapted to online graphics recognition. However, differing from online gesture recognition, online graphics recognition can be interactive. For example, a user does not need to complete the drawing before choosing the desired symbol from a candidate list suggested by the system [2]. A user may also make corrections after the system gives immediate feedback and thus avoid accumulating too many errors. Furthermore, the immediate feedback may also yield user adaptation. If a particular symbol keeps being wrongly recognized, the user can alter his/her sketching style to improve recognition accuracy. On the other hand, some recognizers are capable of adapting itself to suit a particular drawing style [33].

The procedure of online graphics recognition can be activated at different levels of detail, such as at the primitive shape level, and most commonly, at the composite graphics object level [34]. At the primitive shape level, the predicted primitive shape can be continuously morphed while a user is drawing a stroke [1]. This type of recognition is also referred to as simultaneous recognition since recognition is simultaneously and continuously done along with the user's inputting progress. However, it is mainly suitable for simple graphics input. This is because the recognition gets updated whenever a new sample point is inputted, resulting in high computational cost and slow response time for complex strokes. Hence, it is more common that the primitive shape is recognized only after a stroke is completely drawn.

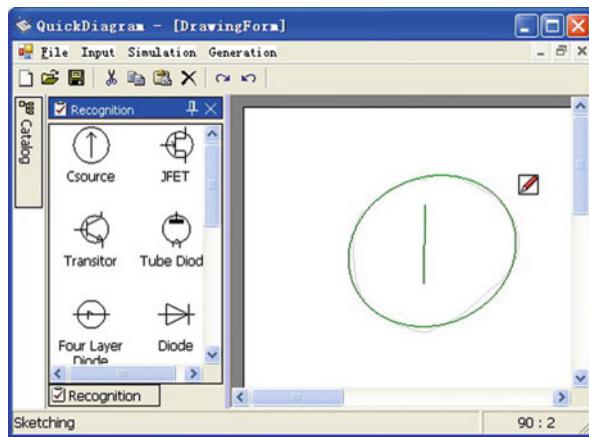
In most situations, online graphics recognition is performed at the level of composite graphics objects, in which the basic processing unit is a primitive. After recognizing and regularizing the current stroke into primitives (each primitive corresponds to a segment of the stroke), it is desired to combine together the latest primitives with the ones recognized from previous strokes based on both their spatial and temporal relationships, determine or predict the type and parameters of the composite graphics object that the user is intending to input, and then redisplay (replace on the screen the freehand one) it in its regular form. The composite graphic objects are usually predefined in a model database.

Depending on whether the composite graphic object is recognized after it is completed or not, the recognition methods are grouped into two categories: partial graphics recognition and complete graphics recognition.

## Partial Graphic Object Recognition

Partial graphic object recognition is referred to as the process that the system returns potential matches before the graphic object is completely drawn. In order to achieve this, recognition must take place while the graphic object is being drawn or, more

**Fig. 28.7** Partial symbol recognition of QuickDiagram. Partial graphic input (composed by a circle and a vertical straight line) on the right is recognized as the potential matches on the left



precisely, after each intermediate stroke is drawn. There are some sketch interfaces supporting partial graphics recognition. Partial similarity is calculated to search for the user-intended object. In order to achieve efficient recognition and prediction of the partially drawn object, the search space is usually reduced by temporal or spatial constraints.

A number of partial graphic object recognition systems rely on temporal constraints to prune the search space. Lank et al. [35] develop a UML system, assuming that objects are drawn using consecutive strokes. Other systems limit the search space by employing greedy recognition strategies [36]. An example of this continuous prediction process from QuickDiagram [2] is shown in Fig. 28.7. While these greedy recognition methods do not limit the graphic objects to being drawn consecutively, they may create undesired interdependencies among objects. Besides, dynamic information is also used to aid recognition; e.g., Sezgin and Davis [37] rely on preferred stroke orders and build an HMM to recognize each symbol.

Many partial graphics recognition systems use spatial constraints to limit their search. Most of the works consider only groups of strokes that locate within a spatial bound to be the same object [38, 39]. To further prune the search space, many systems set a hard threshold on constraints between sub-shapes [38, 40]. Few combinations need to be considered for recognition. However, it is difficult to select a suitable threshold.

Existing researches on online graphics recognition focus more on complete graphic object recognition and only assess the recognition accuracy for completed graphic objects. Partial graphic object recognition has drawn much less attention, let alone its recognition accuracy or prediction performance or efficiency.

## Complete Graphic Object Recognition

There are mainly three categories of approaches to this problem: rule-based approaches, machine learning approaches, and similarity-based approaches.

## 1. Rule-Based Approaches

An intuitive approach to graphics recognition is to build a decision tree where each leaf represents a class of graphics objects and each edge represents a rule to classify to branches. However, this approach suffers from its poor extensibility. The tree structure must be updated or redesigned when adding new classes of graphic objects. There are some improvements of these rule-based approaches to enhance the adaptability. Sun et al. [41] also adopt an incremental decision tree induction method to construct dynamic user models to provide user adaptation.

## 2. Machine Learning Approaches

The recognition of graphic object can be regarded as a classification problem and solved by machine learning algorithms, such as neural networks (NN) [42], HMM [7], and genetic algorithm (GA) [43]. Hybrid methods are also used, e.g., NN/HMM approaches. The advantage of these machine learning approaches is that they are robust to noise and can be easily extended, while the disadvantage is that pretraining is required before the classification process. Also, the performance of the classifiers is greatly affected by the size of the training set and the selection of the training samples.

## 3. Similarity-Based Approaches

In similarity-based approaches, similarities between two graphic objects are defined and calculated. Usually, the graphic object is represented in the form of graph [44]. In general, these graphs are matched and the most similar graphic object is selected from the database as the recognition result. These approaches are easy to extend and do not require a large amount of training data. However, they usually require a high computational cost and have been proven to be NP complete. Furthermore, these approaches are sensitive to the noise and, hence, not robust.

►Chapters 17 (Analysis and Interpretation of Graphical Documents) and ►16 (An Overview of Symbol Recognition) in this handbook deal with the analysis and interpretation of general graphical documents including CAD drawings, electronic diagrams, and geographic maps, most of which are offline data. These kinds of data undergo a similar recognition framework as sketching data does, though detail recognition methods may be different.

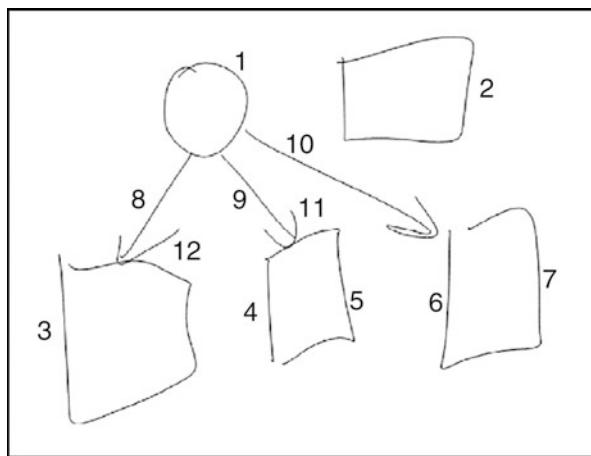
---

## Graphic Document Understanding

Understanding of a sketchy graphic document is similar to those regular graphic documents after the sketchy graphic objects and their relations are recognized. A graphic document (or a diagram/drawing) is usually represented using a graph. Symbols (or graphic objects) are represented by nodes and are linked by their relations. Properties of the graphic objects are represented by the attributes of the nodes. Links (edges in the graph) can also have certain properties/attributes.

The key issue in graphic document understanding is how to represent and store such graph configuration and relevant domain knowledge in a sketching system. There are quite a number of categories of domain knowledge representation, and in this section, we only present a few common examples.

**Fig. 28.8** Snapshot of the family tree understanding in [47]



1. Rule-based hand coding: This category may have the most number of applications. These systems usually combine the graphic document representation with domain knowledge using hand coding. This is the most intuitive way of document understanding since it only needs a number of hand coded rules to represent the domain knowledge and configuration of the graph representation. Although the systems that adopt such method are easy to design, there is almost no room for extension, e.g., to learn a new domain of symbols.
2. XML knowledge representation and understanding: One example of such category is [2]. In this work, not only the graph representation of the document but also the constraints for circuit and the symbol configuration are defined and stored using XML knowledge representation. When the user would like the system to understand a new set of symbols, he/she only needs to define the symbols' XML representation using symbol developing interface. The user can also conduct certain analyses of the graphic documents, with the help of the XML representation of the graphic document.
3. Constraint programming language: Generally, constraint-based techniques enable the user to program the interactive behavior of a graphic document. In computer-aided design, a constraint-based graphic editor has a distinct advantage over a conventional drawing program: a user can state desired relations and the system ensures that they are maintained. As early as "Sketchpad" [45], constraint programming language is used to maintain spatial relationship among the drawn symbols. Alvarado and Davis [46] represent a shape's definition in a language called Ladder. As Fig. 28.8 shows, Ladder descriptions list the basic components making up the shape and the geometric constraints that must hold between them. Recognizing individual shapes is then a process of matching the strokes drawn against a set of shape descriptions for a domain. The description is significant both for what it constrains and for what it does not constrain. For example, to be an arrow the shaft must be longer than the heads. It does not specify the lines'

orientation or length, letting the system recognize arrows of any size and in any orientation.

In the next section, we will also show the graphic document understanding aspects of these systems.

The techniques for understanding sketchy graphic document fall into the category of “hybrid approaches for interpretation” in section “Hybrid Approaches for Graphics Analysis” of ►Chap. 17 (Analysis and Interpretation of Graphical Documents) in this handbook, where both components/elements recognition and domain knowledge are used to reach an overall understanding of the whole graphical document. Readers may also refer to ►Chap. 17 (Analysis and Interpretation of Graphical Documents) for a broader discussion of analysis and interpretation of other kinds of graphical documents, including CAD drawings, electronic diagrams, geographic maps, and mechanic charts, most of which are offline data.

---

## Applications

With the rise of pen-based and touch-based technologies, the number of sketch-based tools and interfaces is increasing. A variety of domains are benefiting from the sketch interfaces, including annotation and editing, user interface design, sketch-based modeling, diagram recognition, and mathematical equations. In addition, a few multi-domain recognition toolkits are also reported.

### Sketch-Based Annotation

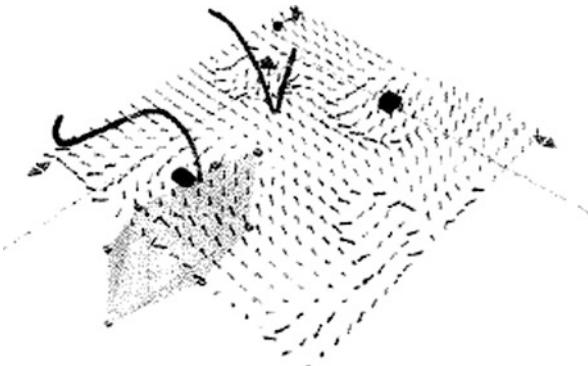
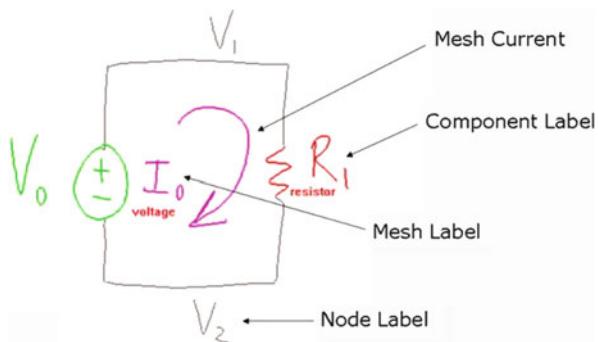
Annotation is one of the most natural applications of sketching interface, since sketching is popularly used as an annotation tool to facilitate information understanding. For example, they can help clarify basic assumptions or specify technical details. This section gives a brief overview of sketch-based annotation tools and the corresponding techniques.

A typical sketch-based annotation system needs to provide all the aspects of a sketching system (including both recognition and understanding) and should provide means for the following: (1) freely selecting contents to annotate, such as graphic symbols or textual words; (2) adaptively adding sketchy annotations to the selected contents; and (3) conveniently establishing relationships among the selected elements and the sketchy annotations.

#### 2D Annotation

Kirchhoff’s Pen [48] is a study system that provides tutoring for Kirchhoff’s voltage law (KVL) and current law (KCL) using sketching technologies. The circuit annotations are shown in Fig. 28.9. The goal of Kirchhoff’s Pen is to figure out the circuit formulas related to KVL or KCL and verify them with students’ answers. As shown in Fig. 28.9, the annotations are sketched freely by users, making it difficult

**Fig. 28.9** Typical circuit annotations in Kirchhoff's Pen



**Fig. 28.10** The example of 3D annotation [49]

to automatically identify the associated object for each annotation or extract the relationships among the annotations.

In the identification process, Kirchhoff's Pen translates the sketched text annotations into characters by a handwriting recognizer and a special-purpose recognizer, simultaneously avoiding the misclassifications of special pairs such as “I”-“1” or “\”-“/.” The handwriting recognizer produces a ranked set of alternative interpretations for each character, and the highest-ranked choice that is consistent with their problem domain is selected. After identifying the annotated text characters, Kirchhoff's Pen associates them with their neighboring circuit symbols to obtain annotated labels.

### 3D Annotation

How to annotate in a 3D scene is an interesting problem. Loughlin and Hughes [49] proposed a system that supports annotation as an integrated part of a fluid flow visualization system, as shown in Fig. 28.10. Unlike typical annotations on static 2D images, this system embeds annotations in 3D data space. This immersion makes it easy to associate user comments with the features they describe. To avoid clutter and data hiding, annotations are represented by graphical annotation markers that have associated information. Therefore, graphical attributes of the markers, such as size and color, can differentiate annotations with different functions, authors, etc.

Annotations can be easily added, edited, and deleted in the 3D space. Moreover, annotations can be simultaneously loaded into a visualization approach. This allows scientists, collaborating on a dataset, to use annotations as a form of communication, as well as a history of data analysis sessions. Annotation markers also aid scientists in navigating through the data space by providing landmarks at interesting positions.

The annotations in their system are actually represented by small geometric markers in the 3D data space. Each marker has an associated content which the user can edit at any time. The geometry of a marker gives visual feedback on the content of the annotation. Interactive shadows could be added to the markers on the planes defined by the principal axes. Each shadow is constrained to move in the plane in which it lies. If a user moves a shadow, the marker moves in a parallel plane. This constrained translation helps to precisely position a marker. Shadows are used to highlight specific zones interested by annotators.

The features of the points highlighted by geometric markers are later recognized and stored. Since the features of unsteady fluid flows change over time, the annotations are relocated by recognizing the features of the annotated points. The time-varying annotations are supported by feature-characterization code to improve the annotation effects. The feature-characterization code is the specifications of the features of annotations found.

## Sketch-Based Diagramming

Sketch-aided diagramming is used in different domains, such as circuit diagram, and UML diagram, to provide a natural and intuitive interaction environment. In this section, we will briefly talk about several typical examples of sketch-based diagramming.

### Circuit Diagram Analysis

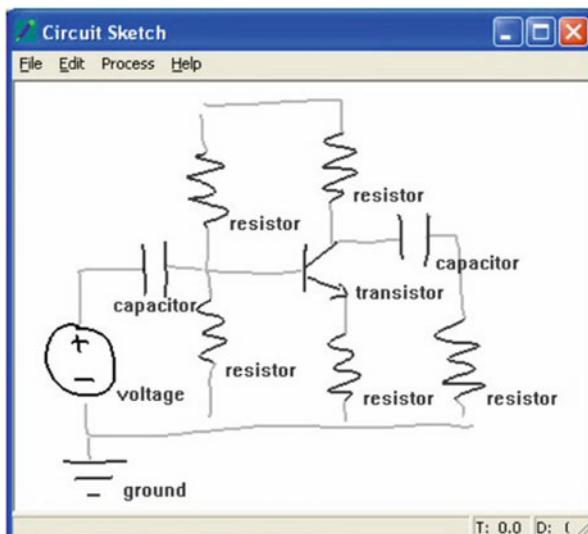
Sketch-aided circuit diagram drawing and analysis is not a new subject to researchers. There are several systems and applications that can be found in literature.

#### AC-SPARC

AC-SPARC is developed by Gennari et al. [50]. The name of this system is the abbreviation for analog circuit sketch pArsing, recognition, and error correction. As Fig. 28.11 shows, users are expected to draw network-like diagrams consisting of symbols, using mouse or digitizing tablet and stylus, and link them together. A parser automatically extracts geometric information from continuous stream of strokes. Candidate symbols of recognition are pruned using domain knowledge, while they are classified with domain independent, probabilistic, feature-based symbol recognizer. Error correction carried out automatically is aided by the domain knowledge and the context within the sketched diagram. For electronic diagrams, SPICE code can be generated for further analysis.

Following the standard procedures of sketch recognition system, AC-SPARC starts with the “ink segmentation” in its architecture of interpreter. As the user

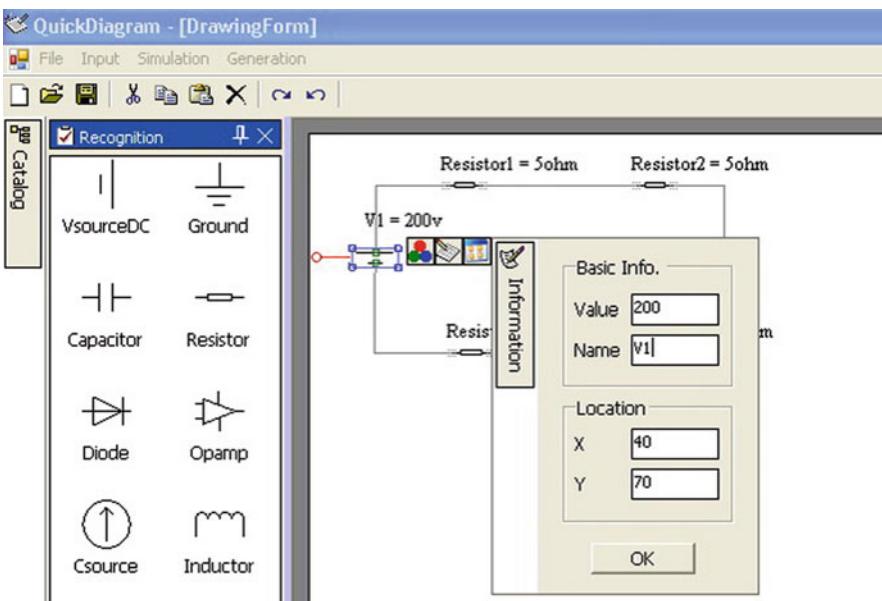
**Fig. 28.11** AC-SPARC sketch interface and interpretation result



draws, pixels of the sketches are segmented into lines and arcs, which serve as the geometric “primitives” for recognition. Stahovich’s algorithm for segmentation [8] is chosen in AC-SPARC.

Given a set of primitives, it is necessary to combine the right ones to form candidate symbols while distinguishing the link primitives. AC-SPARC does this by employing two locators and identifying the starting and ending of one symbol’s drawing. One locator is the “ink density locator” which calculates the density of each primitive among its bounding box. Drawn time of the primitives is also recorded in order to include the imaginative link. Density series are calculated by accumulating primitives and the indication of the end point is that the addition of other segments causes decrease of density. This step is repeated but is started from the end point to find the starting point. The other locator is the segment difference locator. It calculates four characteristics of each primitive, and if two primitives have more than two (or a larger number as a predefined threshold) different characteristics, they are considered as segment difference point. A symbol is likely to be bounded by two-segment difference point, having more than two but less than the user-defined maximum number of primitives.

Candidate symbols are pruned according to domain-based knowledge. For electronic symbols, particular rules can be developed, such as the ink density is high, two primitives touching the candidate symbol are more or less collinear, enough primitives or at least a full circle is generally contained, and so on. After pruning, symbol recognition is performed. Symbol recognition in AC-SPARC involves two stages as training and classification. Firstly, it extracts nine geometric features of symbols, and secondly, it extracts nine types of geometric features and performs classification using a naive Bayesian framework. The classifier is invariant to rotation, scaling, translation, and order of strokes.



**Fig. 28.12** Property dialog box for a device where its name and parameter value can be modified

Finally, automated error correction and some predefined and hard-coded rules for the circuit are used; for example, the number of the connection of a symbol should be 2 or 3. The system will either automatically correct the detected errors or give the user a warning.

### QuickDiagram

QuickDiagram [2] is a system for quick diagramming input and understanding. With a user sketching a (complete or partial) component/symbol or a wire (connecting two components) of the diagram, the system can recognize and beautify it immediately. After the entire diagram is finished, certain understandings can be obtained. Especially, the following two methods are used to interpret the recognized diagram: (1) nodal analysis on resistive circuits and (2) generation of PSpice codes from the recognized diagrams.

QuickDiagram can understand certain circuit diagrams, in which each device is modeled in its regular form with its connection points also marked. After recognition, each device is assigned with a default name automatically numbered and displayed beside the device in the diagram. The default value of its parameters, e.g.,  $R1 = 5 \Omega$ , is also assigned and displayed next to its name. Its name and value can also be modified in its property dialog box, as shown in Fig. 28.12. The connection points of the devices are also numbered such that each connection point in the entire diagram has a unique name. After all the individual devices and their

connections are recognized, the entire diagram is displayed in a neat form prior to further analysis and understanding.

### UML Diagramming

Another popular domain for sketch-based diagramming is UML diagramming.

Dachselt et al. [51] present the usage of digital pens and paper in conjunction with sketch-based UML tools to provide a flexible spontaneous sketching. Their work deals with the seamless integration of pen-based and digital UML sketching. It can be extended to use with tabletops.

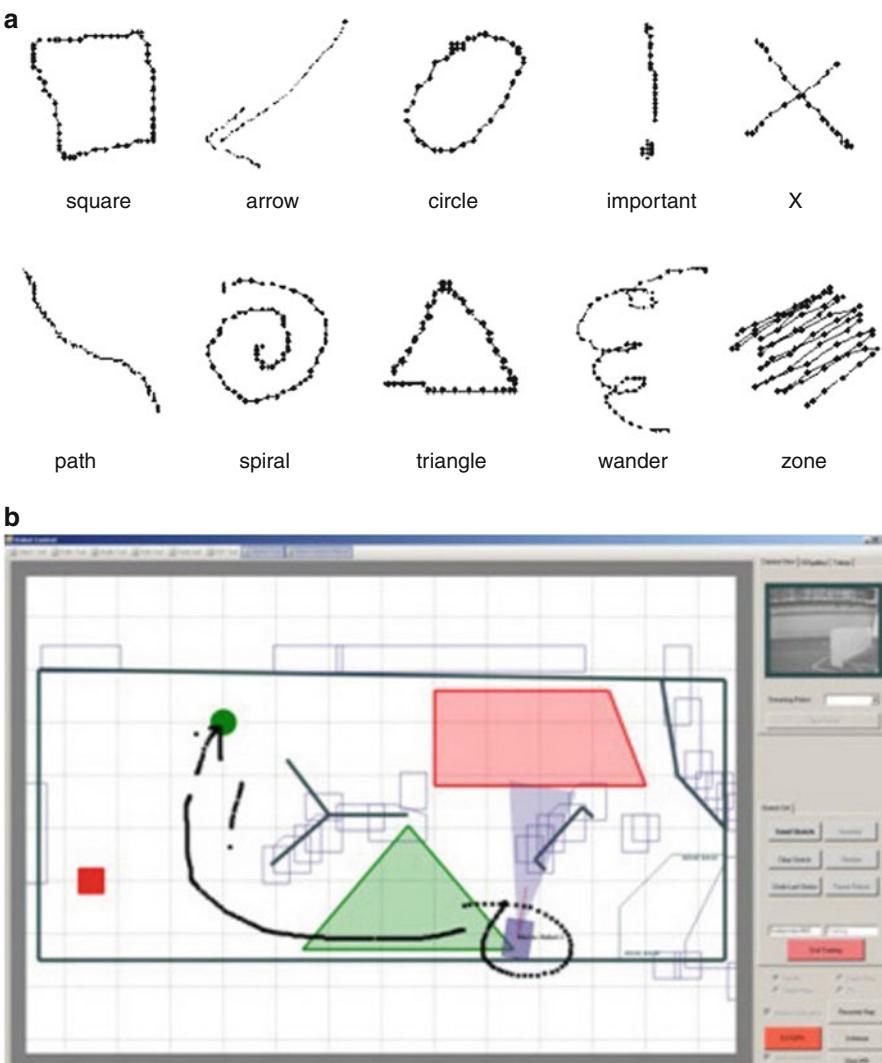
Chen et al. [52] present a UML tool called SUMLOW that allows user to draw UML constructs, mix different UML elements, and annotate diagrams and input text. A key novelty of the tool is the preservation of hand-drawn diagrams and support for manipulation of these sketches using pen-based actions. Sketched diagrams can be automatically “formalized” into computer-recognized and beautified UML diagrams and then exported to a third-party CASE tool for further extension and use. They describe the motivation for SUMLOW, illustrate the use of the tool to sketch various UML diagram types, describe its key architecture abstractions and implementation approaches, and report on two evaluations of the toolset. Their experiences are useful for others developing sketching-based design tools or those looking to leverage pen-based interfaces in software applications.

### Sketch-Based Design and Modeling

There are various fields of designs that can be improved both on performance and efficiency if they are aided by sketching interfaces or systems.

### User Interface Design

Benefiting from the flexible nature of sketch drawing, commands that are made using sketches for computer control are more intuitive and more comfortable. This is particularly evident during the interaction between human and robot, especially under urgent circumstances, such as search-and-rescue scenarios. Shah et al. [53] developed a probabilistic command interface for controlling robots using multi-strokes sketch command. Their system allows the user to draw ten gestures to indicate his/her commands, as Fig. 28.13a illustrates. For example, a user can draw a circle around a robot to active it for further control. An X is referred to as a waypoint that the robot has to pass through during the movement and an arrow indicates the robot’s travel route and the final orientation (indicated by the orientation of the head of the arrow). A zone specifies an area that the robot must explore, and the robot would randomly sample waypoints within the area and pass through. The user can simply draw the sketches to control the robot using a stylus and a tablet. Figure 28.13b presents a snapshot of the interface of the system.

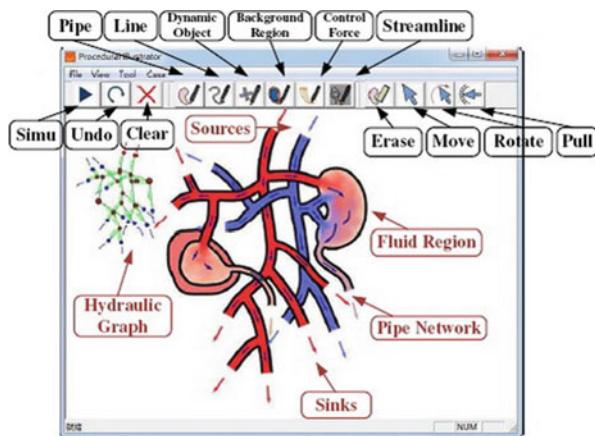


**Fig. 28.13** (a) Ten sketches of command and (b) interface snap shot. Four sketch commands are used: *circle*, *arrow*, *zone*, and *triangle*

## 2.5D Modeling

Zhu et al. [54] propose a sketch-based dynamic illustration method of fluid system. They present a lightweight sketching system that enables interactive design, editing, and illustration of complex fluid systems. The modeling is performed on a 2.5D canvas to design the object shapes and connections of a fluid circuit. The input sketches are automatically analyzed and abstracted. Fluid model is required to enhance the illustration in the background. Rich operations are provided by the

**Fig. 28.14** A screen snapshot of the sketch-based fluid illustration system [54]



system for users to edit the fluid system incrementally, and the new internal flow pattern is modeled in real time. This work is tested in domains of medicine, biology, and engineering. Figure 28.14 shows a screen snapshot of their system.

### 3D Modeling

Sketch-based 3D construction or understanding uses similar techniques at the 2D level of recognition and understanding. However, more techniques specific to the 3D level are required. Especially, Zeleznik et al. [55] invent an interface to input 3D sketchy shapes by recognizing the predefined patterns of some 2D shapes that represent certain sketchy solid shapes.

## Sketch-Based Recognition and Simulation

### Chemical Drawing Recognition

Shi and Zhang [56] propose a SVM-HMM-based classifier for online chemical symbol recognition. They modify the PSR algorithm [57] into the MPSR algorithm to enhance the processing efficiency and user-friendliness.

Given the size of the symbol set and the distinction between organic ring structure (ORS) and non-ring structure (NRS), their method cannot use only a single-stage classifier to perform classification for all symbols. Therefore, they design a double-stage classification architecture, in which the SVM-based classifier roughly classifies the symbols into ORS and NRS at the first stage, then the HMM-based classifier performs fine classification at the second stage. The input of the whole classification architecture is an isolated handwritten chemical symbol and the outputs are the top three candidates.

Handwritten chemical symbol is a temporal sequence of X-Y points captured using a digitizer that senses the pen-tip position while writing. Due to arbitrary handwriting and various writing styles, noises, broken, and conglomerate strokes are

inevitable and make the recognition more difficult. However, stochastic models can effectively deal with noises and variations caused by handwriting. As a stochastic model, HMMs are suitable for handwritten recognition for many reasons. Therefore, HMM is applied to model handwritten chemical symbols at the second stage [56].

### **Mathematical Expression**

Mathematical expression analysis is a frequently studied area and a more complete survey can be found in ►Chap. 20 (Processing Mathematical Notation) of this handbook. In this section, we only focus on mathematical expression analysis in the context of sketching interfaces.

Typical online mathematical expression recognition approaches [58] follow two steps. First, strokes are segmented and grouped to form single symbols, i.e., digits, and operators. Second, structural analysis is employed to determine the mathematical relationship among the symbols. This step is particularly challenging due to different writing styles and irregularity of the layout of the writing. If there are more than two symbols, their relationship can only be determined globally. That is, the first symbol has to be located before any other decisions to be made. To overcome such problems, Tapia and Rojas [59] use a minimum spanning tree and symbol dominance analysis to handle overlapping and controversial association of objects and horizontal layout irregularities. Using contextual information to solve the ambiguities of interpretation is also discussed in the work from Belaid et al. [60]. However, better solution is provided by adopting some context-free frameworks and probabilistic models. Koschinski et al. [61] proposed a system to recognize online handwritten mathematic expressions on a tablet.

### **Mechanical Engineering**

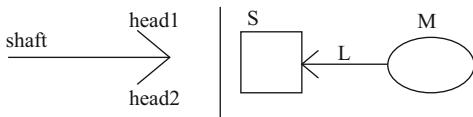
ASSIST, a Shrewd Sketch Interpretation and Simulation Tool [38], can understand mechanical engineering sketches and then interpret their mechanisms using simulation. It is particularly interesting that physical gravity and contact between the objects can be well simulated, like a floating object can fall down to the ground without supporting.

EsQUIsE [62] captures and understands architectural sketches and then builds 3D models based on the recognition results. It can also carry evaluation of the project like the real-time walk time through assessment of the building's energy needs, building and functioning cost, etc.

### **Multi-domain Applications**

Most sketch systems are domain dependent and most attempts on domain-independent systems are proved to be unsuccessful [47]. To specify a certain domain field, graphical lexicon, syntax, and semantics must be well modeled. Attempts on the combination of domain specification and sketch recognition can be found in the work [46]. In their work, a system called SketchREAD is developed. In this system, a novel descriptive language for shape's structural organization

**Fig. 28.15** The LADDER syntax for the shape “arrow.” The domain pattern “mother-son” is also indicated



```

DEFINE ARROW
(Subshapes
L1,L2,L3: (Line shaft head1 head2))

(Constraints
C1: (coincident
      shaft.p1 head1.p1)
C2: (coincident
      shaft.p1 head2.p2)
C3: (equal-length head1 head2)
C4: (shorter head1 shaft)
C5: (acute-angle head1 shaft)
C6: (acute-angle head2 shaft))

DEFINE MOTHER-SON
(Subshapes
M,S,L: (Female M) (Male S) (Child-link L))

(Constraints
C1: (touches L.head S)
C2: (touches L.tail M))

```

called LADDER is used. Under its description, no training data or programming is necessary. Figure 28.15 illustrates an example description on an arrow and some other hierarchy information used in the matching step.

## Conclusion

In this chapter, we present the problems, technologies, and applications of sketch interface systems, which take freehand sketching input as either command gestures or graphic objects. The main procedures of sketch processing are the following: preprocessing, recognition, and understanding.

Preprocessing techniques in this context include denoising, stroke segmentation, and primitive fitting. Classical recognition methods usually omit the preprocessing stage, while modern recognition approaches employ preprocessing techniques to enhance their performance.

Methods proposed for recognizing gestures and graphic objects are presented. As demonstrated, the initial stage of the recognition process distinguishes between gestures and graphic objects. Two approaches for gestures/graphic objects distinction, namely, explicit mode switching and implicit mode detection, are elaborated. Once the distinction is finished, corresponding recognition processes are carried out for gestures or graphic objects. Gestures are often easy to recognize due to their simplicity, whereas recognition of graphic objects requires more sophisticated mathematical models and interactive techniques for obtaining better recognition accuracy. In sections “[Gesture Recognition](#)” and “[On-Line Graphics Recognition](#),” we brief the researches done for both gesture recognition and graphic object recognition, particularly for sketch interfaces, stressing on their online natures. The major approaches can be summarized as follows:

1. Hand-coded or rule-based methods
2. Feature extraction methods
3. Direct sample point matching methods
4. Probabilistic model-based methods

However, as discussed in sections “[Partial Graphic Object Recognition](#)” and “[On-Line Graphics Recognition](#),” none of these methods is perfect for all situations. They either suffer from performance limitations or are computationally expensive. Furthermore, the machine learning approaches which often produce accurate recognition rely on having a cumbersome training stage. However, there are quite a few ready-to-use API’s and toolkits for gesture recognition with moderate performance, as listed in the next section.

Graphic document understanding is discussed in section “[Complete Graphic Object Recognition](#).” This processing stage is domain dependent. Graphic document understanding provides the user with more intuitive understanding of the sketch inputs and, hence, enhances user experience. Three mainstream approaches, namely, rule-based hand-coding, XML knowledge representation and understanding, and constraint programming language, for graphic document understanding are discussed.

In section “[Applications](#),” we look at the sketch interface applications for different purposes: annotation, diagramming, design and modeling, recognition and simulation, and multi-domain applications. We think that domain-specific sketch interface will still be a major research direction. In other words, the sketch recognition and understanding methods should base on a specific application in order to achieve more domain-specific functions. On the other hand, although the major aim of recognition methods is still improving recognition accuracy with low computational cost, recognition methods with specialties, such as partial symbol recognition, start to attract more attentions.

---

## Description of Consolidated Software and/or Reference Datasets

*QuickDiagram* (previously, *SmartSketchpad*) is a sketching interface for quick diagram input and understanding. It involves quite a number of methods for stroke processing, symbol recognition, and syntax and semantic. With a user sketching a (complete or partial) symbol or wire (connecting two symbols) of the diagram, the system can recognize and beautify it immediately. After the entire diagram is complete, it can be analyzed and understood via nodal analysis and generation of PSpice codes. The *QuickDiagram* system is released as an open-source project at <http://code.google.com/p/quickdiagram/>.

de Silva R, Bischel DT Lee WS, Peterson EJ, Calfee RC and Stahovich TF (2007) Kirchhoff’s Pen: A pen-based circuit analysis tutor. University of California, Riverside.

\$1 gesture recognizer is an open-source simple gesture recognition toolkit which has been implemented in various programming languages. The original version only supports uni-stroke gestures. \$N gesture recognizer is the extended version

for uni-touch and multi-stroke gestures (<http://depts.washington.edu/aimgroup/proj/dollar/>).

*Android gesture package* is the open-source gesture recognition package provided by google for android SDK. Its current version (4.0) implements an offline gesture recognition approach which describes each gesture using a statistical descriptor and retrieves a predefined gesture by comparing the descriptors. It supports uni-touch and multi-stroke gestures (<http://developer.android.com/reference/android/gesture/package-summary.html>).

*Apple iOS gesture recognition package* is the open-source gesture recognition package provided by apple for iOS SDK. It supports simple multi-touch gestures ([http://developer.apple.com/library/ios/#documentation/UIKit/Reference/UIGestureRecognizer\\_Class/Reference/Reference.html##apple\\_ref/occ/cl/UIGestureRecognizer](http://developer.apple.com/library/ios/#documentation/UIKit/Reference/UIGestureRecognizer_Class/Reference/Reference.html##apple_ref/occ/cl/UIGestureRecognizer)).

*Silverlight for Windows phone toolkit* also includes API's for touch input and gesture recognition package ([http://create.msdn.com/en-US/education/quickstarts/Touch\\_Input](http://create.msdn.com/en-US/education/quickstarts/Touch_Input)).

*Multi-touch systems I have known and loved* is an online overview of historical multi-touch system written by Bill Buxton (<http://billbuxton.com/multitouchOverview.htm>).

---

## Cross-References

- ▶ [An Overview of Symbol Recognition](#)
- ▶ [Analysis and Interpretation of Graphical Documents](#)
- ▶ [Processing Mathematical Notation](#)

---

## References

1. Arvo J, Novins K (2000) Fluid sketches: continuous recognition and morphing of simple hand-drawn shapes. In: Proceedings of the 13th annual ACM symposium on user interface software and technology 2000, San Diego. ACM, pp 73–80
2. Wenjin L et al (2010) QuickDiagram: a system for online sketching and understanding of diagrams. In: Ogier J-M, Liu W, Lladós J (eds) Graphics recognition. Achievements, challenges, and evolution. Springer, Berlin/Heidelberg, pp 130–141
3. Fonseca MJ, Pimentel C, Jorge JA (2002) CALI: an online scribble recognizer for calligraphic interfaces. In: AAAI 2002 spring symposium – sketch understanding, Palo Alto, pp 51–58
4. Landay JA, Myers BA (1995) Interactive sketching for the early stages of user interface design. In: Proceedings of the SIGCHI conference on human factors in computing systems 1995, Denver. ACM/Addison-Wesley, pp 43–50
5. Saund E et al (2002) Perceptual organization as a foundation for intelligent sketch editing. In: AAAI 2002 spring symposium – sketch understanding 2002, Palo Alto
6. Rubine D (1991) Specifying gestures by example. ACM SIGGRAPH Comput Graph 25(4):329–337
7. Sezgin TM, Davis R (2005) HMM-based efficient sketch recognition. In: Proceedings of the 10th international conference on intelligent user interfaces 2005, San Diego. ACM, pp 281–283

8. Herold J, Stahovich TF (2011) SpeedSeg: a technique for segmenting pen strokes using pen speed. *Comput Graph* 35(2):250–264
9. Horng J-H, Li JT (2001) A dynamic programming approach for fitting digital planar curves with line segments and circular arcs. *Pattern Recognit Lett* 22(2):183–197
10. Ray BK, Ray KS (1994) A non-parametric sequential method for polygonal approximation of digital curves. *Pattern Recognit Lett* 15(2):161–167
11. Fitzgibbon A, Pilu M, Fisher RB (1999) Direct least square fitting of ellipses. *IEEE Trans Pattern Anal Mach Intell* 21(5):476–480
12. Liao C et al (2008) Papercraft: a gesture-based command system for interactive paper. *ACM Trans Comput Hum Interact* 14(4):1–27
13. Kabbash P, Buxton W, Sellen A (1994) Two-handed input in a compound task. In: Proceedings of the SIGCHI conference on human factors in computing systems: celebrating interdependence 1994, Boston. ACM, pp 417–423
14. Ruiz J, Bunt A, Lank E (2008) A model of non-preferred hand mode switching. In: Proceedings of graphics interface 2008, Windsor. Canadian Information Processing Society, pp 49–56
15. Ramos G, Boulos M, Balakrishnan R (2004) Pressure widgets. In: Proceedings of the SIGCHI conference on human factors in computing systems 2004, Vienna. ACM, pp 487–494
16. Forsberg A, Dieterich M, Zeleznik R (1998) The music notepad. In: Proceedings of the 11th annual ACM symposium on user interface software and technology 1998, San Francisco. ACM, pp 203–210
17. Bae S-H, Balakrishnan R, Singh K (2009) EverybodyLovesSketch: 3D sketching for a broader audience. In: Proceedings of the 22nd annual ACM symposium on user interface software and technology 2009, Victoria. ACM, pp 59–68
18. Li Y et al (2005) Experimental analysis of mode switching techniques in pen-based user interfaces. In: Proceedings of the SIGCHI conference on human factors in computing systems 2005, Portland. ACM, pp 461–470
19. Saund E, Lank E (2003) Stylus input and editing without prior selection of mode. In: Proceedings of the 16th annual ACM symposium on user interface software and technology 2003, Vancouver. ACM, pp 213–216
20. Zeleznik R, Miller T (2006) Fluid inking: augmenting the medium of free-form inking with gestures. In: Proceedings of graphics interface 2006, Quebec. Canadian Information Processing Society, pp 155–162
21. Samavati FF, Olsen L, Jorge JA (2011) Introduction (Chapter 1). In: Jorge J, Samavati F (eds) Sketch-based interfaces and modeling. Springer-Verlag London, pp 1–15
22. Kurihara K et al (2011) Toward localizing audiences' gaze using a multi-touch electronic whiteboard with sPieMenu. In: Proceedings of the 16th international conference on intelligent user interfaces 2011, Palo Alto. ACM, pp 379–382
23. Wang D-x, Xiong Z-h, Zhang M-j (2011) An application oriented and shape feature based multi-touch gesture description and recognition method. *Multimed Tools Appl* 23(2):1–23
24. Yee W (2009) Potential limitations of multi-touch gesture vocabulary: differentiation, adoption, fatigue human-computer interaction. In: Jacko J (ed) Novel interaction methods and techniques. Springer, Berlin/Heidelberg, pp 291–300
25. Damaraju S, Kerne A (2008) Multitouch gesture learning and recognition system. In: Extended abstracts of IEEE workshop on tabletops and interactive surfaces 2008, Amsterdam, pp 102–104
26. Li W, He H (2010) Fingertip tracking and multi-point gesture recognition. In: International symposium on intelligent signal processing and communication systems, Chengdu
27. Hinckley K et al (2005) Design and analysis of delimiters for selection-action pen gesture phrases in scriboli. In: Proceedings of the SIGCHI conference on human factors in computing systems 2005, Portland. ACM, pp 451–460
28. Myers BA et al (1997) The Amulet environment: new models for effective user interface software development. *IEEE Trans Softw Eng* 23(6):347–365
29. Ramos G, Balakrishnan R (2003) Fluid interaction techniques for the control and annotation of digital video. In: Proceedings of the 16th annual ACM symposium on user interface software and technology 2003, Vancouver. ACM, pp 105–114

30. Willems D et al (2009) Iconic and multi-stroke gesture recognition. *Pattern Recognit* 42(12):3303–3312
31. Li Y (2010) Protractor: a fast and accurate gesture recognizer. In: Proceedings of the SIGCHI conference on human factors in computing systems 2010, Atlanta. ACM, pp 2169–2172
32. Anderson D, Bailey C, Skubic M (2004) Hidden Markov model symbol recognition for sketch-based interfaces. In: AAAI 2004 Fall symposium, Arlington
33. Tappert CC, Suen CY, Wakahara T (1990) The state of the art in online handwriting recognition. *IEEE Trans Pattern Anal Mach Intell* 12(8):787–808
34. Wenjin L (2004) On-line graphics recognition: state-of-the-art. In: Lladós J, Kwon Y-B (eds) *Graphics recognition. Recent advances and perspectives*. Springer, Berlin/Heidelberg, pp 291–304
35. Lank E et al (2001) On-line recognition of UML diagrams. In: Proceedings of the 6th international conference on document analysis and recognition, Seattle
36. Lin J et al (2001) DENIM: an informal tool for early stage web site design. In: Proceedings of the SIGCHI conference on human factors in computing systems 2001, Seattle. ACM, pp 205–206
37. Sezgin TM, Davis R (2008) Sketch recognition in interspersed drawings using time-based graphical models. *Comput Graph* 32(5):500–510
38. Alvarado C, Davis R (2006) Resolving ambiguities to create a natural computer-based sketching environment. In: ACM SIGGRAPH 2006 courses, Boston. ACM, p 24
39. Hammond T, Davis R (2006) Tahuti: a geometrical sketch recognition system for UML class diagrams. In: ACM SIGGRAPH 2006 courses, Boston. ACM, p 25
40. Mahoney JV, Fromherz MPJ (2002) Three main concerns in sketch recognition and an approach to addressing them. In: AAAI 2002 Spring symposium – sketch understanding, Palo Alto
41. Sun Z et al (2004) User adaptation for online sketchy shape recognition. In: Lladós J, Kwon Y-B (eds) *Graphics recognition. Recent advances and perspectives*. Springer, Berlin/Heidelberg, pp 305–316
42. Lee S-W, Kim Y-J (1995) A new type of recurrent neural network for handwritten character recognition. In: Proceedings of the 3rd international conference on document analysis and recognition, Montreal
43. Chen K-Z et al (2003) Recognition of digital curves scanned from paper drawings using genetic algorithms. *Pattern Recognit* 36(1):123–130
44. Lladós J, Martí E, Villanueva JJ (2001) Symbol recognition by error-tolerant subgraph matching between region adjacency graphs. *IEEE Trans Pattern Anal Mach Intell* 23(10): 1137–1143
45. Sutherland IE (2003) Sketchpad: a man-machine graphical communication system. Technical reports, University of Cambridge
46. Alvarado C, Davis R (2007) SketchREAD: a multi-domain sketch recognition engine. In: ACM SIGGRAPH 2007 courses, San Diego. ACM, p 34
47. Davis R (2007) Magic paper: sketch-understanding research. *Computer* 40(9):34–41
48. de Silva R et al (2007) Kirchhoff's pen: a pen-based circuit analysis tutor. In: Proceedings of the 4th Eurographics workshop on sketch-based interfaces and modeling 2007, Riverside. ACM, pp 75–82
49. Loughlin MM, Hughes JF (1994) An annotation system for 3D fluid flow visualization. In: Proceedings of IEEE conference on visualization, Washington, DC
50. Gennari L et al (2005) Combining geometry and domain knowledge to interpret hand-drawn diagrams. *Comput Graph* 29(4):547–562
51. Dachselt R, Frisch M, Decker E (2008) Enhancing UML sketch tools with digital pens and paper. In: Proceedings of the 4th ACM symposium on software visualization 2008, Ammersee. ACM, pp 203–204
52. Chen Q, Grundy J, Hosking J (2008) SUMLOW: early design-stage sketching of UML diagrams on an E-whiteboard. *Softw Pract Exp* 38(9):961–994
53. Shah D, Schneider J, Campbell M (2010) A robust sketch interface for natural robot control. In: IEEE/RSJ international conference on intelligent robots and systems, Taipei

54. Zhu B et al (2011) Sketch-based dynamic illustration of fluid systems. *ACM Trans Graph* 30(6):1–8
55. Zeleznik RC, Herndon KP, Hughes JF (2007) SKETCH: an interface for sketching 3D scenes. In: ACM SIGGRAPH 2007 courses, San Diego. ACM, p 19
56. Shi G, Zhang Y (2010) An improved SVM-HMM based classifier for online recognition of handwritten chemical symbols. In: Chinese conference on pattern recognition, Chongqing
57. Yang Z, Guangshun S, Kai W (2010) A SVM-HMM based online classifier for handwritten chemical symbols. In: 20th international conference on pattern recognition, Istanbul
58. Chan K-F, Yeung D-Y (2000) Mathematical expression recognition: a survey. *Int J Doc Anal Recognit* 3(1):3–15
59. Tapia E, Rojas R (2004) Recognition of on-line handwritten mathematical expressions using a minimum spanning tree construction and symbol dominance. In: Lladós J, Kwon Y-B (eds) *Graphics recognition. Recent advances and perspectives*. Springer, Berlin/Heidelberg, p 329–340
60. Belaid A, Haton J-P (1984) A syntactic approach for handwritten mathematical formula recognition. *IEEE Trans Pattern Anal Mach Intell PAMI-6(1)*:105–111
61. Koschinski M, Winkler HJ, Lang M (1995) Segmentation and recognition of symbols within handwritten mathematical expressions. In: International conference on acoustics, speech, and signal processing, Detroit
62. Juchmes R, Leclercq P, Azar S (2005) A freehand-sketch environment for architectural design supported by a multi-agent system. *Comput Graph* 29(6):905–915

## Further Reading

- Buxton2007. Buxton B (2007) *Sketching user experiences: getting the design right and the right design*, 1st edn. Morgan Kaufmann, San Francisco, p 448, ISBN-13 978-0123740373
- Jorge2011. Jorge J, Samavati F (eds) (2011) *Sketch-based interfaces and modeling*, vol XII, 1st edn. Springer, London/New York, p 402. ISBN 978-1-84882-811-7

---

## Part G

### Evaluation and Benchmarking

The final part of this handbook deals with the problem of benchmarking. Benchmarking provides a fundamental way of evaluating various proposed solutions against the state of the art. This is typically done for one of three reasons. First to see if a given system or process meets specific performance criteria in terms of accuracy, speed, or, more generally, costs. Performance criteria are set up and used to check whether the systems pass the performance test. Second, systems may be evaluated to determine the best at a particular task...to evaluate the state of the art. In this case, datasets and metrics are set up and solutions compete against each other to determine a winner. In the academic community, competitions such as these are often set up and are very successful at driving research, and motivating the development of new algorithms. Finally benchmarking can be used to simply evaluate current approaches against the challenges of the problem itself to know where effort should be put to improve performance. Such benchmarks are common internally during the research process where development datasets provide both regression data to make sure systems are still performing adequately and challenge data, representing yet unsolved problems.

In ►[Chap. 29](#) (Datasets and Annotations for Document Analysis and Recognition), Ernest Valveny focuses on data set problems, describing various evaluations and datasets that are common in the community, what the needs are for new datasets and the challenges of building them. In ►[Chap. 30](#) (Tools and Metrics for Document Analysis Systems Evaluation), Volker Märgner and Haikal El Abed wrap up the handbook by providing a survey to tools and performance metrics used in the field. Over the past decade we have seen a consolidation of such tools and definite sign that the community is maturing.

There is also a lot to be learned in this area, especially from related communities. While data is not cheap to produce, providing quality datasets for researchers to work on has proven time and again to be one of the most valuable resources to motivate the community to address and solve its most challenging problems.

---

# Datasets and Annotations for Document Analysis and Recognition

29

Ernest Valveny

## Contents

Introduction.....	984
General Issues in the Design and Creation of Datasets.....	985
Data Collection and Annotation.....	987
Real Data vs. Synthetic Data.....	987
Collection and Annotation of Real Data.....	988
Generation of Synthetic Data.....	990
Formats and Standards.....	992
Public Databases for Document Analysis and Recognition.....	992
Document Imaging.....	993
Page Analysis.....	994
Text Recognition.....	996
Graphics Recognition.....	996
Other Applications.....	999
Conclusion.....	1001
Cross-References.....	1002
Notes.....	1002
References.....	1004
Further Reading.....	1009

---

## Abstract

The definition of standard frameworks for performance evaluation is a key issue in order to advance the state-of-the-art in any field of document analysis since it permits a fair and objective comparison of different proposed methods under a common scenario. For that reason, a large number of public datasets have emerged in the last years. However, several challenges must be considered when creating such datasets in order to get a sufficiently large collection of

---

E. Valveny

Departament de Ciències de la Computació, Computer Vision Center, Universitat Autònoma de Barcelona, Bellaterra, Spain

e-mail: [ernest@cvc.uab.es](mailto:ernest@cvc.uab.es); [Ernest.Valveny@uab.cat](mailto:Ernest.Valveny@uab.cat)

representative data that can be easily exploited by the researchers. In this chapter we review different approaches followed by the document analysis community to address some of these challenges, such as the collection of representative data, its annotation with ground-truth information, or the representation using accepted and common formats. We also provide a comprehensive list of existing public datasets for each of the different areas of document analysis.

---

**Keywords**

Benchmarking • Generation of synthetic data • Ground-truthing • Performance evaluation • Public datasets

---

## Introduction

Nowadays, performance evaluation using standard datasets is a common practice in document analysis. This is the only way to do a fair comparison of existing algorithms as has been remarked in several chapters of this book. However, the creation of such standard datasets can be a costly process in terms of data selection, acquisition, and annotation. Thus, throughout the years the research community has devoted a big effort not only to the generation of public collections of data but also to the development of tools and frameworks that could make this process much easier.

The first works on standard datasets related to document analysis date back to the early 1990s mainly in the area of OCR and zone segmentation motivated by the need of “common data sets on which to develop and compare the performance of the algorithms” [50] and the evidence that “it is time for a series of comprehensive standard document databases to be constructed and made them available to researchers” [50]. This line of research was rapidly extended to other applications of document analysis, such as binarization, vectorization, symbol recognition, table recognition, and signature verification, leading to a large number of public datasets that will be presented and discussed later in this chapter. Probably, the best evidence of this effort is the increasing number of competitions related to different areas of document analysis that are being organized in the framework of a number of international conferences and workshops.

Although every particular field of document analysis has its own properties, there are some common issues to be taken into account in the design and creation of any dataset independently of the domain of application. They are mainly related to the selection, acquisition, and annotation of data and to the different ways of storing information and organizing the dataset. In the next section we will briefly introduce them. They will be discussed more deeply in the rest of the chapter. In particular, section “[Data Collection and Annotation](#)” is devoted to the selection, collection, and annotation of data. This is probably the most costly process (in terms of time and human resources) in the creation of any dataset. Therefore, a number of protocols, frameworks, and tools have been proposed to alleviate this effort. They include efficient ways to label either manually or semiautomatically large amounts of real data. An alternative is the generation of synthetic data. In this case, it is

necessary that the definition of models of degradation is able to generate artificial data as similar as possible to real data. Later, section “[Formats and Standards](#)” deals with the representation and storage of data (both input data and annotation data), making emphasis on those efforts undertaken to establish standard formats. Section “[Public Databases for Document Analysis and Recognition](#)” will include a comprehensive list of existing public datasets grouped by domain of application. It will contain a brief description of the main characteristics of each dataset and links on how to obtain them. Additionally, some specific issues that can concern the creation of datasets for each particular domain, such as ground-truth information, the organization of the datasets, or the protocol of evaluation, will also be discussed. Finally, section “[Summary](#)” will summarize the main ideas presented throughout the chapter, drawing the main conclusions about the current state of development of standard datasets for document analysis and also pointing out some challenges for future work. Finally, it is worth noticing that the creation of standard datasets is only one of the necessary elements for benchmarking and evaluating document analysis algorithms. The other required elements, mainly evaluation metrics and protocols, will be discussed in ►[Chap. 30](#) (Tools and Metrics for Document Analysis Systems Evaluation).

---

## General Issues in the Design and Creation of Datasets

A set of general issues should be considered in the process of generation of any standard dataset for performance evaluation. They cover all steps necessary for the creation of a dataset: data collection, ground-truth definition, and data representation, storage, and organization. In the following they will be briefly introduced, being further discussed in the next sections of the chapter.

**Selection of Data:** A requirement of any dataset intended to be used in performance evaluation is being realistic and representative of the set of images encountered in real-world applications. Being realistic implies that the set of images included in the dataset should be as similar as possible to real images. Obviously this is easy to achieve if the dataset contains images directly taken from real applications. However, as it will be further discussed later in this section, the cost of acquisition and annotation of real data sometimes forces the use of synthetic data. In these cases, realism of the dataset relies on the existence of accurate models of data generation – see section “[Generation of Synthetic Data.](#)”

Being representative means that the dataset should contain a balanced mix of all the classes of documents or entities that exist in a given application domain. How to achieve this balance depends very much on the application. For instance, in datasets for page segmentation, the emphasis should be put in collecting pages from many different types of documents including different combinations, layouts and formats of text, graphics, figures, tables, and photographs. However, in a different domain such as word recognition, the importance will be on issues such as having a realistic balance of all the words in the vocabulary or including

multiple handwriting styles through the enrollment of different writers. In this way, for every domain a set of important properties can be defined. This will be illustrated in section “[Public Databases for Document Analysis and Recognition](#)” where a description of the data included in existing datasets will be provided. The existence of multiple datasets for a given domain can be seen as the consequence of the difficulty of gathering all the types of data in a single dataset. Thus, in some cases, multiple datasets can be seen as complementary (more than redundant) in order to get a complete set of representative images.

The presence of noise and distortion is intrinsic to document processing. Thus, the creation of datasets should also guarantee the inclusion of realistic and representative degraded images, that is, images with a right balance of types and levels of degradation similar to those encountered in the real world. Sources of degradation and distortion can be very diverse, depending on the application domain: acquisition noise, manipulation, aging effects, handwriting variability, different layouts of the objects in the document, etc. However, sometimes it can also be interesting to generate challenging datasets with extreme levels of noise, higher than in real images, in order to push the limits of existing methods further away.

**Data Acquisition/Generation:** Once decided which kind of data will be included in a dataset, the next step is collecting a sufficiently large set of data according to those requirements. What a sufficiently large means is not easy to determine. It will also depend on each application domain and will rely on the already discussed properties of being realistic and representative. Collecting and annotating large sets of data is an extremely costly task. Thus, in some cases, dataset developers have decided to generate data in a synthetic way using some model of distortion applied to clean images. Although this makes collection and annotation of data much easier, it can go against the property of realism of data. Advantages and drawbacks of real vs. synthetic data, as well as several models of distortion proposed in the literature, will be discussed in the next section.

**Ground-Truth Definition:** A dataset for performance evaluation is not just a set of images gathered in a directory. Every image must be labeled with enough information to permit the comparison of the real output of an algorithm to the desired result for that image. This information is what is commonly called ground-truth. Therefore, the design of a dataset must also include the definition of which information is associated to every image and how this information will be represented. Ground-truth is something very specific to every application domain, for instance, bounding box coordinates for zone segmentation, a string of characters for text recognition, and the set of foreground pixels for binarization. In addition, not in every field it is absolutely clear which is the best way of defining and representing the ground-truth. In some areas, it has been a subject of intense debate among researchers. The different alternatives of ground-truth for each application domain will be presented and discussed in section “[Public Databases for Document Analysis and Recognition](#).” Some formats that have been proposed to represent the ground-truth will be described in section “[Formats and Standards](#).”

**Ground-Truth Annotation:** As pointed out before, every image of the dataset must be annotated with the ground-truth information. Doing this manually on a large set of images can be a very tedious and costly task and may be prohibitive if the ground-truth is complex. For that reason, dataset developers have used different strategies to reduce this cost. One way is obviously to use data generated synthetically. Then, the ground-truth can be obtained automatically. In real images, some strategies also exist, such as developing interactive and/or collaborative tools to help in the process of annotation or relying on the result of existing document analysis algorithms so that the user only has to confirm or correct the proposed ground-truth. These strategies will be discussed in more detail in the next section.

**File Formats:** Another important issue in the design of a dataset concerns the format of the files used to store images and represent the ground-truth. These formats should be well-known to the researchers and easy to manipulate to help in the dissemination and utilization of the dataset. In the case of input images, usual image formats, such as TIFF or JPEG, are commonly used, except in some particular applications as, for instance, online recognition. In the case of the ground-truth, there are more options, although XML has become a powerful pseudo-standard. Different specific formats proposed both for input images and ground-truth representation will be introduced in section “[Formats and Standards](#).”

**Structure and Organization of the Database:** Finally, all the elements related to a dataset (basically images and ground-truth) have to be put together, organized, and made available to the researchers in a way that permits an easy access and manipulation. Sometimes images are organized into different subsets according to several criteria, such as the category of the document, the degree of noise, or the difficulty of analysis. In any case, it is highly recommended to provide standard training and validation/test sets along with recommendations of use to permit a fair comparison of all algorithms executed on the dataset. These topics are very specific to each dataset and will be discussed in section “[Public Databases for Document Analysis and Recognition](#).”

---

## Data Collection and Annotation

### Real Data vs. Synthetic Data

There are basically two possibilities for collecting data: to use real data or to generate synthetic data.

Clearly, the main advantage of using real data is that it permits to evaluate the algorithms with the same kind of images encountered in real applications. Thus, evaluation can be a very good estimate of performance in real situations. However, manually collecting a large number of real images is a great effort that can be unaffordable in some cases. In addition, the annotation of these images with their corresponding ground-truth is also very costly in terms of time and human resources,

and errors can easily be introduced by manual annotation. Another disadvantage in some domains can be the difficulty of having easy access to a sufficient number of real images. Sometimes, confidentiality issues can make it difficult to provide public and free access to some collections of real data. Moreover, it is not easy to quantify the degree of noise in a real image. Then, it is not possible to define a ranking of difficulty of images according to the degree of noise.

The alternative to real data is to develop automatic methods to generate synthetic data. Clearly, the main advantages of this approach are that it allows to generate as many images as necessary and that the annotation of images with the ground-truth is also automatic. Then, manual effort is reduced. In addition, images generated using these methods can be easily classified according to the type and degree of noise or degradation applied, permitting to assess the reduction in performance with increasing degrees of image degradation. However, the main difficulty is to succeed in the development of models that could generate data as similar as possible to real data taking into account all kinds of noise and deformations. In some domains, this can be relatively easy, but in some other domains, this is really a challenging task.

## **Collection and Annotation of Real Data**

The annotation of large sets of real data is always a costly process. In the literature one may find basically two kinds of approaches to reduce this effort. The first one still relies on using some interactive and/or collaborative tool to help in the process of annotation. It still requires a significant amount of human effort but tries to make it more comfortable and reduce the number of errors. The second approach tries to take advantage of existing techniques in document analysis. It consists in applying such existing methods to generate a first version of the ground-truth that later is revised and, if needed, corrected by a human user.

### **Tools for Manual Annotation of Data**

Some of the first efforts in creating datasets for zone segmentation and text recognition relied heavily on collaborative human intervention [51, 58] for ground-truthing. This was a really costly process. For instance, a rough estimation of the time required to fully annotate 1 page was about a bit more than 1 h per page [51], only for zone segmentation. Moreover, manual annotation usually requires that more than one person creates and/or validates the ground-truth of a given page.

In order to reduce the amount of time required, several interactive tools have been developed and described in the literature. A significant group of such tools [3, 30, 77] deal with the definition and edition of zones in a document. In these tools zones can be represented in different ways such as rectangles, isothetic, or arbitrary polygons. They also permit to associate some kind of information to every zone, such as the type of contents, geometric and semantic attributes, or relation with other zones, for instance, reading order. All this information is stored using some specific format, usually based on XML. Most of these tools were originally conceived to annotate zone information for page segmentation evaluation. However, zones are

a common way to represent elements in document analysis. Therefore, a flexible representation of the zones and their associated information would permit to use these tools for other applications, such as logo detection or graphics recognition.

In contrast to zone-based annotation, many other applications (such as binarization or segmentation) require labeling at pixel level. PixLabeler [60] provides a tool to define such a ground-truth. Pixel labels (number and name) are totally configurable. Labeling of pixels can be done either manually or automatically using specific image-processing operations.

Graphics recognition is another domain where the complex layout and diversity of entities make annotation a costly process. In this context, a collaborative tool to annotate graphics documents has been developed in the framework of the project EPEIRES (available at <http://www.epeires.org>) for the tasks of symbol recognition and localization. Being a collaborative tool, users can contribute their own images, annotate documents, or validate the ground-truth labeled by other users.

For text recognition, the labeling is just the transcription of the text. It is difficult to develop tools to help in the process of transcription. Sometimes, an initial automatic transcription is used as a basis for correction (see next section). An alternative is to use predefined forms or templates that a selected group of users has to replicate. In this case, the ground-truth is implicit in the template, and the correspondence between the writing of the users and the template is usually easy to find. When this is not the case, for instance in online drawings [25], interactive tools can also be used.

Some interactive tools also exist to create ground-truth for table recognition [24, 65] that permit to easily mark row and column separators, indicate cells spanning various rows or columns, and represent the table as a hierarchical structure.

## Semiautomatic Ground-Truthing

In this section we will describe several strategies that have been used in order to somehow automatize the process of ground-truth creation. These strategies depend very much on the application domain of the dataset, but, in general, they consist in applying either some baseline method that permits to automatically obtain a first version of the annotation that later is revised by a human user, either applying some pre-processing steps that allow to extract some components or information that makes the annotation easier. In some cases, they also imply some constraints or requirements in the acquisition of data.

One generic method to get automatically the ground-truth was proposed in [29] when images in the dataset are obtained by transforming somehow real ground-truthed images. In this case, images were generated by printing, and later, faxing, photocopying, and re-scanning original images with geometric ground-truth available at word and zone level. The ground-truth of the new generated images was obtained putting in correspondence a set of feature points in the original and the new generated images and finding the best affine transformation between the two sets of points. Feature points were the coordinates of the bounding boxes of groups of connected components obtained by joining nearby components.

Handwritten text recognition is a specific domain where semiautomatic ground-truthing makes special sense. In most cases, ground-truth simply consists of the labels of the characters or words. Many datasets have been constructed by asking a group of people to transcribe a predefined text. In these cases, a careful design of the acquisition conditions can permit an easy extraction of the text and the creation of the ground-truth by just aligning the input text with its expected predefined transcription. Only some basic image-processing operations, such as binarization, skew correction, or sometimes, simple character or line segmentation, are required [32, 75]. At the end of the process, a human user can verify the ground-truth to detect possible errors in the transcription or in the automatic extraction of data. A bit more complicated is the case where text can consist of several lines. Then, the alignment of the transcription with the input text has to take into account line breaks that are usually marked manually [39]. A similar approach is also used for annotating handwritten music scores [16] where the handwritten symbols are easily extracted after staff removal.

When the data is obtained from real scanned forms, there is no expected transcription to be matched with the input text. Then, a recognition engine trained on a different dataset can be applied to obtain a first version of the ground-truth to be manually validated and corrected [15, 28]. This strategy has also been used in other domains such as recognition of mathematical expressions [18] or chart recognition [76].

Binarization is another specific domain where it is not easy to define and create the ground-truth. In [44] the authors propose an alternative for getting the ground-truth that takes advantage of the intrinsic properties of text. The image is first skeletonized in order to obtain a silhouette of one pixel wide of the characters. The user checks and corrects the result of the skeletonization. Then, the skeleton is dilated in successive iterations until it reaches the edges of the character in the original image.

## Generation of Synthetic Data

The generation of synthetic data to be used for performance evaluation of real systems depends on the existence of accurate models able to reproduce noise and variability of real data. Several models have been proposed in the literature. They can be broadly grouped in two main categories. A first group of models try to reproduce the distortion yielded by the process of acquisition of the image (printing, photocopying, scanning). A second group aims at creating models of the variability of the data. Variability can be due to several factors, such as handwriting, noise, or different spatial layout of objects in the image.

A first attempt to create a model of the distortion due to the acquisition process is presented in [6]. The model has several parameters modeling diverse factors that can influence the acquisition of an image such as the spatial sampling rate (digitizing resolution), blurring, digitizing threshold, sensitivity of pixel sensors,

skew, stretching, and translation. In a similar way, Kanungo proposed another framework for both global and local models of document degradation due to acquisition factors [27]. The global model focused on the degradation motivated by scanning thick books. The model takes into account the bending of the document that produces a perspective distortion and a change of intensity and blurring on the curved parts of the page. The local model accounts for pixel inversion (from foreground to background and vice versa), due to lighting changes, sensitivity of the sensors, and thresholding, and for blurring due to the scanner optical system. Apart from this theoretical models, simple image transformations (such as salt and pepper noise, rotation, blurring, erosion, dilation, slant shrink, or downsampling) have also been applied to generate noisy images in different domains such as line drawings [78] or music scores [10].

The second group of models is related to the generation of handwriting. We can distinguish different strategies to approaching this problem. Several works [52, 61] have explored the definition of explicit models of the human movement of the hand and the pen. These models permit to determine the trajectory of the pen when writing different letters or strokes. Alternatively, other works try to model, not the explicit trajectory, but the result of that trajectory that is, the shape of the stroke or the character. For instance, B-splines can be used [74] to define a generative model of each letter (decomposed into three parts, the head, the body, and the tail) where the distribution of the control points of the B-splines is learned from a set of existing samples. A deformable model is used to smoothly join the tail and the head of two adjacent letters.

In a completely different approach, other works do not try to define a model of handwriting but just to generate new samples similar to other existing ones. This can be done in different ways. One possibility for generating individual characters is using point correspondence among several samples [42]. A template of the character is created as the mean of all the samples. Then, a point correspondence can be established between every sample and this template. New samples can be generated by moving original points along the line that joins every point in the sample with the corresponding point in the template. Distortion can also be generated at the line level [71]. In this case four kinds of pixel level transformations (shearing, bending, and horizontal and vertical scaling) are applied globally to the whole line of text. The amount of transformation applied to every pixel depends on its horizontal position and is determined by a set of underlying functions based on the cosine. Additionally, horizontal and vertical scaling is also applied to the connected component level. Finally, thinning/thickening effects are applied to the whole line. At the word level for online handwriting, the work presented in [63] finds patterns between velocity of writing and the shape of corners, the slant and the size. Distortions are generated by changing the velocity pattern from an original image of the word. There is a limit in the amount of distortion based on keeping some features of the word, such as ascenders and descenders.

A last strategy [23] that has been used for the generation of handwriting consists in the concatenation of models of characters or groups of characters. The basic idea

---

is decomposing a word into a number of basic units (common groups of characters) trying to minimize the number of cuts. New samples of the word are generated by concatenating existing samples of the characters trying to optimize the transition between groups.

In other domains, specific methods have also been proposed with the goal of generating synthetic images as similar as possible to real ones. For instance, in the case of binarization [47], synthetic images are generated by overlapping a clean image with noisy ones. These noisy images correspond to the scanning of blank pages from the eighteenth century that contain usual artifacts such as stains, strains, or uneven illumination.

In addition to noise and degradation at the pixel level, another important source of variability in document images is that of the spatial layout of entities in the document. However, not a lot of attention has been devoted to this matter in the generation of synthetic data. Grammars have been used [38] to generate sketched mathematical expressions. In the field of graphics recognition, a rule-based system [11] has been proposed to generate graphic documents (architectural floor plans and electronic diagrams) based on overlapping entities (in this case, symbols) on a set of predetermined, fixed backgrounds, following a set of domain-specific rules that define where and how symbols can be placed.

---

## Formats and Standards

XML has become a de facto standard to represent ground-truth data in many existing datasets. Many datasets use a specific XML schema for ground-truth in several domains. In some cases some specific standard formats derived from XML have been defined and used to represent ground-truth as it is the case of ALTO (Analyzed Layout and Text Object, at <http://www.loc.gov/standards/alto/>) or PAGE [53] for page analysis, InkML for online text recognition [25], MathML for representing mathematical expressions [45], or SVG (Scalable Vector Graphics, at <http://www.w3.org/Graphics/SVG/>) for graphics recognition.

However, some efforts have also been devoted to creating specific formats in some applications, such as DAFS [17] and RDIFF [77] for page analysis, UNIPEN [21] for online recognition, and VEC for graphics recognition [49].

---

## Public Databases for Document Analysis and Recognition

This section intends to provide a comprehensive summary of available public datasets for document analysis tasks, grouped in five categories: document imaging (mainly binarization and dewarping), page analysis (basically, page/zone segmentation, and document classification), text recognition (both printed and handwritten, offline and online), graphics recognition (including vectorization,

symbol recognition and spotting, music scores, chart and sketch recognition), and other applications (table recognition and understanding, mathematical expressions, signature verification). For each of these categories, several generic aspects concerning the generation of datasets for that domain are discussed, such as collection and acquisition of data and ways of defining the ground-truth for that particular domain. A table summarizing the public datasets for each category is included with information such as the type of data included in the dataset, the size, training, and test sets. It is worth noticing that some of these datasets are also referred, and sometimes further described, throughout the book in the chapters devoted to each particular subfield of document analysis. In some chapters (see for instance, ►Chap. 7 (Page Similarity and Classification)), other non-public datasets are also mentioned to show the state-of-the-art in a particular field. In this section, however, we will only include public datasets. In a notes section at the end of this chapter, we also provide the links to the websites hosting the datasets. Reference to papers is only included when they cannot be found in an institutional website.

## Document Imaging

### Binarization

Two main issues arise concerning the creation and annotation of datasets for binarization: the definition of the ground-truth and the collection/generation of the data.

The definition of the ground-truth is closely related to the kind of evaluation approach. In [44] different approaches to the evaluation of binarization algorithms are distinguished. The first category is a human-centered evaluation where binarization is evaluated by visual inspection of the result according to a set of predefined criteria. Therefore, there is no need for ground-truth. However, the evaluation is subjective and time consuming and lacks robustness. The second category is the so-called goal-directed evaluation where the result of binarization is evaluated through its impact in the performance of a high-level module, usually OCR. In this case, ground-truth must be defined in terms of the output of this high-level module (i.e., usually at character level). In the third category the evaluation is done directly on the result of the binarization at pixel level, and, therefore, ground-truth must also be defined at pixel level. The definition of the ground-truth at character level is usually easier and more objective than defining it at pixel level. It also avoids having to define a set of subjective criteria for evaluating the performance. However, the evaluation could depend on the specific method selected for implementing the high-level module. The definition of the ground-truth at pixel level is the only way to evaluate the output of the binarization itself, and this is how ground-truth is defined in all public datasets described in this section. However, it is not easy to define this kind of ground-truth since it is very subjective. A study analyzing the influence of different ways of ground-truthing in performance evaluation of binarization can be found in [66].

**Table 29.1** Binarization datasets

Name	No. images	Type of images
DIBCO'09 [19] <sup>1</sup>	10	Real images
H-DIBCO'10 [54] <sup>2</sup>	10	Real images
DIBCO'11 [55] <sup>3</sup>	16	Real images
ICFHR'10 Contest [47] <sup>4</sup>	60 training/210 test	Synthetic images

**Table 29.2** Image dewarping datasets

Name	No. images	Type of images
Document Image Dewarping Contest [64] <sup>5</sup>	102	Real images

The collection of large amounts of real data is not easy, especially when ground-truthing has to be done at pixel level, since this is a time-consuming process. Therefore, existing datasets rely mainly on using synthetic images [47] or on semiautomatic methods [44] for the generation of ground-truth.

Table 29.1 summarizes existing datasets for evaluation of binarization methods. They have been generated for their use in competitions organized in the framework of international conferences, such as ICDAR (in the case of the series of DIBCO datasets) and ICFHR. In the DIBCO dataset, real images (including machine-printed and handwritten text) are used, obtained from collections of different libraries, containing a set of representative degradations such as variable background intensity, shadows, smear, smudge, low contrast, bleed-through, or show-through. The ICFHR 2010 dataset consists of synthetic images generated by the combination of clean images with noisy ones.

### Dewarping

For page dewarping, one public dataset exists, created in the context of the Document Image Dewarping Contest organized at CBDAR'07 [64]. It is composed of real images from several technical books captured by an off-the-shelf digital camera in a normal office environment, binarized using a local adaptative thresholding technique. Three different kinds of ground-truth are provided. Two of them (text lines and text zones) are intended to serve as internal evaluation of the methods as most of them perform an intermediate step consisting in the detection of curved text lines. The third kind of ground-truth information is the ASCII transcription of the text in the document that can be used to perform a global goal-oriented evaluation based on OCR accuracy. Details are summarized in Table 29.2.

### Page Analysis

The evaluation of page analysis algorithms (mainly physical layout analysis or zone segmentation) has received a lot of attention throughout the years. An important issue that has been largely discussed in many works is the type of ground-truth

information required, as mentioned in ►Chap. 5 (Page Segmentation Techniques in Document Analysis). The answer to this question has influence in many aspects of the evaluation framework, including the goal of the evaluation process, the generation of the dataset, or the evaluation measures. Three alternatives are described in the literature: text-based, pixel-based, or zone-based ground-truth information.

Text-based evaluation uses the transcription of the document as a ground-truth. The main advantage of this approach is simplicity as the labeling information is easy to obtain and store, and there is no need for a specific output format of zoning systems. Although it was used in some of the early works on zoning evaluation [26], it was later dismissed due to multiple drawbacks. It is not a direct evaluation method for page segmentation results, and thus, it does not permit to provide any characterization of the segmentation errors. In addition, it can only deal with text regions and requires the zoning algorithm to be part of an OCR system.

Pixel-based approaches [60] define the ground-truth in terms of pixel labels. Every pixel is assigned a different label depending on the zone they belong to. In this way, they do not depend on an accurate definition of the shape of the zones and can handle better noise or graphics elements. They are more appropriate to evaluate zone classification (label assigned to every pixel) than zone segmentation (segmentation of the document into zones).

Zone based is the most used approach for representing ground-truth information. The ground-truth is defined in terms of the set of physical zones that compose the document. The shape of the zones has been described mostly using rectangles [30], but also isothetic polygons [4] or arbitrary polygons [69]. In this way the ground-truth contains all the information needed to evaluate zone segmentation as a stand-alone process and characterize the output of the system according to different segmentation errors. In addition, a label can be added to every zone describing its contents and allowing for evaluation of zone classification.

In any case, one of the critical issues is the criteria followed to define the zones or the regions in the ground-truth. In some cases, a set of rules is defined to limit the subjectivity of ground-truthers when determining the zones of a document and their shape [41, 69].

XML has been mostly used to define the format to store the ground-truth files, although other specific formats have also been defined such as DAFFS [31] or RDIFF [77]. In some cases there is also an internal graph structure [30, 69] that permits to represent the logical order of the zones.

Table 29.3 summarizes the details of existing public datasets in terms of the type of documents included in the datasets, ground-truth information, and tasks for which the dataset could be used (segmentation, page classification, or logical layout analysis). All datasets contain real images. Only one dataset, the UVA dataset [69], includes color images. A slightly different dataset is that released for the Book Structure Extraction Competition held at ICDAR'09 and ICDAR'11 [12]. This dataset is intended to evaluate the detection of the logical book structure in terms of the table of contents.

**Table 29.3** Page analysis datasets

Name	No. images	Type	Ground-truth	Task
UW-III	1,600	Technical	Rectangle	Segmentation/OCR
ICDAR'09 <sup>6</sup>	1,240	Magazines/ journals	Isothetic pols.	Segmentation
ICDAR'11 [5] <sup>7</sup>	100	Historical	Isothetic pols.	Segmentation
MARG <sup>8</sup>	3,337	Journals	Rectangle	Segmentation/ classification
UvA [69] <sup>9</sup>	1,000	Color magazines	Arbitrary shapes	Segmentation/ logical order
Book structure [12] <sup>10</sup>	1,040	Books	ToC	Logical structure

## Text Recognition

Text recognition has traditionally been the main activity within the field of document analysis. Therefore, there has been a lot of work around the creation of datasets for evaluation of text recognition methods. Tables 29.4 and 29.5 compile a comprehensive list of all these existing datasets for text recognition. They are organized according to the writing script since usually, methods for recognition follow different approaches depending on the particularities of each script (see ►Chaps. 9 (Language, Script, and Font Recognition), ►13 (Middle Eastern Character Recognition), and ►14 (Asian Character Recognition)). The table also includes the type of acquisition of images (online/offline) and a summary of the contents and the number of samples contained in each dataset. For more details, we refer the reader to the original paper describing the dataset or the dataset website. Some of these datasets have also been described in the chapters devoted to text recognition (see ►Chaps. 12 (Continuous Handwritten Script Recognition), ►13 (Middle Eastern Character Recognition), and ►14 (Asian Character Recognition)). It is worth mentioning that several of these datasets have been used in international competitions or evaluation campaigns.

Additionally, in Table 29.6 we list two existing datasets for evaluation of writer identification for Arabic and Latin text. Both datasets have been used in competitions organized in the framework of last ICDAR conference.

Finally, and also related to text recognition, we present in Table 29.7 one dataset that has been used to evaluate word and line segmentation in ICDAR competitions.

## Graphics Recognition

The field of graphics recognition is very diverse, covering different topics such as vectorization, symbol recognition and spotting, drawing interpretation, and music score analysis. Therefore, the existing datasets also reflect this diversity. The first efforts in providing standard datasets for graphics recognition were related to vectorization and arc detection and segmentation, in the context of the GREC workshop.

**Table 29.4** Text recognition datasets (1)

Name	Script	Mode	Contents	Volume
UW-III dataset [51] <sup>11</sup>	Latin	Offline	Printed text	≈1,000 pages
Noisy text [35] <sup>12</sup>	Latin	Offline	Noisy printed text	3,305 pages
ISRI-UNLV [58] <sup>13</sup>	Latin	Offline	Printed characters + OCR-based zone segmentation	≈2,000 pages
Sofia-Munich corpus [40]	Latin Cyrillic	Offline	Printed characters	2,306 pages
CENPARMI database [68]	Latin	Offline	HW digits	≈17K digits
CEDAR database <sup>14</sup>	Latin	Offline	HW characters, zip codes, digits, city names, state names	≈1,000 words, 10K codes, 50K characters
NIST database <sup>15</sup>	Latin	Offline	HW digits, characters, sentences	≈810K characters
MNIST database <sup>16</sup>	Latin	Offline	HW digits	70K digits
IAM-database <sup>17</sup>	Latin	Offline	HW words/lines/sentences	82,227 words/9,285 lines
RIMES dataset <sup>18</sup>	Latin	Offline	HW words and lines	≈250K words
IAM-HistDB <sup>19</sup>	Latin	Offline	HW words/sentences – historical documents	6,543 lines/39,969 words
George Washington dataset [57] <sup>20</sup>	Latin	Offline	HW words for word spotting	20 pages/5,751 words
The Rodrigo database [62] <sup>21</sup>	Latin	Offline	HW lines – historical documents	≈20K lines/231K words
The Germana database [48] <sup>22</sup>	Latin	Offline	HW lines – historical documents	≈21K lines/217K words
IRONOFF database [72] <sup>23</sup>	Latin	Offline Online	HW characters, digits, words	≈32K characters/50K words
UNIPEN project [21] <sup>24</sup>	Latin	Online	HW characters, digits, words, text	≈16K digits, 300K characters, 160K words, 16K sentences
IAM-OnDB <sup>25</sup>	Latin	Online	HW sentences	≈86K words/13K lines
IAM-OnDo <sup>26</sup>	Latin	Online	HW varied documents: text, tables, formulas, diagrams, drawings, markings	1,000 documents

Symbol recognition has been another active field in proposing datasets for performance evaluation. Going beyond symbol recognition, symbol spotting appears as a challenging task for the graphics recognition community. Two datasets have been released in the last years that can be used for the evaluation of symbol spotting,

**Table 29.5** Text recognition datasets (2)

Name	Script	Mode	Contents	Volume
APTI database <sup>27</sup>	Arabic	Offline	Printed words	45,313,600 words
IFN/ENIT <sup>28</sup>	Arabic	Offline	HW words	≈26,400 city names
AHDB database <sup>29</sup>	Arabic	Offline	HW words and sentences	
IBN-SINA dataset <sup>30</sup>	Arabic	Offline	HW words – word spotting	≈1,000 subwords
ADAB database [13]	Arabic	Online	HW words	15,158 words
CENPARMI dataset [2]	Arabic	Offline	HW digits and words (from cheques)	23,325 words/9,865 digits
CENPARMI dataset [1]	Arabic	Offline	HW digits, numeral strings, isolated letters, words, dates	46,800 digits, 13,439 strings, 21,426 letters, 11,375 words, 284 dates
Hoda database <sup>31</sup>	Farsi	Offline	HW digits	102,352 digits
CENPARMI dataset [67]	Farsi	Offline	HW digits, numerical strings, legal amounts, letters, and dates	18,000 digits, 7,350 strings, 11,900 letters, 8,575 words, 175 dates
IFHCDB database <sup>32</sup>	Farsi	Offline	HW characters and numerals	52,380 characters/17,740 numerals
HIT-MW dataset <sup>33</sup>	Chinese	Offline	HW characters, documents	186,444 characters
SCUT-COUCH2009 dataset <sup>34</sup>	Chinese	Online	HW characters words	1,392,900 characters
CASIA dataset <sup>35</sup>	Chinese	Online Offline	HW characters, documents	≈3.9 million characters/ 5,090 pages
[7] <sup>36</sup>	Indian scripts	Online Offline	HW numerals and characters	45,948 numerals

**Table 29.6** Writer identification datasets

Name	Alphabet	Volume
Arabic WI contest [22] <sup>37</sup>	Arabic	54 writers/162 paragraphs
ICDAR'11 WI contest [36] <sup>38</sup>	Latin	26 writers/208 pages

**Table 29.7** Word and line segmentation datasets

Name	Alphabet	Contents	Volume
ICDAR'09 HW contest [20] <sup>39</sup>	Latin	Lines and words	300 images

the first one based on images of scanned real drawings and a second one based on a semi-automatic method of generation of drawings that has allowed the generation of an extensive dataset composed of synthetic images of architectural floor plans and electronic diagrams.

In the field of music interpretation, datasets exist for two tasks, namely, staff removal and writer identification in handwritten music scores. Finally, datasets also exist in the context of chart analysis, to evaluate the process at different levels:

**Table 29.8** Datasets for graphics recognition

Name	Domain	Tasks	Contents
Arc segmentation contests <sup>40</sup>	Engineering drawings	Arc segmentation	
SymbolRec <sup>41</sup>	Symbols	Symbol recognition	7,500 images of 150 symbols
NicIcon <sup>42</sup>	Symbols	Symbol recognition	Online/offline. 14 classes. 26,163 images
FPLAN-POLY [59] <sup>43</sup>	Architectural drawings	Symbol spotting	42 floor plan images + 38 query symbols
SESYD [11] <sup>44</sup>	Architectural and electronic drawings	Symbol spotting	2,000 images/≈40,000 symbols
CVC-MUSCIMA <sup>45</sup>	Music scores	Writer identification staff removal	1,000 scanned images (50 writers × 20 documents) + 12,000 synthetic images
Synthetic Score Database <sup>46</sup>	Music scores	Staff removal	Synthetic images from 32 generated music scores
CHIME Chart Image [76] <sup>47</sup>	Charts	Chart interpretation	200 real images + 3,200 synthetic images

vector level (lines and arcs), text level (blocks and words), and chart level (axis, titles, labels, data points).

Table 29.8 summarizes the main features of all these datasets.

## Other Applications

In this section we describe datasets for miscellaneous document analysis applications that did not fit in any of the previous sections.

Firstly, we encounter a set of datasets for text localization and recognition in images obtained from non-paper sources, such as web images or scene images acquired with a camera. Some of these datasets have been used in the series of competitions on robust reading in several editions of ICDAR starting in 2003. In addition, several other datasets have been created in the last years, mainly due to the rise of digital cameras that have led to the existence of a large amount of natural images containing text. The characteristics of these datasets are summarized in Table 29.9 and also referred to in ▶ Chap. 25 (Text Localization and Recognition in Images and Video).

In a second group there is a set of datasets for recognition of mathematical expressions, both symbols and relations among them, to derive the structure of the expression. Table 29.10 summarizes these datasets.

Another important group concerns datasets for signature verification, mostly used in competitions organized in the context of ICDAR and ICFHR. Two important issues to take into account in this domain (see ▶ Chap. 27 (Online Signature Verification))

**Table 29.9** Datasets for robust reading

Name	Type of images	Tasks	Contents
ICDAR'03 and ICDAR'05 [37] <sup>48</sup>	Scene images	Text localization, character recognition, and word recognition	
ICDAR'11 Challenge 1 <sup>49</sup>	Web and e-mail images	Text localization, segmentation and word recognition	420 images (3,583 words) for training and 102 (918 words) for test
ICDAR'11 Challenge 2 <sup>50</sup>	Scene images	Text localization and word recognition	485 images and 1,564 words
KAIST database <sup>51</sup>	Scene images	Text localization and segmentation	3,000 images
Google Street View dataset [73] <sup>52</sup>	Scene images	Word spotting	350 images
Street View House Numbers images Dataset <sup>53</sup>	Digits from natural images	Digit recognition	600,000 digits
IUPR dataset [9] <sup>54</sup>	Document pages	Line/zone segmentation, dewarping, text recognition	100 images
NEOCR dataset <sup>55</sup>	Scene images	Text localization and recognition	659 images
Chars74K dataset <sup>56</sup>	Characters from natural images	Character recognition	74,000 characters

**Table 29.10** Datasets for recognition of mathematical expressions

Name	Type of images	Contents
Infy [70] <sup>57</sup>	Printed text	108,914 words and 21,056 mathematical expressions. 688,580 character samples, from 476 pages of text
[18]	Printed text	400 images (297 real/103 synthetic). 2,459 displayed and 3,101 embedded expressions
MathBrush <sup>58</sup>	Offline handwritten	4,655 expressions
CROHME [43] <sup>59</sup>	Online	921 expressions for training and 348 for test
UW-III dataset [51] <sup>60</sup>	Offline	100 expressions from 25 pages
Hamex [56] <sup>61</sup>	Online	4,350 expressions
Marmot <sup>62</sup>	PDF documents	9,482 expressions

are the collection of a representative set of genuine and forgery signatures and the specific protocol for using reference, genuine and forgery signatures. Table 29.11 summarizes the datasets for signature verification. Some of them are further described in ▶Chap. 27 (Online Signature Verification).

Finally, the last group contains a pair of datasets for table processing, mainly table localization and cell segmentation. The first one uses the UNLV and UW-III datasets that have been described in section “Text Recognition.” The ground-truth for table processing has been generated and released<sup>70</sup> for 427 images in the UNLV dataset and 120 images in the UW-III dataset [65]. The second one has been

**Table 29.11** Datasets for signature verification

Name	Type	Domain	Writers	Genuine	Forgeries
GPDS-960 <sup>63</sup>	Offline	Western	960	24 per writer (23,049)	30 per writer (28,800)
MCYT [46] <sup>64</sup>	Offline	Western	330	5 per writer (1,650)	25 per writer (8,250)
	Online				
BioSecurId [14] <sup>65</sup>	Offline	Western	400	16 per writer (6,400)	12 per writer (4,800)
	Online				
SVC2004 <sup>66</sup>	Online	Western Chinese	100	20 per writer (2,000)	20 per writer (2,000)
SigComp09 [8] <sup>67</sup>	Offline	Western	12 (training) 100 (test)	60 (training)/ 1,200 (test)	1,860 (training)/ 753 (test)
	Online				
4NSigComp10 [33] <sup>68</sup>	Offline	Western	1	85 (training)/ 28 (test)	124 (training)/ 97 (test)
SigComp11 [34] <sup>69</sup>	Offline	Western	10 (Chinese)	240 (Chinese)/	400 (Chinese)/
	Online	Chinese	54 (Western)	1,330 (Western)	630 (Western)
Caltech	Camera based	Western	105	25–30 per writer	10 per writer

generated for an ICDAR’11 competition using PDF documents (19 for training, 3 for validation and 19 for test) containing in total 1,003 tables for training, 120 for validation and 968 for test.<sup>71</sup>

## Conclusion

The need of standard and public tools for a fair evaluation of all the existing methods in document analysis has become an evidence in the last years. The creation of representative datasets is the first step in this process. Thus, the number of such datasets has largely increased along the last years in all fields of document analysis. Currently, we can find at least one dataset to evaluate almost any document analysis algorithm. This will certainly help to boost the research results as any contribution can be easily put in the context of the current state-of-the-art.

However, the creation of datasets arises important issues that, in some cases or domains, are far from to be completely solved. Throughout this chapter we have discussed the more important ones, mainly concerned with the collection of a representative and sufficiently large number of images and with the annotation of such large data. We have analyzed the different approaches used for that, and which have allowed to generate really large annotated datasets.

As a result of all this work, many datasets have been generated in all fields of document analysis. We have summarized all the existing public datasets in an organized way, according to the domain and we have described the main characteristics of each dataset. We can see this large number of datasets as a consequence of mainly two factors: on one hand, the own evolution of each field, demanding every time for more complex and complete datasets to evaluate new advances in the field, and, on

the other hand, the diversity of each particular domain and, thus, the need of having datasets that cover all this diversity (types of documents, fonts, handwriting styles, layouts, noise, distortion, etc.). In this way, different aspects of the same problem can be evaluated, and the more challenging data can be identified. In this sense, probably, in many fields there is no need for more datasets but an identification of the most relevant and challenging subsets of data to focus new research on it.

---

## Cross-References

- [A Brief History of Documents and Writing Systems](#)
  - [Asian Character Recognition](#)
  - [Continuous Handwritten Script Recognition](#)
  - [Middle Eastern Character Recognition](#)
  - [Online Signature Verification](#)
  - [Page Segmentation Techniques in Document Analysis](#)
  - [Page Similarity and Classification](#)
  - [Text Localization and Recognition in Images and Video](#)
  - [Tools and Metrics for Document Analysis Systems Evaluation](#)
- 

## Notes

### Links to datasets for binarization

- <sup>1</sup> DIBCO'09: <http://www.iit.demokritos.gr/bgat/DIBCO2009/benchmark>
- <sup>2</sup> H-DIBCO'10: <http://www.iit.demokritos.gr/bgat/H-DIBCO2010/benchmark>
- <sup>3</sup> DIBCO'11: <http://utopia.duth.gr/ipratika/DIBCO2011/benchmark>
- <sup>4</sup> ICFHR'10 Contest: <http://users.dsic.upv.es/iaprtc5/icfhr2010contest/index.php>

### Links to datasets for image dewarping

- <sup>5</sup> Document Image Dewarping Contest: <http://www.dfki.uni-kl.de/shafait/downloads.html>

### Links to datasets for page analysis

- <sup>6</sup> ICDAR'09: <http://dataset.primaresearch.org/>
- <sup>7</sup> ICDAR'11: [http://www.prima.cse.salford.ac.uk:8080/ICDAR2011\\_competition/](http://www.prima.cse.salford.ac.uk:8080/ICDAR2011_competition/)
- <sup>8</sup> MARG: <http://marg.nlm.nih.gov/index2.asp>
- <sup>9</sup> UvA: <http://www.science.uva.nl/UvA-CDD/>

- <sup>10</sup> Book structure: <http://users.info.unicaen.fr/doucet/StructureExtraction/2011/>

### Links to datasets for text recognition

- <sup>11</sup> UW-III dataset: available on CD-ROM
- <sup>12</sup> Noisy Text: <http://www.cse.lehigh.edu/lopresti/noisytex.html>
- <sup>13</sup> ISRI-UNLV: <http://code.google.com/p/isri-ocr-evaluation-tools/>
- <sup>14</sup> CEDAR database: <http://www.cedar.buffalo.edu/Databases/index.html>
- <sup>15</sup> NIST database: <http://www.nist.gov/srd/nistsd19.cfm>
- <sup>16</sup> MNIST database: <http://yann.lecun.com/exdb/mnist/>
- <sup>17</sup> IAM-database: <http://www.iam.unibe.ch/fki/databases/iam-handwriting-database>

- <sup>18</sup>RIMES dataset: <http://www.rimes-database.fr/wiki/doku.php>
- <sup>19</sup>IAM-HistDB: <http://www.iam.unibe.ch/fki/databases/iam-historical-document-database>
- <sup>20</sup>George Washington dataset: <http://ciir.cs.umass.edu/cgi-bin/downloads/downloads.cgi>
- <sup>21</sup>The Rodrigo database: <http://prhl.itи.upv.es/page/projects/multimodal/idoc/rodrigo>
- <sup>22</sup>The Germana database: <http://prhl.itи.es/page/projects/multimodal/idoc/germania>
- <sup>23</sup>IRONOFF database: <http://www.irccyn.ec-nantes.fr/viardgau/IRONOFF/IRONOFF.html>
- <sup>24</sup>UNIPEN project: <http://www.unipen.org/products.html>
- <sup>25</sup>IAM-OnDB: <http://www.iam.unibe.ch/fki/databases/iam-on-line-handwriting-database>
- <sup>26</sup>IAM-OnDo: <http://www.iam.unibe.ch/fki/databases/iam-online-document-database>
- <sup>27</sup>APTI database: <http://diuf.unifr.ch/diva/APTI/>
- <sup>28</sup>IFN/ENIT: <http://www.ifnenit.com/index.htm>
- <sup>29</sup>AHDB database: <http://www.cs.nott.ac.uk/cah/Databases.htm>
- <sup>30</sup>IBN-SINA dataset: [http://www.causality.inf.ethz.ch/al\\_data/IBN\\_SINA.html](http://www.causality.inf.ethz.ch/al_data/IBN_SINA.html)
- <sup>31</sup><http://farsioocr.ir/farsi-digit-dataset>
- <sup>32</sup>IFHCDB database: <http://ele.aut.ac.ir/imageproc/downloads/IFHCDB.htm>
- <sup>33</sup>HIT-MW dataset: <http://sites.google.com/site/hitmwdp/>
- <sup>34</sup>SCUT-COUCH2009 dataset: <http://www.hcii-lab.net/data/scutcouch/EN/introduction.html>
- <sup>35</sup>CASIA dataset: <http://www.nlpr.ia.ac.cn/databases/handwriting/Home.html>
- <sup>36</sup><http://www.isical.ac.in/ijjwal/download/database.html>

#### Links to datasets for writer identification

- <sup>37</sup>Arabic WI contest: <http://wic2011.qu.edu.qa/>
- <sup>38</sup>ICDAR'11 WI contest: [http://users.iit.demokritos.gr/louloud/Writer\\_Identification\\_Contest/](http://users.iit.demokritos.gr/louloud/Writer_Identification_Contest/)

#### Links to datasets for word and line segmentation

- <sup>39</sup>ICDAR'09 HW contest: <http://users.iit.demokritos.gr/bgat/HandSegmCont2009/Benchmark/>

#### Links to datasets for graphics recognition

- <sup>40</sup>Arc segmentation contests: <http://www.cs.usm.my/arcseg2011/>, <http://www.cs.usm.my/arcseg2009/>, <http://www.cs.cityu.edu.hk/liuwy/ArcContest/ArcSegContest.htm>
- <sup>41</sup>SymbolRec: <http://iapr-tc10.univ-lr.fr/index.php/symbol-contest-2011/153>
- <sup>42</sup>NicIcon: <http://www.unipen.org/products.html>
- <sup>43</sup>FPLAN-POLY: <http://www.cvc.uab.cat/marcal/FPLAN-POLY/index.html>
- <sup>44</sup>SESYD: <http://mathieu.delalandre.free.fr/projects/sesyd/>
- <sup>45</sup>CVC-MUSCIMA: <http://www.cvc.uab.es/cvcmuscima>
- <sup>46</sup>Synthetic Score Database: <http://music-staves.sourceforge.net>
- <sup>47</sup>CHIME Chart Image: <http://www.comp.nus.edu.sg/tanc1/ChartImageDataset.htm>

#### Links to datasets for robust reading

- <sup>48</sup>ICDAR'03 and ICDAR'05: <http://algoval.essex.ac.uk/icdar/Competitions.html>
- <sup>49</sup>ICDAR'11 Challenge 1: <http://www.cvc.uab.es/icdar2011competition/>
- <sup>50</sup>ICDAR'11 Challenge 2: <http://robustreading.opendfki.de/wiki/SceneText>
- <sup>51</sup>KAIST database: [http://www.iapr-tc11.org/mediawiki/index.php/KAIST\\_Scene\\_Text\\_Database](http://www.iapr-tc11.org/mediawiki/index.php/KAIST_Scene_Text_Database)
- <sup>52</sup>Google Street View dataset: <http://vision.ucsd.edu/kai/svt/>
- <sup>53</sup>Street View House Numbers Dataset: <http://ufldl.stanford.edu/housenumbers/>
- <sup>54</sup>IUPR dataset: <http://www.sites.google.com/a/iupr.com/bukhari/>
- <sup>55</sup>NEOCR dataset: <http://www6.informatik.uni-erlangen.de/research/projects/pixtract/neocr/>
- <sup>56</sup>Chars74K dataset: <http://www.ee.surrey.ac.uk/CVSSP/demos/chars74k/>

### **Links to datasets for recognition of mathematical expressions**

- <sup>57</sup>Infty: <http://www.inftyproject.org/en/database.html>  
<sup>58</sup>MathBrush: <https://www.scg.uwaterloo.ca/mathbrush/corpus>  
<sup>59</sup>CROHME: <http://www.isical.ac.in/crohme2011/index.html>  
<sup>60</sup>UW-III dataset: available on CD-ROM  
<sup>61</sup>Hamex: <http://www.projet-depart.org/>  
<sup>62</sup>Marmot: [http://www.founderrd.com/marmot\\_data.htm](http://www.founderrd.com/marmot_data.htm)

### **Links to datasets for signature verification**

- <sup>63</sup>GPDS-960: <http://www.gpds.ulpgc.es/download/index.htm>  
<sup>64</sup>MCYT: <http://atvs.ii.uam.es/databases.jsp>  
<sup>65</sup>BioSecurId: <http://atvs.ii.uam.es/databases.jsp>  
<sup>66</sup>SVC2004:  
<sup>67</sup>SigComp09: <http://sigcomp09.arsforensica.org/>  
<sup>68</sup>4NSigComp10: <http://www.isical.ac.in/icfhr2010/CallforParticipation4NSigComp2010.html>  
<sup>69</sup>SigComp11: <http://forensic.to/webhome/afha/SigComp.html>

### **Links to datasets for table processing**

- <sup>70</sup><http://www.dfki.uni-kl.de/shahab/t-truth/>  
<sup>71</sup><http://www.liaad.up.pt/acs/Competition.htm>

---

## **References**

1. Alamri H, Sadri J, Suen CY, Nobile N (2008) A novel comprehensive database for Arabic off-line handwriting recognition. In: Proceedings of the 11th international conference on frontiers in handwriting recognition (ICFHR 2008), Montréal, pp 664–669
2. Al-Ohali Y, Cheriet M, Suen C (2003) Databases for recognition of handwritten arabic cheques. *Pattern Recognit* 36(1):111–121. doi:10.1016/S0031-3203(02)00064-X, URL: <http://www.sciencedirect.com/science/article/pii/S003132030200064X>
3. Antonacopoulos A, Karatzas D, Bridson D (2006) Ground truth for layout analysis performance evaluation. In: Proceedings of the 7th IAPR workshop on document analysis systems (DAS2006), Nelson. Springer, pp 302–311
4. Antonacopoulos A, Bridson D, Papadopoulos C, Pletschacher S (2009) A realistic dataset for performance evaluation of document layout analysis. In: 10th international conference on document analysis and recognition (ICDAR’09), Barcelona, 2009, pp 296–300. doi:10.1109/ICDAR.2009.271
5. Antonacopoulos A, Clausner C, Papadopoulos C, Pletschacher S (2011) Historical document layout analysis competition. In: 11th international conference on document analysis and recognition (ICDAR’11), Beijing, 2011
6. Baird HS (1995) Document image defect models. In: O’Gorman L, Kasturi R (eds) Document image analysis. IEEE Computer Society, Los Alamitos, pp 315–325. URL: <http://dl.acm.org/citation.cfm?id=201573.201660>
7. Bhattacharya U, Chaudhuri B (2009) Handwritten numeral databases of Indian scripts and multistage recognition of mixed numerals. *IEEE Trans Pattern Anal Mach Intell* 31(3): 444–457. doi:10.1109/TPAMI.2008.88
8. Blakers V, Heuvel C, Franke K, Vuurpijl L (2009) ICDAR 2009 signature verification competition. In: 10th international conference on document analysis and recognition (ICDAR’09), Barcelona, 2009, pp 1403–1407. doi:10.1109/ICDAR.2009.216

9. Bukhari SS, Shafait F, Breuel TM (2012) The IUPR dataset of camera-captured document images. In: Proceedings of the 4th international conference on camera-based document analysis and recognition (CBDAR'11), Beijing. Springer, Berlin/Heidelberg, pp 164–171
10. Dalitz C, Droettboom M, Pranzas B, Fujinaga I (2008) A comparative study of staff removal algorithms. *IEEE Trans Pattern Anal Mach Intell* 30:753–766. doi:<http://doi.ieeecomputersociety.org/10.1109/TPAMI.2007.70749>
11. Delalandre M, Valveny E, Pridmore T, Karatzas D (2010) Generation of synthetic documents for performance evaluation of symbol recognition & spotting systems. *Int J Doc Anal Recognit* 13:187–207. doi:<http://dx.doi.org/10.1007/s10032-010-0120-x>, URL: <http://dx.doi.org/10.1007/s10032-010-0120-x>
12. Doucet A, Kazai G, Dresovic B, Uzelac A, Radakovic B, Todic N (2011) Setting up a competition framework for the evaluation of structure extraction from OCR-ed books. *Int J Doc Anal Recognit* 14:45–52. doi:<http://dx.doi.org/10.1007/s10032-010-0127-3>, URL: <http://dx.doi.org/10.1007/s10032-010-0127-3>
13. El Abed H, Kherallah M, Märgner V, Alimi AM (2011) On-line Arabic handwriting recognition competition: ADAB database and participating systems. *Int J Doc Anal Recognit* 14: 15–23. doi:<http://dx.doi.org/10.1007/s10032-010-0124-6>, URL: <http://dx.doi.org/10.1007/s10032-010-0124-6>
14. Fierrez J, Galbally J, Ortega-Garcia J, Freire M, Alonso-Fernandez F, Ramos D, Toledoano D, Gonzalez-Rodriguez J, Siguenza J, Garrido-Salas J, Anguiano E, Gonzalez-de Rivera G, Ribalda R, Faundez-Zanuy M, Ortega J, Cardeñoso-Payo V, Viloria A, Vivaracho C, Moro Q, Igarza J, Sanchez J, Hernaez I, Orrite-Uruñuela C, Martinez-Contreras F, Gracia-Roche J (2010) BiosecurID: a multimodal biometric database. *Pattern Anal Appl* 13:235–246. doi:[10.1007/s10044-009-0151-4](http://10.1007/s10044-009-0151-4), URL: <http://dx.doi.org/10.1007/s10044-009-0151-4>
15. Fischer A, Indermühle E, Bunke H, Viehhäuser G, Stoltz M (2010) Ground truth creation for handwriting recognition in historical documents. In: Proceedings of the 9th IAPR international workshop on document analysis systems (DAS'10), Boston. ACM, New York, pp 3–10. doi:<http://doi.acm.org/10.1145/1815330.1815331>, URL: <http://doi.acm.org/10.1145/1815330.1815331>
16. Fornés A, Dutta A, Gordo A, Lladós J (2012) CVC-MUSCIMA: a ground truth of handwritten music score images for writer identification and staff removal. *Int J Doc Anal Recognit* 15(3), 243–251. doi:<10.1007/s10032-011-0168-2>, URL: <http://dx.doi.org/10.1007/s10032-011-0168-2>
17. Fruchterman T (1995) DAFS: a standard for document and image understanding. In: Proceedings of the symposium on document image understanding technology, Bowes, pp 94–100
18. Garain U, Chaudhuri B (2005) A corpus for OCR research on mathematical expressions. *Int J Doc Anal Recognit* 7:241–259. doi:<10.1007/s10032-004-0140-5>, URL: <http://dl.acm.org/citation.cfm?id=1102243.1102246>
19. Gatos B, Ntirogiannis K, Pratikakis I (2009) ICDAR2009 document image binarization contest (DIBCO 2009). In: 10th international conference on document analysis and recognition (ICDAR'09), Barcelona, 2009, pp 1375–1382. doi:<10.1109/ICDAR.2009.246>
20. Gatos B, Stamatopoulos N, Louloudis G (2011) ICDAR2009 handwriting segmentation contest. *Int J Doc Anal Recognit* 14:25–33. doi:<10.1007/s10032-010-0122-8>, URL: <http://dx.doi.org/10.1007/s10032-010-0122-8>
21. Guyon I, Schomaker L, Plamondon R, Liberman M, Janet S (1994) Unipen project of on-line data exchange and recognizer benchmarks. In: Proceedings of the international conference on pattern recognition, Jerusalem, pp 29–33
22. Hassaï andne A, Al-Maadeed S, Alja'am JM, Jaoua A, Bouridane A (2011) The ICDAR2011 Arabic writer identification contest. In: International conference on document analysis and recognition (ICDAR), Beijing, 2011, pp 1470–1474. doi:<10.1109/ICDAR.2011.292>
23. Helmers M, Bunke H (2003) Generation and use of synthetic training data in cursive handwriting recognition. In: Perales F, Campilho A, de la Blanca N, Sanfeliu A (eds) *Pattern recognition and image analysis. Lecture notes in computer science*, vol 2652. Springer, Berlin/Heidelberg, pp 336–345

24. Hu J, Kashi RS, Lopresti DP, Wilfong GT (2002) Evaluating the performance of table processing algorithms. *Int J Doc Anal Recognit* 4(3):140–153
25. Indermühle E, Liwicki M, Bunke H (2010) IAMonDo-database: an online handwritten document database with non-uniform contents. In: Proceedings of the 9th IAPR international workshop on document analysis systems (DAS'10), Boston. ACM, New York, pp 97–104. doi:<http://doi.acm.org/10.1145/1815330.1815343>, URL: <http://doi.acm.org/10.1145/1815330.1815343>
26. Kanai J, Rice SV, Nartker TA, Nagy G (1995) Automated evaluation of OCR zoning. *IEEE Trans Pattern Anal Mach Intell* 17:86–90. doi:<http://doi.ieeecomputersociety.org/10.1109/34.368146>
27. Kanungo T, Haralick RM, Stuezle W, Baird HS, Madigan D (2000) A statistical, nonparametric methodology for document degradation model validation. *IEEE Trans Pattern Anal Mach Intell* 22:1209–1223. doi:<http://dx.doi.org/10.1109/34.888707>, URL: <http://dx.doi.org/10.1109/34.888707>
28. Khosravi H, Kabir E (2007) Introducing a very large dataset of handwritten Farsi digits and a study on their varieties. *Pattern Recognit Lett* 28:1133–1141. doi:[10.1016/j.patrec.2006.12.022](http://dl.acm.org/citation.cfm?id=1243503.1243603), URL: <http://dl.acm.org/citation.cfm?id=1243503.1243603>
29. Kim DW, Kanungo T (2002) Attributed point matching for automatic groundtruth generation. *Int J Doc Anal Recognit* 5:47–66. doi:[10.1007/s10032-002-0083-7](http://dx.doi.org/10.1007/s10032-002-0083-7), URL: <http://dx.doi.org/10.1007/s10032-002-0083-7>
30. Lee CH, Kanungo T (2003) The architecture of TRUEVIZ: a groundtruth/metadata editing and visualizing toolkit. *Pattern Recognit* 36(3):811–825. doi:[10.1016/S0031-3203\(02\)00101-2](http://www.sciencedirect.com/science/article/pii/S0031320302001012), URL: <http://www.sciencedirect.com/science/article/pii/S0031320302001012>
31. Liang J, Phillips IT, Haralick RM (1997) Performance evaluation of document layout analysis algorithms on the UW data set. In: Proceedings of the SPIE document recognition IV, San Jose, pp 149–160
32. Liwicki M, Bunke H (2005) IAM-OnDB – an on-line English sentence database acquired from handwritten text on a whiteboard. In: Proceedings of the eighth international conference on document analysis and recognition (ICDAR'05), Seoul. IEEE Computer Society, Washington, DC, pp 956–961. doi:<http://dx.doi.org/10.1109/ICDAR.2005.132>, URL: <http://dx.doi.org/10.1109/ICDAR.2005.132>
33. Liwicki M, van den Heuvel C, Found B, Malik M (2010) Forensic signature verification competition 4NSigComp2010 – detection of simulated and disguised signatures. In: International conference on frontiers in handwriting recognition (ICFHR), Kolkata, 2010, pp 715–720. doi:[10.1109/ICFHR.2010.116](http://dx.doi.org/10.1109/ICFHR.2010.116)
34. Liwicki M, Malik M, van den Heuvel C, Chen X, Berger C, Stoel R, Blumenstein M, Found B (2011) Signature verification competition for online and offline skilled forgeries (SigComp2011). In: International conference on document analysis and recognition (ICDAR), Beijing, 2011, pp 1480–1484. doi:[10.1109/ICDAR.2011.294](http://dx.doi.org/10.1109/ICDAR.2011.294)
35. Lopresti D (2009) Optical character recognition errors and their effects on natural language processing. *Int J Doc Anal Recognit* 12:141–151. doi:[10.1007/s10032-009-0094-8](http://dx.doi.org/10.1007/s10032-009-0094-8), URL: <http://dx.doi.org/10.1007/s10032-009-0094-8>
36. Louloudis G, Stamatopoulos N, Gatos B (2011) ICDAR 2011 writer identification contest. In: International conference on document analysis and recognition (ICDAR), Beijing, 2011, pp 1475–1479. doi:[10.1109/ICDAR.2011.293](http://dx.doi.org/10.1109/ICDAR.2011.293)
37. Lucas SM, Panaretos A, Sosa L, Tang A, Wong S, Young R (2003) ICDAR 2003 robust reading competitions. In: Proceedings of the seventh international conference on document analysis and recognition (ICDAR'03), Edinburgh, vol 2. IEEE Computer Society, Washington, DC, pp 682–687. URL: <http://dl.acm.org/citation.cfm?id=938980.939531>
38. MacLean S, Labahn G, Lank E, Marzouk M, Tausky D (2011) Grammar-based techniques for creating ground-truthed sketch corpora. *Int J Doc Anal Recognit* 14: 65–74. doi:<http://dx.doi.org/10.1007/s10032-010-0118-4>, URL: <http://dx.doi.org/10.1007/s10032-010-0118-4>

39. Marti UV, Bunke H (1999) A full English sentence database for off-line handwriting recognition. In: Proceedings of the fifth international conference on document analysis and recognition (ICDAR'99), Bangalore. IEEE Computer Society, Washington, DC, pp 705–708. URL: <http://dl.acm.org/citation.cfm?id=839279.840504>
40. Mihov S, Schulz K, Ringlstetter C, Dojchinova V, Nakova V, Kalpakchieva K, Gerasimov O, Gotscharek A, Gercke C (2005) A corpus for comparative evaluation of OCR software and postcorrection techniques. In: Proceedings of the eighth international conference on document analysis and recognition, Seoul, 2005, vol 1, pp 162–166. doi:10.1109/ICDAR.2005.6
41. Moll M, Baird H, An C (2008) Truthing for pixel-accurate segmentation. In: The eighth IAPR international workshop on document analysis systems (DAS'08), Japan, 2008, pp 379–385. doi:10.1109/DAS.2008.47
42. Mori M, Suzuki A, Shio A, Ohtsuka S (2000) Generating new samples from handwritten numerals based on point correspondence. In: Proceedings of the 7th international workshop on frontiers in handwriting recognition (IWFHR2000), Amsterdam, pp 281–290
43. Mouchere H, Viard-Gaudin C, Kim DH, Kim JH, Garain U (2011) CROHME2011: competition on recognition of online handwritten mathematical expressions. In: International conference on document analysis and recognition (ICDAR), Beijing, 2011, pp 1497–1500. doi:10.1109/ICDAR.2011.297
44. Ntiogiannis K, Gatos B, Pratikakis I (2008) An objective evaluation methodology for document image binarization techniques. In: The eighth IAPR international workshop on document analysis systems (DAS'08), Nara, 2008, pp 217–224. doi:10.1109/DAS.2008.41
45. Okamoto M, Imai H, Takagi K (2001) Performance evaluation of a robust method for mathematical expression recognition. In: International conference on document analysis and recognition, Seattle, p 0121. doi:<http://doi.ieeecomputersociety.org/10.1109/ICDAR.2001.953767>
46. Ortega-Garcia J, Fierrez-Aguilar J, Simon D, Gonzalez J, Faundez-Zanuy M, Espinosa V, Satue A, Hernaez I, Igarza JJ, Vivaracho C, Escudero D, Moro QI (2003) MCYT baseline corpus: a bimodal biometric database. IEE Proc Vis Image Signal Process 150(6):395–401. doi:10.1049/ip-vis:20031078
47. Paredes R, Kavallieratou E, Lins RD (2010) ICFHR 2010 contest: quantitative evaluation of binarization algorithms. In: International conference on frontiers in handwriting recognition, Kolkata, pp 733–736. doi:<http://doi.ieeecomputersociety.org/10.1109/ICFHR.2010.119>
48. Perez D, Tarazon L, Serrano N, Castro F, Terrades O, Juan A (2009) The GERMANA database. In: 10th international conference on document analysis and recognition (ICDAR'09), Barcelona, 2009, pp 301–305. doi:10.1109/ICDAR.2009.10
49. Phillips IT, Chhabra AK (1999) Empirical performance evaluation of graphics recognition systems. IEEE Trans Pattern Anal Mach Intell 21:849–870. doi:<http://dx.doi.org/10.1109/34.790427>, URL: <http://dx.doi.org/10.1109/34.790427>
50. Phillips I, Chen S, Haralick R (1993) CD-ROM document database standard. In: Proceedings of the second international conference on document analysis and recognition, Tsukuba, 1993, pp 478–483. doi:10.1109/ICDAR.1993.395691
51. Phillips I, Ha J, Haralick R, Dori D (1993) The implementation methodology for a CD-ROM English document database. In: Proceedings of the second international conference on document analysis and recognition, Tsukuba, 1993, pp 484–487. doi:10.1109/ICDAR.1993.395690
52. Plamondon R, Guerfali W (1998) The generation of handwriting with delta-lognormal synergies. Biol Cybern 132:119–132
53. Pletschacher S, Antonacopoulos A (2010) The page (page analysis and ground-truth elements) format framework. In: 20th international conference on pattern recognition (ICPR), Istanbul, 2010, pp 257–260. doi:10.1109/ICPR.2010.72
54. Pratikakis I, Gatos B, Ntiogiannis K (2010) H-DIBCO 2010 – handwritten document image binarization competition. In: International conference on frontiers in handwriting recognition (ICFHR), Kolkata, 2010, pp 727–732. doi:10.1109/ICFHR.2010.118
55. Pratikakis I, Gatos B, Ntiogiannis K (2011) ICDAR 2011 document image binarization contest (DIBCO 2011). In: International conference on document analysis and recognition (ICDAR), Beijing, 2011, pp 1506–1510. doi:10.1109/ICDAR.2011.299

56. Quiniou S, Mouchere H, Saldarriaga S, Viard-Gaudin C, Morin E, Petitrenaud S, Medjkoune S (2011) HAMEX – a handwritten and audio dataset of mathematical expressions. In: International conference on document analysis and recognition (ICDAR), Beijing, 2011, pp 452–456. doi:10.1109/ICDAR.2011.97
57. Rath TM, Mammatha R (2007) Word spotting for historical documents. *Int J Doc Anal Recognit* 9(2):139–152. doi:10.1007/s10032-006-0027-8, URL: <http://dx.doi.org/10.1007/s10032-006-0027-8>
58. Rice SV, Jenkins FR, Nartker TA (1996) The fifth annual test of OCR accuracy. Technical report TR-96-01. AIInformation Science Research Institute (University of Nevada, Las Vegas)
59. Rusiñol M, Borris A, Lladós J (2010) Relational indexing of vectorial primitives for symbol spotting in line-drawing images. *Pattern Recognit Lett* 31:188–201. doi:<http://dx.doi.org/10.1016/j.patrec.2009.10.002>, URL: <http://dx.doi.org/10.1016/j.patrec.2009.10.002>
60. Saund E, Lin J, Sarkar P (2009) PixLabeler: user interface for pixel-level labeling of elements in document images. In: Proceedings of the 2009 10th international conference on document analysis and recognition (ICDAR'09), Barcelona. IEEE Computer Society, Washington, DC, pp 646–650. doi:<http://dx.doi.org/10.1109/ICDAR.2009.250>, URL: <http://dx.doi.org/10.1109/ICDAR.2009.250>
61. Schomaker L, Thomassen A, Teulings HL (1989) A computational model of cursive handwriting. In: Plamondon R, Suen CY, Simmer ML (eds) Computer recognition and human production of handwriting. World Scientific, Singapore/Teaneck, pp 153–177
62. Serrano N, Castro F, Juan A (2010) The RODRIGO database. In: LREC, Valletta
63. Setlur S, Govindaraju V (1994) Generating manifold samples from a handwritten word. *Pattern Recognit Lett* 15(9):901–905. doi:10.1016/0167-8655(94)90152-X, URL: <http://www.sciencedirect.com/science/article/pii/016786559490152X>
64. Shafait F (2007) Document image dewarping contest. In: 2nd international workshop on camera-based document analysis and recognition, Curitiba, pp 181–188
65. Shahab A, Shafait F, Kieninger T, Dengel A (2010) An open approach towards the benchmarking of table structure recognition systems. In: Proceedings of the 9th IAPR international workshop on document analysis systems (DAS'10), Boston. ACM, New York, pp 113–120. doi:<http://doi.acm.org/10.1145/1815330.1815345>, URL: <http://doi.acm.org/10.1145/1815330.1815345>
66. Smith EHB (2010) An analysis of binarization ground truthing. In: Proceedings of the 9th IAPR international workshop on document analysis systems (DAS'10), Boston. ACM, New York, pp 27–34. doi:<http://doi.acm.org/10.1145/1815330.1815334>, URL: <http://doi.acm.org/10.1145/1815330.1815334>
67. Solimanpour F, Sadri J, Suen CY (2006) Standard databases for recognition of handwritten digits, numerical strings, legal amounts, letters and dates in Farsi language. In: Lorette G (ed) Tenth international workshop on frontiers in handwriting recognition, Université de Rennes 1, Svisoft, La Baule. URL: <http://hal.inria.fr/inria-00103983/en/>
68. Suen C, Nadal C, Legault R, Mai T, Lam L (1992) Computer recognition of unconstrained handwritten numerals. *Proc IEEE* 80(7):1162–1180. doi:10.1109/5.156477
69. Todoran L, Worring M, Smeulders M (2005) The UvA color document dataset. *Int J Doc Anal Recognit* 7:228–240. doi:10.1007/s10032-004-0135-2, URL: <http://dl.acm.org/citation.cfm?id=1102243.1102245>
70. Uchida S, Nomura A, Suzuki M (2005) Quantitative analysis of mathematical documents. *Int J Doc Anal Recognit* 7:211–218. doi:10.1007/s10032-005-0142-y, URL: <http://dl.acm.org/citation.cfm?id=1102243.1102248>
71. Varga T, Bunke H (2003) Generation of synthetic training data for an HMM-based handwriting recognition system. In: Proceedings of the seventh international conference on document analysis and recognition (ICDAR'03), Edinburgh, vol 1. IEEE Computer Society, Washington, DC, pp 618–622. URL: <http://dl.acm.org/citation.cfm?id=938979.939265>
72. Viard-Gaudin C, Lallican PM, Binter P, Knerr S (1999) The IRESTE On/Off (IRONOFF) dual handwriting database. In: Proceedings of the fifth international conference on document

- analysis and recognition (ICDAR'99), Bangalore. IEEE Computer Society, Washington, DC, pp 455–458. URL: <http://dl.acm.org/citation.cfm?id=839279.840372>
73. Wang K, Belongie S (2010) Word spotting in the wild. In: Proceedings of the 11th European conference on computer vision: part I (ECCV'10), Heraklion. Springer, Berlin/Heidelberg, pp 591–604. URL: <http://dl.acm.org/citation.cfm?id=1886063.1886108>
74. Wang J, Wu C, Xu YQ, Shum HY, Ji L (2002) Learning-based cursive handwriting synthesis. In: Proceedings of the eighth international workshop on frontiers of handwriting recognition, Niagara-on-the-Lake, pp 157–162
75. Wang DH, Liu CL, Yu JL, Zhou XD (2009) CASIA-OLHWDB1: a database of online handwritten Chinese characters. In: Proceedings of the 2009 10th international conference on document analysis and recognition (ICDAR'09), Barcelona. IEEE Computer Society, Washington, DC, pp 1206–1210. doi:<http://dx.doi.org/10.1109/ICDAR.2009.163>, URL: <http://dx.doi.org/10.1109/ICDAR.2009.163>
76. Yang L, Huang W, Tan CL (2006) Semi-automatic ground truth generation for chart image recognition. In: Workshop on document analysis systems (DAS), Nelson, pp 324–335
77. Yanikoglu BA, Vincent L (1998) Pink panther: a complete environment for ground-truthing and benchmarking document page segmentation. Pattern Recognit 31(9): 1191–1204. doi:10.1016/S0031-3203(97)00137-4, URL: <http://www.sciencedirect.com/science/article/pii/S0031320397001374>
78. Zhai J, Wenjin L, Dori D, Li Q (2003) A line drawings degradation model for performance characterization. In: Proceedings of the seventh international conference on document analysis and recognition, Edinburgh, 2003, pp 1020–1024. doi:10.1109/ICDAR.2003.1227813

## Further Reading

Datasets are only one of the pillars of any performance evaluation system. The other one is everything related to evaluation, mainly metrics and protocols of evaluation. All these are deeply discussed in ►Chap. 30 (Tools and Metrics for Document Analysis Systems Evaluation) of this book, and the interested reader can find there a detailed discussion about the different alternatives and approaches used in document analysis.

The topic covered in this chapter is very broad and presents many specific particularities for each subfield of document analysis. Thus, we have only been able to include a very generic description of the datasets. For a further description of the datasets, the reader can go either to the chapter of this book devoted to a specific problem (where some of these datasets are further described) either to the specific references cited throughout the chapter and the appendix. In addition, many of these datasets are being used in the international conferences related to document analysis (mainly, ICDAR, DAS, and ICFHR). The proceedings of these conferences can be another good source of information about the datasets and the current state-of-the-art results for each of them.

---

# Tools and Metrics for Document Analysis Systems Evaluation

# 30

Volker Märgner and Haikal El Abed

## Contents

Introduction.....	1012
Document Processing Evaluation.....	1013
Evaluation of Systems and Modules.....	1015
Evaluation Metrics.....	1016
Preprocessing Step: Binarization.....	1018
Page/Document Layout Analysis.....	1021
Text Segmentation.....	1024
Character/Text Recognition.....	1025
Evaluation Tools.....	1028
Ground-Truthing Tools.....	1028
Ground Truth in Document Analysis.....	1029
Architecture of an Evaluation Framework.....	1030
Contests and Benchmarking.....	1032
Conclusion.....	1034
References.....	1034
Further Reading.....	1036

---

## Abstract

Any development within a specific research field like document analysis and recognition comes along with the need for data and corresponding measurement devices and test equipment. This chapter introduces the basic issues of evaluation methods for different kind of document analysis systems and modules with a special emphasis on tools and metrics available and used today.

This chapter is organized as follows: After a general introduction including general definitions of terms used in document analysis system evaluation and general overviews of evaluation processes in section “[Introduction](#),” different

---

V. Märgner (✉) • H.E. Abed

Institute for Communications Technology, Technische Universität Braunschweig, Braunschweig, Germany

e-mail: [maergner@ifn.ing.tu-bs.de](mailto:maergner@ifn.ing.tu-bs.de); [el-abed@ifn.ing.tu-bs.de](mailto:el-abed@ifn.ing.tu-bs.de)

evaluation metrics are discussed in section “[Evaluation Metrics](#).” These metrics cover the different aspects of the Document Analysis Handbook as presented in ►Chaps. 2 (Document Creation, Image Acquisition and Document Quality)–►8 (Text Segmentation for Document Recognition), from image-processing evaluation metrics to special metrics for selected applications e.g., character/text recognition. In section “[Evaluation Tools](#),” an overview of ground-truth file structure and a selection of available ground-truth tools are presented. Performance evaluation tools and competitions organized within the last years are also listed in section “[Evaluation Tools](#).”

---

#### Keywords

Benchmark • Competition • Evaluation • Evaluation tools • Ground truth • Metrics • Quality measure

---

## Introduction

Recognition tasks usually transform signals – one-, two- or more dimensional – into symbols of a known alphabet or vocabulary/lexicon. From the very beginning the most important measurement of the quality of such systems is based on a comparison of the ground truth and the system output symbol string. At first for such tasks, metrics and tools have to be developed. Very soon it became clear that obviously the comparison with the output only of such complex systems represents a general approach and does not take into account different aspects which may play an important role in the transformation process from an image to a symbol string. Especially systems for dedicated applications or single modules development of document analysis systems necessitate a more decided quality measurement. As a consequence different metrics and tools were developed on color, grey, or binary value of the image and on the object shape.

Before a short introduction to system evaluation starts, the definitions of the terms used in system development and evaluation are given:

**Assessment:** “**Assessment** is the collection of relevant information that may be relied on for making decisions. **Evaluation** is the application of a standard and a decision-making system to assessment data to produce judgments about the amount and adequacy of the learning that has taken place.” Too often these processes are confused ... it is said assess, but meant evaluate ... or the term evaluation is used, when really assessment [11] is done.

**Evaluation:** As defined by the American Evaluation Association (<http://www.eval.org/>), evaluation involves assessing the strengths and weaknesses of programs, policies, personnel, products, and organizations to improve their effectiveness.

**Benchmarking:** For the term benchmarking the Oxford Dictionaries gives the following definition: Benchmarking means a standard or point of reference against which things may be compared. This is for example a problem designed to evaluate the performance of a computer system.

**Ground truth:** Ground truth is a term first used in cartography, meteorology, analysis of aerial photographs, satellite imagery, and a range of other remote sensing techniques in which data are gathered at a distance. Ground truth refers to information that is collected “on location.” In document analysis and recognition techniques, the term “ground truth” refers to the correct meaning or classification of a data set, used for the training and testing for supervised learning techniques.

**Comparison:** Means the act of comparing. It is an examination of two or more objects with the view of discovering the resemblances or differences. (<http://wwwdefinitions.net/definition/comparison>)

**Contest/Competition:** These terms mean the act of seeking, or endeavoring to gain, what another is endeavoring to gain at the same time. (<http://wwwdefinitions.net/definition/competition>)

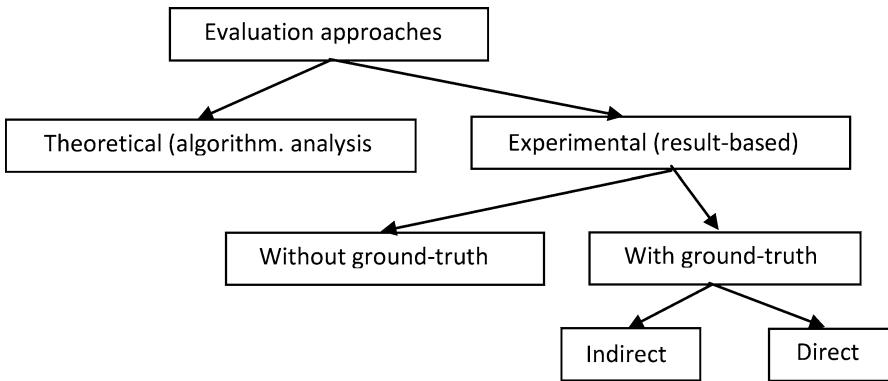
## Document Processing Evaluation

Any document analysis and recognition system need quality evaluation or benchmarking as the increasing quality and diversity of document analysis systems have made it more and more difficult to compare systems. Moreover, as the complexity of the systems has increased with the effect that modifications within one module – even if just one parameter is modified – may often lead to an unpredictable behavior towards other modules. The increasing amount of ready-to-use algorithms and the exploitation of new application fields for document analysis are making it rather difficult to find a suitable configuration. In conclusion, both, a detailed qualitative and quantitative failure analysis are needed for further improvement.

When making an evaluation, it can be distinguished between two different objectives:

- Benchmarking for the system user: In this case, only the final results (e.g., the ASCII text output of an OCR system (Optical Character Recognition) or a set of categories output of a document categorization system) are of interest. There is no motivation to look in more detail at internal modules.
- Benchmarking for the system developer: In this case a detailed failure analysis is necessary. This requires focusing not only on final system output but also on intermediate results, i.e., on the output of single system modules. That means a module’s output data must be accessible for an evaluation, e.g., comparison with corresponding ground-truth data.

Evaluation approaches itself may be classified with the scheme shown in Fig. 30.1. It can be distinguished between theoretical and experimental approaches. Using the theoretical approaches, the algorithm implemented will be analyzed in order to derive its behavior. The input data are assumed to be composed of ideal data degraded by noise. Input data degradation is propagated by analytical description throughout the algorithm. Approaches like these theoretical ones are mainly used in the domain of low-level computer vision algorithms. Such approaches would hardly be applicable for segmentation modules as part of a document analysis system as



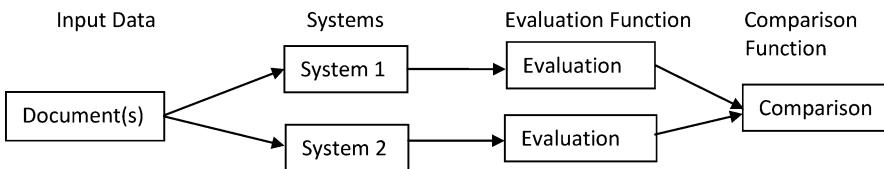
**Fig. 30.1** Classification of evaluation approaches

the implemented algorithms are quite more complex to that of image preprocessing modules.

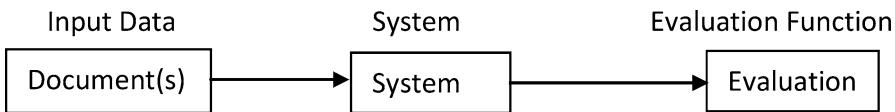
On the contrary, experimental evaluation without ground truth considers a system or a module characterized as a black box. The evaluation is based on the system or module output only. Approaches like this are much more pragmatic and especially they are independent of the underlying algorithm. Using evaluation methods without ground truth, it is possible to implement online measures which can be used to evaluate a specific aspect of quality. In this context online means that the system itself can calculate the measure of assessing the quality of its results, ground truth is not available. But when making a detailed and comprehensive quality evaluation, this methodology cannot be sufficient.

When using ground truth, quality evaluation means to make a comparison of the output with the ground truth. It depends on the type of data whether the comparison is simple or more complicated. It can easily be seen that the comparison of segmentation results is more complicated than, e.g., the comparison of character classification results because segmentation results are neither isolated like character patterns nor of a symbolical kind.

Not least because of this, there exists the idea that comparison may not be performed on the basis of the direct results but on the basis of indirect results. That means a further processing of the results has to be done. For example, the textual results which a document analysis system generates may be compared with the plain ground-truth text even to evaluate the zoning quality of a system. The comparison may be done by means of a string-matching algorithm. A great advantage of this approach is that ground truth is only needed on a textual level and not on the segment level. But the disadvantage is that an exact quality evaluation cannot be performed, due to the loss of information when the segments are converted to a stream of character symbols. This disadvantage does not exist when an evaluation is made on the basis of direct results, say, of the segment data itself.



**Fig. 30.2** Comparison of two systems



**Fig. 30.3** Evaluation of one system

Besides the question how to perform the evaluation, there is the task to provide test data. There are two extremes: the collection of real data and, since this is often a very laborious task, the generation of synthetic data.

## Evaluation of Systems and Modules

Evaluation basically pursues three aims:

- Improvement and quality control of a system
- Comparison of two systems
- Definition of the absolute performance of a system

As the first aim tends to increase or maintain a system's performance, the second compares systems to get the information which one is the better with respect to a certain task. The third aim is the definition of the absolute performance of a system. Figures 30.2 and 30.3 show block diagrams of different types of document analysis systems evaluation.

Now let us have a closer look to the blocks of the evaluation system: input data, system, evaluation function, and comparison function. The input data, in our case scanned documents, need to be representative for the population of the system's task. The system itself is a description of the vaguely defined task but not necessarily of a completely defined algorithm. This is important to have in mind when an ideal system and not a real system is in consideration. That means a system may be described through the desired output only. This output is on the same symbolic level as the ground truth. The evaluation function assigns a value to the output, e.g., based on a comparison of the output symbol sequence with the corresponding ground truth. The evaluation function is less decisive for OCR or computer systems, for instance, but a crucial factor in areas like image processing or segmentation. Finally the comparison function compares the evaluation values of different systems. Evaluation methods may be defined by a human expert, the subsequent system, or a comparison with ground truth. The evaluation and the

comparison functions have to use metrics useful to measure most important system core module features. In the following evaluation metrics, at first general and later specific document-processing tasks are presented.

---

## Evaluation Metrics

Image quality in general plays an important role in document analysis and recognition, as all tasks applied to the document image need a certain image quality to be successful. But the definition of image quality metrics is difficult as it usually needs a ground-truth image as a basis to calculate a quality measure. In addition to the often used peak signal-to-noise ratio (PSNR) metric as a simple statistical measure using pixel differences, especially for image coding human visual perception is integrated into the metric, for example, the knowledge that in an image noise close to a high contrast edge has a different influence to the image quality than on a part of the image with constant color. For document analysis systems the quality metric has to be aligned to the typical content and also to the typical noise in scanned documents. Performing a system evaluation needs a measure for quantifying the performance of the whole system or module or even parts of them. A metric is used as such an objective measure.

Image quality metrics are widely used for image coding quality measure. In case of image coding both the original and the coded image are available, the quality loss introduced by the coding can be measured by comparing both images. The image preprocessing task is more difficult as the original image which is the goal of any preprocessing task is usually not known.

A metric to measure the distance between original and distorted image is the peak signal-to-noise ratio (PSNR). PSNR is the ratio between the maximum possible power of a signal, in our case the signal is an image, and the power of the noise affecting the quality of the signal (image). For image processing this metric is primarily used to measure the quality of a lossy image compression codec, where the compression introduces some noise into the original image. PSNR is defined by using the mean square error (MSE). Given an image of size  $N_x \times N_y$ , the noise-free original image is given by  $I_1(x, y)$  and the noisy version by  $I_2(x, y)$ .

The MSE is defined as

$$MSE = \frac{\sum_{x=1}^{N_x} \sum_{y=1}^{N_y} (I_1(x, y) - I_2(x, y))^2}{N_x \times N_y} \quad (30.1)$$

And the corresponding PSNR as

$$PSNR = 10 \cdot \log_{10} \left( \frac{\text{Max}_I^2}{MSE} \right) \quad (30.2)$$

$\text{Max}_1$  is the maximum possible pixel value appearing in the image. In case of an 8 bit per pixel grey value image, this value is 255. For color images with RGB presentation of the colors, the definition is the same except the MSE is the sum of all 3 (R, G, and B) channels squared differences divided by image size and by 3. To use this measure for image preprocessing, e.g., for noise reduction, a noise free version of the image is needed as reference. The measure of PSNR uses this image and the preprocessed one. The best possible image preprocessing finally results in an image equal to the noise-free version of the image with  $\text{MSE} = 0$  and an infinite PSNR. Typical preprocessing steps result in an  $\text{MSE} > 0$  and  $\text{PSNR} < \infty$ .

Metrics used in document analysis systems evaluation often originate from the field of information retrieval or information extraction. This is understandable as the main goal of document analysis can be seen as an information extraction process also, since the information content out of a document has to be extracted.

Most important and widely used metrics in the field of information extraction are recall (R) and precision (P). Information extraction – the task for the extraction of a certain word out of a text-deals with completeness of retrieval (all objects are extracted) and the purity of retrieval (only correct objects are retrieved). Recall is defined as correctly retrieved objects out of all objects in the data set, which gives a measure for the completeness of retrieval. Precision is defined as correctly retrieved objects out of all objects retrieved, which gives a measure for the purity of retrieved data.

Recall and precision consider two different aspects of retrieval systems. One may be of more interest in the precision if wrongly retrieved objects are highly undesirable; others have more problems if objects present in the data set are missed. To evaluate the overall quality of a system, a further metric is defined taking into account both recall and precision. Basic idea is to bring the two measures together and calculate the weighted mean of both. The weight is used to take different importance of recall and precision into account. As precision and recall are ratios with the same denominator but different numerator, the best mean is not the often used arithmetic mean but the harmonic mean, defined as

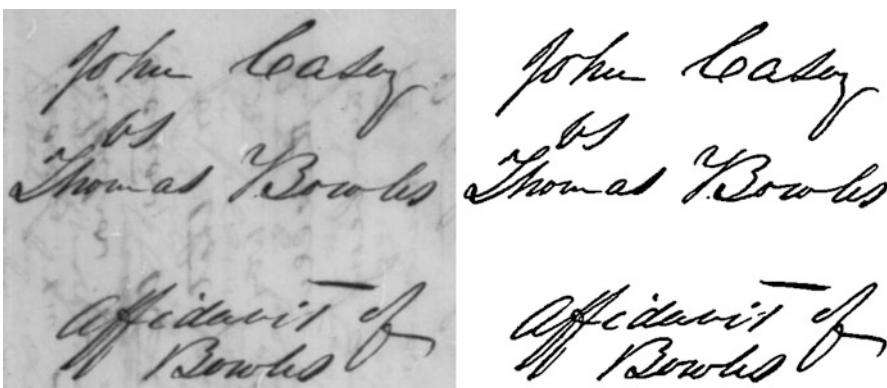
$$H = \frac{2}{\frac{1}{P} + \frac{1}{R}} = \frac{2PR}{P + R} = FM \quad (30.3)$$

This unweighted harmonic mean is called F-measure (FM) and used as a metric for different applications in data retrieval.

In the following metrics to evaluate document preprocessing, page layout analysis, text segmentation, and recognition are presented. It can be seen that the aforementioned metrics are used in many different applications in the document processing field.



**Fig. 30.4** Image of a part of a book page, *left side* color image, *right side* corresponding binary ground-truth (GT) image



**Fig. 30.5** Image of a part of a letter, *left side* grey scale, *right side* corresponding binary ground-truth (GT) image

### Preprocessing Step: Binarization

Noise reduction is a commonly used preprocessing technique with usually not a strong influence to the subsequent processing steps. Binarization is a preprocessing step in many cases essential for the next steps in the processing chain. Using a proper binarization is therefore an extremely important decision for the system performance. The comparison between binarization methods is based on several metrics described in the following. These metrics were used in document image binarization contests and first defined in [31]. The evaluation of binarization methods is basically based on comparison between the binary image and the corresponding ground-truth image (GT). In both images black pixels (equal to 1) are classified as foreground and white pixels (equal to 0) are classified as background (►Chap. 4 (Imaging Techniques in Document Analysis Processes)).

Figures 30.4 and 30.5 show a small part of a printed and a handwritten document together with the corresponding ground-truth binary image. The images are examples from the H-DIBCO 2009 Handwritten Document Image Binarization Contest

held during the International Conference on Document Analysis and Recognition in Barcelona 2009 [12].

### F-Measure (FM)

A very important metric often used for binarization evaluation is the F-measure (*FM*) (see Eq. 30.3 for definition) which is calculated pixel-wise in case of binarization. The metric *FM* is calculated according to *Recall* (Eq. 30.5) and *Precision* (Eq. 30.6) as it is shown in Eq. 30.4.

$$FM = \frac{2 \times Recall \times Precision}{Recall + Precision} \quad (30.4)$$

$$Recall = \frac{\sum_{x=1}^{N_x} \sum_{y=1}^{N_y} GT(x, y) \times B(x, y)}{\sum_{x=1}^{N_x} \sum_{y=1}^{N_y} GT(x, y)} \times 100 \quad (30.5)$$

$$Precision = \frac{\sum_{x=1}^{N_x} \sum_{y=1}^{N_y} GT(x, y) \times B(x, y)}{\sum_{x=1}^{N_x} \sum_{y=1}^{N_y} B(x, y)} \times 100 \quad (30.6)$$

In both Eqs. 30.5 and 30.6, *GT* and *B* denote the ground-truth and the binary image respectively with foreground (black) pixels coded with 1 and background (white) pixels coded with 0.

It is easy to see that in this case FM is the harmonic mean of the ratio of number of same pixels in the ground-truth binary image and the binary image under test to pixels in the ground-truth binary image in case of recall and the binary image under test in case of precision.

### Peak Signal-to-Noise Ratio (PSNR)

As second metric for the evaluation of binarization methods the aforementioned peak signal-to-noise ratio (PSNR) (see Eq. 30.2) is used. *PSNR* measures the difference of the binary image under test compared to the ground-truth binary image and is calculated as follows:

$$PSNR = 10 \cdot \log_{10} \left( \frac{C^2}{MSE} \right) \quad (30.7)$$

*C* is a value showing the maximal difference between foreground and background pixel intensities; this value is set to 1 in the case of binary images. The mean square error *MSE* is calculated by Eq. 30.8.

$$MSE = \frac{\sum_{x=1}^{Nx} \sum_{y=1}^{Ny} (GT(x, y) \times B(x, y))^2}{Nx \times Ny} \quad (30.8)$$

$GT$  and  $B$  denote the ground-truth and the binary images respectively with foreground (black) pixel coded with 1 and background (white) pixel coded with 0.

This objective measure takes into account different pixel values without consideration of the subjective visual perception of the difference.

### Negative Rate Metric (NRM)

A more specific metric is the Negative Rate Metric (NMR).  $NRM$  represents the relationship between the ground-truth (GT) and the binary image (B) pixels.  $\#TP$ ,  $\#FP$ ,  $\#FN$ , and  $\#TN$  denote the number of true positive ( $GT = 1$  and  $B = 1$ ), false positive ( $GT = 0$  and  $B = 1$ ), false negative ( $GT = 1$  and  $B = 0$ ), and true negative ( $GT = 0$  and  $B = 0$ ), respectively.  $NRM$  is calculated using Eq. 30.9.

$$NRM = \frac{\frac{\#FN}{\#FN + \#TP} + \frac{\#FP}{\#FP + \#TN}}{2} \quad (30.9)$$

$NRM$  is a value calculated as arithmetic mean of foreground pixels changed to background in relation to all foreground pixels and background pixel changed to foreground in relation to all background pixels.

### Distance Reciprocal Distortion Metric (DRD)

The Distance Reciprocal Distortion Metric (DRD) takes into account not only the pixel value itself but also the value of pixels close to the one changed its color.  $DRD$  has been used also to measure the subjective visual distortion in binary document images [22]. It properly correlates with the human visual perception, and it measures the distortion for all S pixels which color has changed from white to black or vice versa (flipped pixels) as follows:

$$DRD = \frac{\sum_{k=1}^S DRD_k}{NUBN} \quad (30.10)$$

where  $DRD_k$  is the distortion of the  $k$ th flipped pixel and is calculated using a  $5 \times 5$  normalized weight matrix  $W_{Nm}$  (Table 30.1) as defined in [22], where each neighboring pixel is weighted by a reciprocal of the distance to the flipped pixel in the center.  $DRD_k$  equals to the weighted sum of the pixels in the  $5 \times 5$  block of the ground-truth  $GT$  that differ from the centered  $k$ th flipped pixel at  $(x,y)$  in the binarization result image  $B$  (Eq. 30.11).

**Table 30.1** Normalized weight matrix  $W_{Nm}$ 

0.0256	0.0324	0.0362	0.0324	0.0256
0.0324	0.0512	0.0724	0.0512	0.0324
0.0362	0.0724	0	0.0724	0.0362
0.0324	0.0512	0.0724	0.0512	0.0324
0.0256	0.0324	0.0362	0.0324	0.0256

$$DRD_k = \sum_{i=-2}^2 \sum_{j=2}^2 |G T_k(i, j) - B_k(x, y)| \times W_{Nm}(i, j) \quad (30.11)$$

Finally,  $NUBN$ , the denominator in Eq. 30.10, is the number of the nonuniform (not all black or all white)  $8 \times 8$  blocks in the GT image.

$DRD$  is a value where each pixel with different value to the ground truth is not only counted but weighted by the number of pixels in the neighborhood with different value than the centered ground-truth pixel value. The normalization factor is the sum of all nonuniform  $8 \times 8$  pixels in the ground-truth image.

### Misclassification Penalty Metric (MPM)

Finally the Misclassification Penalty Metric (MPM), a metric which uses the mean position change of contour points between ground-truth and binary image. As defined in Eq. 30.12, MPM measures a mean distance between contour points of the ground-truth (GT) and the binary image ( $B$ ).  $d_{FN}^a$  is here defined as the distance between the  $a$ th false negative ( $GT = 1$  and  $B = 0$ ) and the closest contour pixel of the binary image.  $d_{FP}^b$  is defined as the distance between the  $b$ th false positive ( $GT = 0$  and  $B = 1$ ) and the closest contour pixel of the ground-truth image. The normalization factor  $D$  is the overall sum of the pixel-to-contour distances of the ground-truth object.

$$MPM = \frac{\sum_{a=1}^{\#FN} d_{FN}^a + \sum_{b=1}^{\#FP} d_{FP}^b}{D} \quad (30.12)$$

The MPM metric takes into account that the mean distance of contour points in GT and B images is a good quality measure as a small distance is a less important error than a large distance.

All these different metrics used for the evaluation of binarization results show us a large variety. It depends on the application which metric is the best to use as different errors result in different penalties. In the following some of the most important metrics used for page layout evaluation are presented.

## Page/Document Layout Analysis

Document layout analysis aims to the segmentation of a document page into logical areas (see ►Chaps. 5 (Page Segmentation Techniques in Document Analysis) and



**Fig. 30.6** Sample pages for layout analysis on the *left* and an example for a ground-truth layout analysis of a magazine page on the *right*. (These types of pages are used in [3] for layout analysis performance evaluation.)

►6 (Analysis of the Logical Layout of Documents)). The image of a scanned page is used first to analyze the physical structure of the page image. The physical structure is subsequently transformed into the page logical structure which is essential to read (understand) a document's content. Metrics, used to evaluate the performance of page layout analysis, have to deal with the comparison of image areas and not so much with pixel comparison and the corresponding values as it is the case for evaluation of image quality, preprocessing, and binarization as seen in section “[Preprocessing Step: Binarization](#).”

Figure 30.6 shows some examples of pages with different layouts. As can be seen the layout differs a lot. The layout depends a lot on the document type (e.g., scientific journal or boulevard magazine). The ground truth is given by assigning each physical block to a logical part of the document, e.g., text, image, and graphic, as shown in the right part of Fig. 30.6. In the following metrics to measure the performance of page layout analysis systems are presented.

The performance evaluation method presented here is that one used in ICDAR 2005 Page Segmentation Competition [2] which is based on counting the number of matches between the entities detected by the algorithm and the entities in the ground-truth image [8, 29, 30]. From these results a global *MatchScore* table for all entities is build whose values are calculated according to the intersection of the ON pixel sets of the result and the ground truth (a similar technique is used in [40]). Let  $I$  be the set of all image points,  $G_j$  the set of all points inside the ground-truth region  $j$ ,  $R_i$  the set of all points inside the result region  $i$ , and  $T(s)$  a function that counts the elements of set  $s$ . Table *MatchScore*( $i, j$ ) represents the matching results

of the ground-truth region  $j$  and the result region  $i$ . The global *MatchScore* table for all entities is defined as:

$$\text{MatchScore}(i, j) = \frac{T(G_j \cap R_i \cap I)}{T((G_j \cup R_i) \cap I)} \quad (30.13)$$

If  $N_i$  is the count of ground-truth elements belonging to entity  $i$ ,  $M_i$  is the count of result elements belonging to entity  $i$ , and  $w_1, w_2, w_3, w_4, w_5, w_6$  are predetermined weights, the detection rate and recognition accuracy for entity  $i$  can be calculated as follows:

$$\text{DetectRate}_i = w_1 \frac{\text{one2one}_i}{N_i} + w_2 \frac{g\_one2many_i}{N_i} + w_3 \frac{g\_many2one_i}{N_i} \quad (30.14)$$

$$\text{RecognAccuracy}_i = w_4 \frac{\text{one2one}_i}{M_i} + w_5 \frac{d\_one2many_i}{M_i} + w_6 \frac{d\_many2one_i}{M_i} \quad (30.15)$$

where the entities  $\text{one2one}_i, g\_one2many_i, g\_many2one_i, d\_one2many_i$ , and  $d\_many2one_i$  are calculated from *MatchScore table* (defined in Eq. 30.13) following the steps for every entity  $i$  described in [5]. Here  $\text{one2one}_i$  is the number of correctly detected elements,  $g\_one2many_i$  is the number of cases where more than one detected element match with one ground-truth entity,  $g\_many2one_i$ , is the number of cases where one detected element match with more than one ground-truth entity,  $d\_one2many_i$  and  $d\_many2one_i$  are equivalent cases with detected and ground-truth entity interchanged.

Based on this definition of detection rate and recognition accuracy, the Entity Detection Metric (EDM) is defined which represents a performance metric for detecting each entity. *EDM* is based on the harmonic mean of detection rate and recognition accuracy as defined in Eqs. 30.14 and 30.15. The Entity Detection Metric ( $\text{EDM}_i$ ) of entity  $i$  is defined as follows:

$$\text{EDM}_i = \frac{2 \times \text{DetectRate}_i \times \text{RecognAccuracy}_i}{\text{DetectRate}_i + \text{RecognAccuracy}_i} \quad (30.16)$$

Finally a global performance metric using all entities detected is defined by the combination of all values of detection rate and recognition accuracy. If  $I$  is the total number of entities and  $N_i$  is the count of ground-truth elements belonging to entity  $i$ , then by using the weighted average (arithmetic mean) for all  $\text{EDM}_i$  values, the Segmentation Metric (SM) is defined with the following formula:

$$SM = \frac{\sum_I N_i \text{EDM}_i}{\sum_I N_i} \quad (30.17)$$

## Text Segmentation

Once the page layout is finished, all recognized text areas have to be prepared for the recognition process (see ►Chap. 8 (Text Segmentation for Document Recognition)). In this step text blocks have to be segmented into lines. Evaluation of the text segmentation results is again based on comparison of ground-truth areas with segmented areas. Several evaluation metrics are used all based on ground-truth images where the output label image is compared to its corresponding label ground truth.  $S_i$  denotes the  $i$ th text-line detected with the segmentation method under evaluation, where  $i \in \{1; \dots; N_r\}$  and  $N_r$  is the number of the extracted lines.  $G_j$  denotes the  $j$ th text-line in the ground truth image, where  $j \in \{1; \dots; N_g\}$  and  $N_g$  is the number of text-lines in the input image.

### F-Measure (FM)

For line segmentation again the F-measure (FM) as it was used for the binarization step (compare section “[F-measure \(FM\)](#)”) is again a very important metric. FM here depends on the match MatchScore( $i,j$ ) between  $S_i$  and  $G_j$ . This score is slightly modified to that one presented in section “[Page/Document Layout Analysis](#).” A match is counted if the MatchScore( $i,j$ ) is equal or higher than a threshold  $T_a$ .  $T_a$  is fixed based on experiments.  $No$  is called number of counted match (Eq. 30.18).

$$No = \#\text{MatchScore}(i,j) \geq T_a \quad (30.18)$$

FM is calculated using precision here called detection rate  $DR$  ( $DR = No/N_g$ ) and recall here called recognition accuracy  $RA$  ( $RA = No/N_r$ ) as follows:

$$FM = \frac{2 \times DR \times RA}{DR + RA} \quad (30.19)$$

### Error Rate (U)

The error rate ( $U$ ) is a metric, used to integrate all different types of misclassification into one value  $U$ . The error rate  $U$  is calculated using the following equation:

$$U = \frac{\#(\psi_1 N_s + \psi_2 N_m + \psi_3 N_e + \psi_4 N_p)}{\#N_g} \quad (30.20)$$

where  $N_s$  represents the number of split regions. A region  $G_j$  is a split region if it coincides at minimum with two different  $S_i$  in such a way that the percentage of the  $G_j$  pixels in each segmentation region  $S_i$  is equal or higher than a predefined threshold  $T_b$ .

$N_m$  is a value for the number of merged regions. A region  $S_q$  is called a merged region if it coincides at minimum with two different  $G_j$ , such as the percentage of each  $G_j$  region in  $S_q$  is equal or higher than a predefined threshold  $T_b$ .

$N_e$  is the number of missed regions, where  $G_j$  is a missed region if it does not coincide with any  $S_i$ . Finally  $N_p$  is the number of partially missed regions, where  $G_j$

is called a partially missed region, if the percentage of detected pixels of  $G_j$  in  $S_i$  is less than a predefined threshold  $Tb$ .

The weight values  $\psi_1$ ,  $\psi_2$ ,  $\psi_3$ , and  $\psi_4$  have to be fixed according to test on training pages. The same has to be done for the threshold  $Tb$ .

For the segmentation of lines into words and words into characters, similar metrics are used.

## Character/Text Recognition

Character/text recognition is a main module of a document recognition system as with this module, the image of a character, a word, or a text is transferred into the symbolic representation of a character, word, or text. The performance of this part is essential for the performance of the whole system. Comparing different character recognition systems with each other is not really simple as such systems need all modules mentioned in the paragraphs before, e.g., preprocessing, layout analysis, and segmentation. The evaluation of printed character recognition therefore has to take into account the presence of all modules. Let us divide the problem into the following different steps:

1. The input of the recognition system is the image of a single character.
2. The input of the system is the image of a single word.
3. The input of the system is the image of a single line.
4. The input of the system is the image of a page.

Except for case 1 for all other cases, an error may come from recognition or segmentation error. In the following lists showing different types of errors which may occur as output of a character/text recognition system for each of the above mentioned, four cases of input are given:

### Case 1 errors:

**Substitution** – recognizing one character as another

### Case 2 errors:

**Substitution** – recognizing one character as another. This often happens for structurally close characters.

**Deletion** – ignoring a character because it is regarded as noise or as part of another character.

**Insertion** – one symbol is recognized as two symbols or a noise is recognized as character.

**Rejection** – either the system can't recognize a character or it is not sure in its recognition.

### Case 3 errors:

In addition to errors on character level from case 2, errors on word level may occur:

**Word substitution** – recognizing one word as another.

**Word deletion** – ignoring a word because it is regarded as noise or as part of another word.

**Word insertion** – one word is recognized as two words or noise is recognized as a word.

**Word rejection** – either the system can't recognize a word or it is not sure in its recognition.

**Case 4 errors:** In addition to errors on character level (case 2) or word level (case 3), errors on page level may occur:

**Line deletion** – ignoring a line because it is regarded as noise or as part of another line

**Line insertion** – one line is recognized as two lines or noise is recognized as a line

It is easy to understand that compared to case 1 errors, all other errors depend on classification and segmentation, as only in case 1 the system “knows” that the input is a character. In the following most common metrics used for a character recognition task are presented.

### **Character Error Rate (CER)/Character Recognition Rate (CRR)**

The error rate on character level CER is the percentage of number of wrong recognized characters #CERR in relation to the total number of characters #CHR in the ground-truth text. Wrong recognized characters mean all types of errors: substitution, deletion, insertion, and rejection on character level.

$$CER = \frac{\#CERR}{\#CHR} \quad (30.21)$$

The recognition rate on character level CRR has a similar definition, but here the number of correct recognized characters #CRC is set into relation to all characters.

$$CRR = \frac{\#CRC}{\#CHR} \quad (30.22)$$

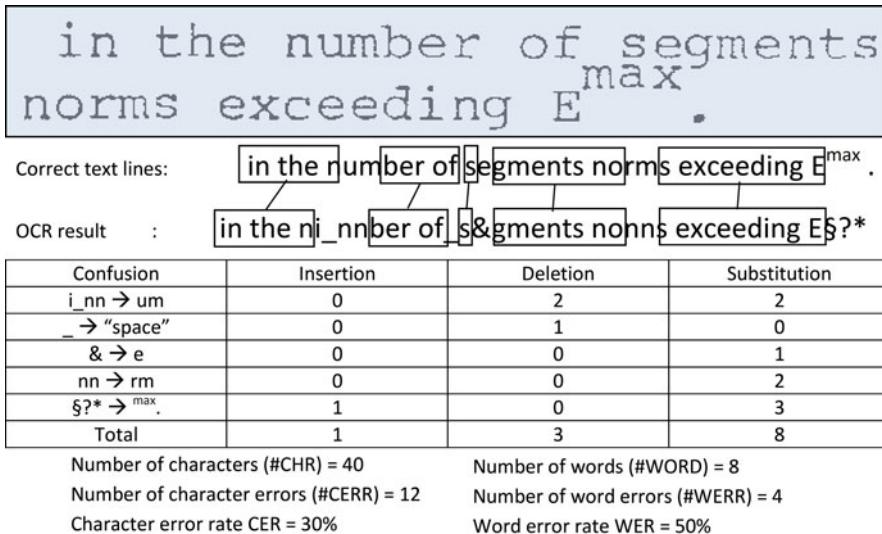
### **Word Error Rate (WER)/Word Recognition Rate (WRR)**

The word error rate and word recognition rate are in the same way defined as for characters. The number of wrong recognized words is #WERR, the number of correct recognized words is #CRW, and the number of words in the ground-truth text #WORD. With these definitions WER and WRR are given by

$$WER = \frac{\#WERR}{\#WORD} \quad (30.23)$$

$$WRR = \frac{\#CRW}{\#WORD} \quad (30.24)$$

Figure 30.7 gives an example of a page segment with recognized text and different types of errors.



**Fig. 30.7** From top to bottom: Part of a text page image, ground-truth text on this image, recognition (OCR) result, errors classified and three types of errors: Insertion, deletion, and substitution

### Accuracy (or Recall)

Accuracy (or recall, see also beginning of section “[Evaluation Metrics](#)” ▶ [Chap. 2](#) (Document Creation, Image Acquisition and Document Quality)) is defined as the ratio of the number of correctly recognized characters (#CCR) to the number of characters in the ground-truth data (#CHR)

$$\text{Accuracy (or recall)} = \frac{\#CCR}{\#CHR} \quad (30.25)$$

Accuracy (or recall) is in the same way defined for any entity recognized, e.g., number of correctly recognized text frames divided by total number of text frames in data set under research.

### Precision

Precision is defined as the ratio of correctly recognized number of characters #CCR to number of characters from OCR output (#COCR)

$$\text{Precision} = \frac{\#CCR}{\#COCR} \quad (30.26)$$

Precision is in the same way defined for any entity recognized, e.g., number of correct recognized words divided by total number of words returned by the recognizer for the whole data set under research.

### **Cost**

The cost value is defined as the weighted sum of editing operations needed to correct the recognized character string. Operations are deletion (**D**), insertion (**I**), and substitution (**S**). The number of each operation is weighted by a predefined factor. The cost is defined as follows:

$$\text{Cost} = w_D \times \#D + w_I \times \#I + w_S \times \#S \quad (30.27)$$

where  $w_D$ ,  $w_I$ ,  $w_S$  are weights which are set to a predefined value given dependent to the text content.

---

### **Evaluation Tools**

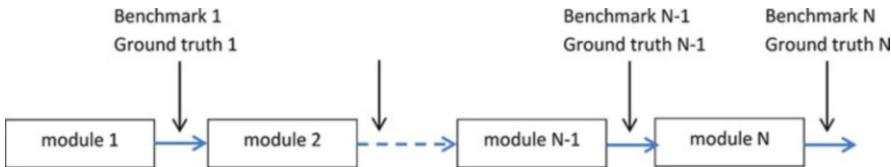
Performance evaluation is one of most important steps in a document-processing system. The complexity of this step depends on the system modules and outputs. Last 20 years, different performance evaluation tools were developed and used as “standard” evaluation tools for specific applications. In this section, in the first part ground-truthing principals and tools are presented and in the second part an overview about competitions and benchmarking campaigns.

### **Ground-Truthing Tools**

In this section, ground truth is defined in terms that were introduced before. In document analysis, GT denotes the optimal output of a system. This term will be adopted and defined for benchmarking in general. To define GT, one should consider that optimal always asks for a reference: “optimal with respect to what?”

For example, in document analysis, GT was mainly defined for OCR output. For the OCR output, GT was simply the sequence of characters that the human would recognize from the input image. So in this case, a person determined what is optimal. Suppose, for a handwritten character GT has to be found. One person recognizes an “A” and another one an “H.” In this case, the question is “optimal with respect to whom?”

When defining GT at the output of some interior module of a system, the optimal output of the module will depend on the subsequent modules following to the final output of the overall system. Thus GT at any interface within the system has to be defined with respect to the following subsystem (Fig. 30.8).



**Fig. 30.8** Ground truth can be defined anywhere in a system

## Ground Truth in Document Analysis

In the previous sections, GT in general was discussed and its role in document analysis explained. In general, it is used for comparison with the actual output of a system to benchmark. In every goal of benchmarking, it has to be known. Thus, it is essential for benchmarking. It was already stressed how GT is gained for OCR systems, but it is still unclear how it can be found for image-processing systems in document analysis without any classification. Thus, in the following preprocessing steps will be in the focus.

Basically, there are several possibilities to determine GT:

- Definition by a person using adequate rules
- Synthetic generation
- Iterative approximation based on an appropriate assessment function

In the following a short overview of the usage of GT as mentioned in literature and our approach will be given. Haralick, Jaishima, and Dori [14] claim that synthetic imagery is necessary for performance measurement because of the limited real images with GT. Their method requires a system existing in an algorithmic form for which only a set of parameters is still undetermined. Moreover, an ideal world of input data has to be defined which will be propagated through the system to constitute GT. Afterwards the input images are perturbed according to an input perturbation model and the output is compared to GT.

Another possibility is asking a human to determine GT, i.e., to replace the assessment function. Lee, Lam, and Suen [20] defined GT by asking five experts and five nonexperts to draw an optimal skeleton. From these images, some kind of average skeleton was computed.

As third method to define GT, an iterative approximation of GT using an assessment function is presented. The idea is to give an input to the assessment function and to judge its closeness to GT by looking at the output. The former output is taken as a reference for the following modifications of the input which is treated as described above. GT is found if the assessment function equals zero. The assessment could be, for example, a human or the subsequent system as mentioned above.

**Table 30.2** Overview of published ground-truthing tools

Tool	Description	Topic(s)
INSEGD [7], 1995	Tools for document database generation	Document
Pink Panther [40], 1998	Ground-truthing and benchmarking document page segmentation environment	Page segmentation
TrueViz [19], 2001	TrueViz (Ground TRUth/metadata Editing & VIsualiZing Toolkit) is a tool for visualizing and editing ground truth and metadata for OCR	OCR, Document
PerfectDoc [39], 2005	Ground-Truthing Environment for Complex Documents	Document
DIAU system [15], 2007	Automatic Ground-truth Generation for Document Image Analysis and Understanding	Document analysis
PixLabeler [35], 2009	Tool for labeling elements in a document image at a pixel level	Document
GEDI [10], 2010	Ground-truthing Environment for Document Images	Document
CBRDIA [6], 2011	Administrative Document Analysis and Structure	Document
Aletheia [9], 2011	Document Layout and Text Ground-truthing System for Production Environments	Document

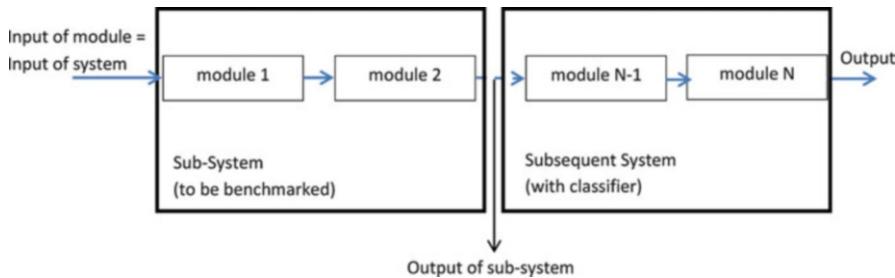
The most important result of this section is that GT depends on the kind of assessment function chosen. Table 30.2 presents an overview of ground-truthing tools developed for document analysis and processing tasks.

## Architecture of an Evaluation Framework

In this section, the view of benchmarking is restricted to document analysis. Furthermore, a focus is set on the preprocessing steps of document analysis.

In benchmarking of document analysis systems, the same aims are pursued as for benchmarking in general. Not only the whole system is the subject of interest but also modules of this system are considered, because the improvement of modules of the system will cause an improvement of the overall system. Considering a document analysis system as a chain of modules, the detection of the weakest point in the chain might be the goal. This could not be done by looking at the system as a whole, thus a more detailed view is needed. In consequence, what is wanted to benchmark might be a single module, a sequence of modules or a subsystem of the overall system, see Fig. 30.9.

Benchmarking for whole systems had been conducted for years already. Generally accepted tests are conducted by institutions like NIST [13] and ISRI [27].



**Fig. 30.9** System in document analysis

In preprocessing, the need of benchmarks has been claimed but few have been published since today. This is mainly due to the fact that finding the ideal assessment is complicated, often subjective, and there is no standard describing test data sets with GT. These problems make benchmarking much harder in preprocessing than in other domains.

### The Assessment Function

Benchmarking is possible when knowing GT. In order to know GT for real data, it has to be generated. A procedure to generate GT can be defined which is very similar to realizing a benchmark: the input data must be known, the subsystem or module, the assessment, and eventually the evaluation function. In most of the cases, these items will be identical to the ones for a benchmark. So, a focus is set on the generation of GT which includes all parts of the final benchmark. Since it is assumed that the input data and subsystem are given, finding the assessment function is the important task. Finding an ideal assessment function is nearly impossible, therefore one has to be satisfied with an approximation. In this context an objective validation of the assessment function is helpful.

Now, the focus is set first on the existent assessment methods and its references in literature. They may be defined by:

- A human
- The subsequent system or
- A direct comparison to ground truth

In the expert approach, a human evaluates the module's output. The advantage of this method is its simplicity; however the disadvantage is that it is quite subjective. This approach was applied by Trier and Taxt [38] and by Nadal and Suen [26]. This method makes sense if the output of the system is produced at a machine/man interface or if the subsequent system is unknown.

The second approach is the replacement of the assessment function by the subsequent system up to a point where GT may easily be defined. For example, in Fig. 30.9 the modules 1 and 2 are the modules to benchmark. The following modules 3 till N represent the subsequent system, thus the assessment function. This is a widely used approach, e.g., Lam and Suen [18], Kanai, Rice, and Nartker [17],

Trier and Jain [37] and Lee, Park, and Tang [21] assessed a sub-system (pre-processing modules) using the results of a subsequent OCR system or other classifiers.

The third method is to replace the ideal assessment by a comparison with GT. In this case, GT has to be explicitly known. If synthetic data are used, GT is also known and a sufficient amount of data may be generated without difficulties. If real world data are used, GT can be defined by experts. The latter is close to the expert approach mentioned above. The shortcoming of real data and GT is the subjectiveness of GT and the difficulty to define rules to enable the human expert to generate objective GT. However, a test with real data produces the most meaningful results, whereas the synthetic data are always restricted in modeling the reality. Knowing the actual output and GT, a distance between them is computed to evaluate the quality of the actual output with respect to GT. (e.g., Haralick, Jaishima, and Dori [14], Lee, Lam, and Suen [20], Palmer, Dabis, and Kittler [28], Yanikoglu and Vincent [40], Randriamasy [32], and Randriamasy and Vincent [33] used this approach.)

These approaches inherit some problems. As the results are dependent on the assessment function and since the optimal assessment function can only be approximated, one has to be aware of errors and their consequences on the results. Performing a benchmark on an arbitrary interface, using the subsequent modules and their final result for assessment, poses the problem of not being able to identify clearly the origin of detected errors, which may lie either in the subsystem to be benchmarked or in the subsequent modules used in the assessment. The assumption of Kanai et al. [17] of an independence of these two types of errors may not hold. Furthermore, and even more important, when trying to generate GT directly at the benchmarking interface, one has to take into account that GT might not be clearly definable and will be faulty itself, making the identification of the error origin even more difficult. After presenting the existing approaches, it is important to point out that it is not evident to judge whether the system to benchmark or the assessment function is erroneous.

## Contests and Benchmarking

Testing recognition systems with large identical datasets is crucial for performance evaluation. Another challenge comes from their complexity because they consist of many specialized parts solving very diverse tasks. The recognition rate is a convenient measure for comparing different systems, but it is a global parameter hardly significant for system component development. To improve the overall system quality, it is essential to know the effectiveness of its modules.

The development of meaningful aspects of system evaluation methods was an important part of the aforementioned annual OCR tests at ISRI. The goal of these tests not only publicized the state of the art of page-reading systems but also provided information for improvement through competition and objective assessment. While much has been achieved concerning the evaluation problem (e.g., [34]),

**Table 30.3** Competitions and benchmarking campaigns overview

Competition/benchmarking	Conference	Topic(s)
<b>Pre-processing and document structure</b>		
Document image binarization contest	ICDAR 2009, 2011 and, ICFHR 2010	Image binarization
Page segmentation	ICDAR 2003, 2005, 2009	Page layout, structure
Text locating	ICDAR 2005	
Book structure extraction competition	ICDAR 2009, 2011	Book structure
Historical document layout analysis competition	ICDAR 2011	Layout analysis
Handwriting segmentation contest	ICDAR 2007, 2009 and ICFHR 2010	Segmentation
<b>Character/text recognition</b>		
OpenHaRT	2010, 2013	Word/text recognition
Arabic handwriting recognition competition	ICDAR 2005, 2007, 2009, 2011 and ICFHR 2010	Word/text recognition
Handwritten Farsi/Arabic character recognition competition	ICDAR 2009	Character recognition
French handwriting recognition competition	ICDAR 2009, 2011	Word/text recognition
Chinese handwriting recognition competition	ICDAR 2011	Word/text recognition
UNLV/ISRI evaluations	1992–1996	Annual tests of OCR accuracy
NIST evaluations	1994–1998	Word/text recognition
<b>Others</b>		
Signature verification competition	ICDAR 2009, 2011, and ICFHR 2010	Signature verification
Music scores competition: staff removal and writer identification	ICDAR 2011	Music score document analysis
Arc segmentation contest	GREC 2001, 2003, 2005, 2007, 2009	Arc segmentation

the availability of tools and data remains an issue for research, as discussed in the paper [25], published in 2005. For example, it is not enough to measure the quality, based on the symbol output of the recognizer, only by considering the word accuracy. The quality of zoning and the segmentation into words or characters represent an important feature of a recognition system and should be evaluated too [36]. A more general concept for evaluating system modules separately is presented in [24]. A list of competitions and benchmarking campaigns organized last 20 years are presented in Table 30.3. Databases, tools, and software used in different competitions and evaluations are collected and presented online by the IAPR TC10 and TC11 [16].

## Conclusion

This chapter describes tools and metrics useful for evaluation of document analysis systems or even parts of systems. To measure the performance of complex systems like systems for document analysis and recognition is a very difficult task. In early days first systems were evaluated only on private data sets; it followed that an objective quality measure was impossible. In recent years more and more data were made available to the research community which made it from that time possible to test and compare systems on same data sets. With these common data sets also common metrics were introduced to calculate a value of system quality. The most challenging task is the evaluation of segmentation modules as in this case not only symbols but also segments have to be compared in size and position. Metrics for different modules of the system are discussed in this chapter; some of them are consolidated and used frequently. Increasing interest in document system evaluation also becomes apparent in more and more open competitions for different fields of applications. Finally tools for ground-truth generation and system benchmarking are presented.

Most interesting new perspective in system quality measure is the fact that tools to support a fast and semiautomatic ground-truth generation allow an effective training with adaptation to a given data distribution. Together with a detailed quality measure even of parts of a system, only a fast system optimization is possible and a better understanding of the weakness of system parts.

Anybody interested in more details is recommended to read papers presenting the results of the competitions or system benchmarks.

---

## References

1. Antonacopoulos A, Gatos B, Karatzas D (2003) ICDAR 2003 page segmentation competition. In: Proceedings the 7th international conference on document analysis and recognition (ICDAR), Edinburgh, pp 688–692
2. Antonacopoulos A, Bridson D, Gatos B (2005) Page segmentation competition. In: Proceedings the 8th international conference on document analysis and recognition (ICDAR), Seoul, pp 75–79
3. Antonacopoulos A, Karatzas D, Bridson D (2006) Ground truth for layout analysis performance evaluation. In: Proceedings of the 7th international conference on document analysis systems (DAS 06), Nelson, pp 302–311
4. Antonacopoulos A, Gatos B, Bridson D (2007) Page segmentation competition. In: Proceedings the 8th international conference on document analysis and recognition (ICDAR), Curitiba, pp 1279–1283
5. Baird HS, Govindaraju V, Lopresti DP (2004) Document analysis systems for digital libraries: challenges and opportunities. In: Proceedings of the IAPR international workshop on document analysis systems (DAS), Florence, pp 1–16
6. Belaid A, D'Andecy VP, Hamza H, Belaid Y (2011) Administrative document analysis and structure. In: Learning structure and schemas from documents. Springer, Berlin/Heidelberg, pp 51–72

7. Bippus R, Märgner V (1995) Data structures and tools for document database generation: an experimental system. In: Proceedings of the third international conference on document analysis and recognition (ICDAR 1995), Montreal, 14–16 Aug 1995, pp 711–714
8. Chhabra A, Phillips I (1998) The second international graphics recognition contest – raster to vector conversion: a report. In: Graphics recognition: algorithms and systems. Lecture notes in computer science, vol 1389. Springer, Berlin/Heidelberg, pp 390–410
9. Clausner C, Pletschacher S, Antonacopoulos A (2011) Aletheia – an advanced document layout and text ground-truthing system for production environments. In: Proceedings of the 11th international conference on document analysis and recognition (ICDAR), Beijing, pp 48–52
10. Doermann D, Zotkina E, Li H (2010) GEDI – a groundtruthing environment for document images. In: Proceedings of the ninth IAPR international workshop on document analysis systems, DAS 2010, Boston, 9–11 June 2010
11. Fenton R (1996) Performance assessment system development. *Alsk Educ Res J* 2(1):13–22
12. Gatos B, Ntiogiannis K, Pratikakis I (2011) DIBCO 2009: document image binarization contest. *Int J Doc Anal Recognit* (Special issue on performance evaluation) 14(1):35–44
13. Geist J et al (ed) (1994) The second census optical character recognition systems conference. Technical report NISTIR-5452, National institute of standards and technology, U.S. Department of Commerce, U.S. Bureau of the Census and NIST, Gaithersburg
14. Haralick RM, Jaisimha MY, Dori D (1993) A methodology for the characterisation of the performance of thinning algorithms. In: Proceedings of the ICDAR' 93, Tsukuba Science City, pp 282–286
15. Héroux P, Barbu E, Adam S, Trupin É (2007) Automatic ground-truth generation for document image analysis and understanding. In: Proceedings of the 9th international conference on document analysis and recognition (ICDAR 2007), Curitiba, Sept 2007, pp 476–480
16. IAPR TC11 Website for Datasets, Software and Tools (2012). [http://www.iapr-tc11.org/mmediawiki/index.php/Datasets\\_List](http://www.iapr-tc11.org/mmediawiki/index.php/Datasets_List), Dec 2012
17. Kanai J, Rice V, Nartker TA (1995) Automated evaluation of OCR zoning. *IEEE Trans TPAMI* 17(1):86–90
18. Lam L, Suen CY (1993) Evaluation of thinning algorithms from an OCR viewpoint. In: Proceedings of the ICDAR'93, Tsukuba Science City, pp 287–290
19. Lee CH, Kanungo T (2003) The architecture of TRUEVIZ: a groundtruth/metadata editing and visualizing toolkit. *Pattern Recognit* 36(3):811–825
20. Lee S-W, Lam L, Suen CY (1991) Performance evaluation of skeletonization algorithms for document analysis processing. In: Proceedings of the ICDAR'91, Saint Malo, pp 260–271
21. Lee S-W, Park J-S, Tang YY (1993) Performance evaluation of nonlinear shape normalization methods for the recognition of large-set handwritten characters. In: Proceedings of the ICDAR'93, Tsukuba Science City, pp 402–407
22. Lu H, Kot AC, Shi YQ (2004) Distance-Reciprocal distortion measure for binary document images. *IEEE Signal Process Lett* 11(2):228–231
23. Lucas SM (2005) Text locating competition results. In: Proceedings of the 8th international conference on document analysis and recognition (ICDAR), Seoul, pp 80–85
24. Märgner V, Karcher P, Pawłowski AK (1997) On benchmarking of document analysis systems. In: Proceedings of the 4th international conference on document analysis and recognition (ICDAR), Ulm, vol 1, pp 331–336
25. Märgner V, Pechwitz M, El Abed H (2005) ICDAR 2005 Arabic handwriting recognition competition. In: Proceedings of the 8th international conference on document analysis and recognition (ICDAR), Seoul, vol 1, pp 70–74
26. Nadal C, Suen CY (1993) Applying human knowledge to improve machine recognition of confusing handwritten numerals. *Pattern Recognition* 26(3):381–389
27. Nartker TA, Rice SV (1994) OCR accuracy: UMLVs third annual test. *INFORM* 8(8):30–36
28. Palmer PL, Dabis H, Kittler J (1996) A performance measure for boundary detection algorithm. *Comput Vis Image Underst* 63(3):476–494

29. Phillips I, Chhabra A (1999) Empirical performance evaluation of graphics recognition systems. *IEEE Trans Pattern Anal Mach Intell* 21(9):849–870
30. Phillips I, Liang J, Chhabra A, Haralick R (1998) A performance evaluation protocol for graphics recognition systems. In: Graphics recognition: algorithms and systems. Lecture notes in computer science, vol 1389. Springer, Berlin/Heidelberg, pp 372–389
31. Pratikakis I, Gatos B, Ntirogiannis K (2011) ICDAR 2011 Document image binarization contest (DIBCO 2011). In: Proceedings of the 11th international conference on document analysis and recognition, Beijing, Sept 2011, pp 1506–1510
32. Randriamasy S (1995) A set-based benchmarking method for address bloc location on arbitrarily complex grey level images. In: Proceedings of the ICDAR'95, Montreal, pp 619–622
33. Randriamasy S, Vincent L (1994) Benchmarking and page segmentation algorithms. In: Proceedings of the IEEE conference on computer vision and pattern recognition (CVPR'94), Seattle, pp 411–416
34. Rice SV (1996) Measuring the accuracy of page-reading systems. PhD thesis, Department of Computer Science, University of Nevada, Las Vegas
35. Saund E, Lin J, Sarkar P (2009) PixLabeler: user interface for pixellevel labeling of elements in document images. In: Proceedings of the 10th international conference on document analysis and recognition (ICDAR2009), Barcelona, 26–29 July 2009, pp 446–450
36. Thulke M, Märgner V, Dengel A (1999) A general approach to quality evaluation of document segmentation results. In: Document Analysis Systems: Theory and Practice. Third IAPR workshop DAS'98, Nagano, Japan, selected papers. LNCS vol 1655. Springer, Berlin/Heidelberg, pp 43–57
37. Trier OD, Jain AK (1995) Goal-directed evaluation of binarization methods. *IEEE Trans PAMI* 17(12):1191–1201
38. Trier OD, Taxt T (1995) Evaluation of binarization methods for document images. *IEEE Trans PAMI* 17(3):312–315
39. Yacoub S, Saxena V, Sami SN (2005) PerfectDoc: a ground truthing environment for complex documents. In: Proceedings of the 2005 eight international conference on document analysis and recognition (ICDAR'05), Seoul, pp 452–457
40. Yanikoglu BA, Vincent L (1998) Pink Panther: a complete environment for ground-truthing and benchmarking document page segmentation. *Pattern Recognit* 31(9):1191–1204
41. Yanikoglu BA, Vincent, L (1995) Ground-truthing and benchmarking document page segmentation. In: Proceedings of the third international conference on document analysis and recognition (ICDAR), Montreal, vol 2, pp 601–604

## Further Reading

Many books about performance evaluation and benchmarking are on the market, especially for benchmarking of computer systems. But there is no book about document analysis methods evaluation. In most image processing, pattern recognition, and document analysis books chapters about evaluation of methods can be found. Readers interested in modern evaluation and benchmarking methods in general may find more details in the following recently published books:

Madhavan R, Tunstel E, Messina E (eds) (2009) Performance evaluation and benchmarking of intelligent systems. Springer, New York

Obaidat MS, Boudriga NA (2010) Fundamentals of performance evaluation of computer and telecommunication systems. Wiley, Hoboken



---

## Correction to: Language, Script, and Font Recognition

Umapada Pal and Niladri Sekhar Dash

### Correction to:

**Chapter 9 in: D. Doermann, K. Tombre (eds.), *Handbook of Document Image Processing and Recognition*,  
[https://doi.org/10.1007/978-0-85729-859-1\\_9](https://doi.org/10.1007/978-0-85729-859-1_9)**

Owing to an unfortunate oversight the second author Niladri Sekhar Dash was missing in the initially published html version of this chapter. He has now been added.

---

The updated online version of this chapter can be found at  
[https://doi.org/10.1007/978-0-85729-859-1\\_9](https://doi.org/10.1007/978-0-85729-859-1_9)

---

# Index

- A**
- Abjad, 430, 891
  - Abugida, 430, 891
  - Academic systems, 429, 448
  - Accidental forgery, 932
  - Accuracy, 44, 50, 51, 54, 1023, 1024, 1027, 1033
  - Acid-free paper, 15
  - Acquisition, 11–60, 984–986, 989–991, 993, 996
  - Action plan, 919, 921, 926, 929, 930
  - AdaBoost, 862, 865, 866
  - Adaptive local connectivity map, 476
  - Address block(s), 715, 717, 720, 721, 723, 724
  - Address block location, 720, 723–724, 727
  - Address database, 715, 720, 729, 730, 744
  - Address interpretation, 723
  - Address recognition systems, 709, 710, 720–723, 732, 733, 744
  - Adjacency
    - grammars, 528, 537
    - matrices, 541, 544
  - Affine covariant, 629
  - AHARONI, 440
  - Algebraic invariants, 617
  - Allograph, 303, 304
  - Alphabet(s), 7–9, 303, 304, 306, 307, 312, 891, 892, 896, 897, 902, 904, 906, 908, 913
  - Alphabetic class, 429
  - Alphabetic fields, 719
  - Ambiguity, 680
  - Analysis of Invoices, 184
  - Analysis System to Interpret Areas in Single-sided Letters (ANASTASIL), 181, 182, 207, 208
  - Analytical word recognition, 725
  - Anchor points, 717
  - Angular radial transform (ART) descriptors, 526, 530, 532
  - Animated, 625, 630, 633, 636, 637
  - Anisotropic Gaussian, 269–271
  - ANN. *See Artificial neural networks (ANN)*
  - Annotation, 630, 639, 966–968, 976, 983–1002
  - Application domains, 182, 198
  - Approximate NN search, 170
  - APTI, 450
  - Arabic
    - alphabet, 430, 434–436, 449
    - extension, 435
    - letters, 432, 435, 436
    - OCR, 450
    - writing styles, 437–438
    - writing system, 432
  - Arabic and Persian signatures, 930
  - Arabic and Syriac are cursive, 428
  - Arabic recognition approaches, 446
  - Aramaic letters, 433
  - Arc detection, 505, 511, 512
  - Architectural drawings, 493, 495, 511–513
  - Area under the curve (AUC), 931
  - Area Voronoi diagram, 146, 147, 156, 157, 161, 163
  - Arial, 323
  - Arrowheads, 493, 512, 513
  - Artificial intelligence, 332
  - Artificial neural networks (ANN), 617, 632, 636, 821, 835
  - Aruspix, 753, 768, 771
  - Aryan language, 304
  - Ascenders and descenders, 325, 442–444, 810, 812, 814
  - ASCII characters, 817
  - Asian scripts, 460–462, 471, 475, 483

- Aspect, 279, 284  
 Assamese, 301, 304  
 Assessment  
     function, 1029–1032  
     methods, 1031  
 Associative graphs, 527, 536–538  
 Assyrian script, 439  
 Attributed grammars, 695  
 Attributed relational graph (ARG), 904  
 Attributive symbols, 750  
 Automatic document processing, 614  
 Automatic letter sorting machines, 709  
 Automatic number plate recognition (ANPR), 846
- B**  
 Background analysis method, 144–146, 148, 149, 158, 164  
 Backtracking, 148  
 Bag-of-features, 871, 873, 874  
 Bag-of-words, 618, 620, 627, 629  
 Bangla, 301, 305, 306, 308, 309, 315, 316, 321, 326, 461, 465  
 Bank check(s), 363  
 Bank check recognition, 67  
 Banners, 640  
 Bar codes, 708, 709, 723  
 Bar lines, 751, 761, 767, 769  
 Bar units, 752  
 Baselines, 260, 263, 265, 266, 275–276, 287, 684, 686–688, 691, 692, 696, 810–815  
 Baselines extraction, 442, 686, 687, 691, 692  
 Base-region points, 311  
 Baum-Welch algorithm, 350  
 Bayesian combination rules, 739  
 Bayesian network, 902, 903  
 BBN, 335, 354, 355  
 Behavioural characteristics, 918  
 Belga Logos, 632, 640  
 Bengali  
     alphabet, 304  
     script, 303, 304  
 Bezier curve, 893  
 Bibliographic  
     citation, 209  
     metadata, 209  
 Biblio system, 196, 213  
 Bidimensional  
     grammars, 513  
     patterns, 514
- Bidirectional long short-term memory (BLSTM), 822, 823  
 Bi-gram probability, 410, 411, 415  
 Bi-lingual script, 314, 316  
 Billboards, 629, 640  
 Binarization, 44, 49, 54, 189, 262, 275, 335, 336, 338–339, 355, 466, 475–477, 716, 814, 984, 986, 989, 990, 992–994, 1018–1022, 1024, 1033  
 algorithms, 716  
 of handprinted text, 367  
 skew correction, and noise, 755  
 Binary  
     classifier, 738  
     images, 716  
     mask, 633  
 Bipartite graph, 540  
 Black-and-white document, 144, 147  
 Bleed-through, 49, 95, 97, 99  
 Blind attacker, 932  
 Blob noise, 621  
 Block(s), 752  
 Block adjacency graphs, 479  
 Block overlay phenomena, 778  
 Blur, 46, 48, 53, 54, 57, 59, 573, 844, 853, 856, 857  
 Blurring, 623, 850, 854, 856–857  
 Boldface, 322, 325  
 BongNet, 902, 903, 910  
 Book  
     binding, 52, 57–59  
     scanners, 846  
 Boosting methods, 540, 542, 543  
 Borda count, 607  
 Border removal, 102, 103  
 Born digital documents, 687, 690  
 Bottom-up strategies, 137, 144, 148, 149, 155, 157, 565, 566, 571, 722, 723, 928, 940, 941  
 Bounding box, 367, 377, 382  
     projections, 761, 764  
 Boxing systems, 723  
 Brain stroke risk factors, 941  
 Branch-and-bound, 628, 632  
 Brightness, 35, 37, 40, 43, 44  
 Broadcasting industry, 623  
 Brodatz textures, 314  
 Brute-force attacker, 932  
 B-splines, 125, 126  
 Burmese, 314, 318  
 Business Letter, 181, 182, 185, 186, 192, 197, 199, 204–210, 216

- C**
- C4.5 algorithm, 616
  - Calligraphic
    - characters, 852
    - interfaces, 951
  - Calliope, 20
  - Camera(s), 44–47, 52, 59
  - Camera-based OCR, 844–850, 860, 861, 872, 874
  - Camera-captured, 546
  - Cancellable biometrics, 931
  - Candidate(s), 274, 280, 281, 284
  - Candidate staff-line segments, 757
  - Canonical structure, 462
  - Caption texts, 489, 848, 854, 855, 862, 866, 872
  - Carbon paper, 28, 31
  - Car license plates, 846
  - Case-based reasoning (CBR), 197, 211
  - CBIR. *See* Content-based image retrieval (CBIR)
  - CCH. *See* Co-occurrence histograms (CCH)
  - CDDK. *See* Class dependent domain knowledge (CDDK)
  - Census forms, 363
  - Chain Codes, 120–123, 369, 377, 384, 525, 527, 534
  - Character
    - forms, 363
    - recognition, 331–356, 359–385, 427–453, 459–484, 524, 525, 534, 707, 718, 725, 729, 737, 739, 1013, 1025, 1026, 1033
    - segmentation, 466, 467, 469, 807, 811, 824, 830, 834
  - Character error rate (CER), 1026
  - Character recognition rate (CRR), 1026
  - Character Shape Coding, 807, 809–811, 817–818, 824, 826–828
  - Charged couple devices (CCDs), 39
  - Chart analysis, 998
  - CHAYIM, 440
  - Check
    - forms, 731, 734–736
    - models, 735
    - processing, 705–745
    - readers, 710
  - Check 21, 744
  - Check recognition applications, 708–711, 714–716, 718, 731, 735, 741, 745
  - Check recognition system, 707, 708, 712, 731, 733, 743, 744
  - Chemical drawing recognition, 973–974
  - Chinese, 296, 305, 307–310, 312–316, 318–321, 325
  - Chinese Academy of Sciences (CASIA), 465
  - Chinese and Japanese signatures, 930
  - Chinese handwriting recognition competition, 463, 465
  - Chinese, Japanese, and Korean (CJK)
    - ideographic character, 462
    - scripts, 460–463, 465–474
  - Chrominance, 628
  - CIDK. *See* Class independent domain knowledge (CIDK)
  - City name recognition, 720, 722, 725–727
  - Class dependent domain knowledge (CDDK), 197, 204
  - Classes of forgeries, 932
  - Classification, 463, 470–473, 478, 479, 482–484, 535, 538–543, 545, 760–764, 766–769
  - Classification units, 478, 479
  - Classifiers, 279, 281, 282, 284–286, 446–449, 451
  - Classifiers combination, 383, 689, 692
  - Class independent domain knowledge (CIDK), 197, 207
  - Client-entropy measure, 930
  - Cluttered background, 623
  - CNN. *See* Convolutional neural network (CNN)
  - Coding scheme, 809, 817, 826, 827
  - Cognitive psychology, 195
  - Color, 18–20, 28–30, 33, 34, 36–38, 40, 44, 46, 47, 50, 598–600, 603, 605–610, 612–615, 617, 623, 629, 631, 635, 848, 849, 851, 853, 862–864, 868
  - clustering, 147, 164, 165, 626, 864, 867, 868
  - descriptors, 628
  - naming, 604, 611, 638
  - printing, 30, 34–36
  - spaces, 625–628, 631
  - Color clustering, dither, 164
  - Combination rules, 934
  - Command parameter, 921, 930
  - Commercial
    - language, 301
    - OCRs, 452
    - systems, 429, 448, 450
  - Committee of experts, 716, 719
  - Comparative study, 759, 762

- Comparison, 1012–1016, 1018, 1022, 1024, 1029, 1031, 1032  
 Competitions, 759, 1013, 1022, 1028, 1032–1034  
 datasets, 448  
 Complementary metal-oxide silicon (CMOS), 39, 40  
 Complete graphic object recognition, 963–964, 976  
 Complex background, 844, 851, 861  
 Complexity of a signature, 929  
 Compression, 46, 48–50  
 Compression artifacts, 623  
 Computational complexity, 714, 727  
 Computer language, 295  
 Concavity features, 343  
 Conditional random field (CRF), 870  
 Condorcet method, 607, 609  
 Confidence scores, 715, 716, 718  
 Confidence values, 719, 731, 734, 739–741  
 Conjugent consonants, 477  
 Connected components  
     analysis, 85, 99, 101, 102, 496, 498  
     labeling, 74–76, 100, 102  
 Connected components based method, 148, 149  
 Connectionist Temporal Classification (CTC), 821, 822  
 Consistency checking, 579  
 Consistency model, 924  
 Consonant, 462–464, 477, 478  
 Constraints, 762, 764, 765  
 Content  
     ownership, 623  
     stream, 779, 781  
 Content-based image retrieval (CBIR), 544, 545  
 Content-based video retrieval, 848, 849  
 Contest, 1013, 1018, 1032–1033  
 Context/contextual, 680, 681, 683, 685, 691, 693–695, 862, 864, 873, 919, 924, 931  
     information, 627, 638, 690, 696, 752, 761–765, 770  
     knowledge, 568, 571, 582  
 Contours  
     extraction, 816  
     tracking, 756–760, 767  
 Contrast, 40, 43, 44, 47–50  
 Control variable, 925  
 Convolutional neural network (CNN), 471–473  
 Co-occurrence histograms (CCH), 626, 631  
 Copy machines, 20  
 Corner, 851, 863, 864, 866, 870  
 Correlation filter, 545  
 Cost value, 1028  
 Courtesy amount recognition, 710, 711, 719, 731, 733–741  
 Creole languages, 295  
 Critter, 323  
 Cross-correlation, 100, 102, 114, 118  
 Cross-related attributes, 194, 195  
 Cryptographic key generation, 931  
 Cryptography, 303  
 Cube search, 906  
 Cultural habits, 930  
 Currency sign, 731, 736, 738  
 Cursive script, 890, 892, 896, 902, 906–908  
 Curvature, 317, 534, 535  
 Curvature Scale Space (CSS), 601  
 Curvilinear, 260, 265, 266, 268  
 Cyrillic, 305, 314, 316, 318, 319, 321
- D**
- Damaged characters, 74, 95, 97, 99, 101, 127  
 Data acquisition devices, 922  
 Data-driven approaches, 214  
 Data reduction, 893, 895  
     techniques, 928  
 Dataset(s), 448–451, 453, 596, 602, 610, 612, 613, 620, 631, 632, 636, 639–641, 983–1002  
 Data variability, 583  
 DAVOS, 182, 209  
 DCT, 863, 868  
 3-D document shape, 123, 126  
 Decision  
     making, 719–720, 733, 736, 741  
     strategy, 739  
 Decorative characters, 872–874  
 Defects, 46, 48, 51, 53, 57  
 Deformations, 988  
 Degradation, 68, 260, 282, 984, 986, 988, 991, 994, 998  
 Degraded document, 812, 817, 824  
 Delaunay triangulation, 146, 149, 155–158, 213  
 Delayed strokes, 890, 891, 894, 896, 901, 905  
 Delta features, 895  
 Denoising, 953, 954, 975  
 Descenders, 325, 442–444  
 Descriptors, 526, 528–539, 541–547  
 Deskew(ing), 112, 894  
 Deslant, 112  
 Detecting expressions in pen-based input, 685, 688

- Detecting mathematical expressions, 686  
Detection, 592, 595, 596, 604, 614–616, 618–641  
Detection rate, 1023, 1024  
Devanagari script, 301, 303–306, 308, 309, 314–321, 326, 461, 465, 475, 477–483  
Device interoperability, 940  
Dewarping, 123, 126, 992, 994, 1000  
3D gradients, 634  
Diacritic(al), 303, 304, 891, 901, 902 marks, 428, 434, 435, 438, 441, 442 marks removal, 441  
Diagram(s), 492, 493  
Diagramming, 968, 970, 971, 976  
Dialects, 681, 695, 697  
Diaspora, 300  
Dictionary, 66, 67, 303  
Differential invariants, 617, 618  
Digital-born documents, 649, 651, 652, 655, 662, 667–672, 674  
Digital camera images, 845  
Digital library(ies), 69, 180, 184, 198, 224, 226, 244, 245, 337, 353, 809, 811, 836  
Digital paper, 777  
Digitizing tablets, 922, 932–934  
Digit recognizers, 738  
Digit strings, 722, 897–899, 907  
Dilation, 153  
Dimension sets, 512, 513  
Directed graphs, 526, 527  
Directional element features, 344  
Direct matching points (DMPs), 929  
Disambiguation, 200  
Displaying recognition results, 698  
Distance, 530, 536, 538–542 calculation, 74, 75 reciprocal distortion metric, 1020–1021  
Distance-based approaches, 506  
Distance-based classification techniques, 934, 935  
Distance-based verification techniques, 926  
Distortion, 616, 621, 623, 626, 628, 986, 990, 991, 1002  
Dithering, 30, 34–37  
Ditto machines, 24, 29, 31  
Docstrum, 154, 339  
Document analysis systems, 68 categorization, 192–193, 716–717 classification, 595, 614, 623, 640 degradation, 189 genre, 226, 234, 244 layout analysis, 778, 791, 799, 1021–1024 models, 712, 717 recognition, 1024, 1025 segmentation, 258, 259 segments, 778, 780, 798 similarity, 807 understanding, 178–186 vector, 828, 829, 834  
Document image analysis, 301 binarization, 74, 77–79, 84, 91, 93, 127 enhancement, 74, 95, 99, 127, 476 normalization, 74, 104, 127 processing, 806, 807 retrieval, 138, 806–812, 823–838  
Document layouts analysis, 1021–1024  
Domain knowledge, 563, 571, 577 semantics, 695  
Dot gain, 16, 36  
Dot matrix printers, 27, 30  
DTW. *See* Dynamic time warping (DTW)  
Dynamic information, 889, 890, 904, 912 programming, 5, 366, 380–382, 627, 814, 816, 828  
Dynamic and tempo markings, 750, 751, 753  
Dynamic time warping (DTW), 531, 820, 835, 923, 926–929, 934–937
- E**
- Early fusion, 602, 606–607  
Eastern scripts, 428, 450  
Edge detection, 165 pixel, 626  
Edge-backpropagation, 617, 620  
Edit distance, 527, 536, 539–541  
eInk, 38  
Elastic matching, 896  
Electronic books, 37–38 check remittance, 708 displays, 37 documents, 69 pens, 922 whiteboards, 838  
Electrophotographic printing, 31, 32, 52  
Embedded expressions, 686–688, 697  
Embedding, 524, 527, 536, 538–540, 542, 543, 546  
Emission, 405–410  
Ending, 752

- Engineering drawings, 66, 67, 491, 493–496, 512, 516  
 Enrolment stage, 926  
 Entity Detection Metric (EDM), 1023  
 Entropy, 83  
 ePaper, 38  
 ePens, 23, 37  
 Equal error rate (EER), 931, 935, 938, 939  
 eReaders, 37, 38  
 Erosion, 153  
 Error rates, 216, 219, 709–711, 713, 714, 730, 731, 734, 741, 1024–1027  
 Estrangelo, 430, 431  
 Ethiopic, 314, 318  
 Euclidean, 539  
 Euramerican scripts, 315  
 Evaluation, 1011–1034  
 Evaluation framework, 1030–1031  
 Expectation-maximization, 350  
 Expert knowledge, 580  
 Explicit character segmentation, 725, 736, 737  
 Extraction of fields, 735
- F**  
 Facsimile, 32, 44, 55  
 False  
   acceptance, 931  
   rejection, 931  
 False-positives, 285  
 Fax machine, 30, 32, 33, 44  
 Featural alphabet, 429, 902  
 Feature-based word spotting model, 832  
 Features  
   character, 361, 364, 373, 374, 376  
   extraction, 524, 528, 529, 531, 538, 541, 544, 719, 738, 739, 892, 895–896  
   sets, 725, 737  
   vector, 814, 817–819, 821, 822, 824, 831, 834, 835  
 File formats, 987  
 Finite-state automata, 738  
 First-order language, 212  
 Fisher classifiers, 616, 618  
 Fisher linear discriminant (FLD), 316, 321  
 Flatbed scanner, 39–44, 46, 57, 59  
 Flood-fill algorithms, 102–103  
 F-measure (FM), 202, 1017, 1019, 1024  
 Font(s)  
   formation, 322  
   generation, 294, 322, 326  
   recognition, 291–327  
 Foreground analysis, 144, 148  
 Form(s), 66, 67, 69, 647–674, 709, 714, 717, 720–722, 731, 734, 743, 744  
 Formal language, 295  
 Formal problem statements, 698  
 Formats, 985–988, 992, 995  
 Form(s) processing, 224  
 Fourier  
   descriptors, 526, 530, 533  
   transform, 501  
 Fourier–Mellin transform, 530  
 Fractal features, 313, 315, 316  
 Freehand/skilled forgeries, 932  
 French national archives, 214  
 Frequency (IDF), 835  
 Frequency (TF) and inverse document, 835  
 FRESCO, 180, 182  
 Full interpretation systems, 556  
 Function features, 924–926, 928, 934, 937  
 Fusion, 602, 604, 606–607, 617, 639  
 Fuzzy  
   classification, 317  
   grammars, 691, 693  
   logic, 447, 449  
   rule matching, 195
- G**  
 Gabor filters, 314–317, 319, 320, 342, 345, 346  
 Gamera, 753, 771  
 Gamut, 36, 771  
 Gaussian, 269–271  
 Gaussian kernel, 535  
 Gaussian mixture models (GMM), 347, 350, 366, 821  
 Gaussian smoothing filter, 94  
 Generalization, 596, 621, 635  
 Generalized median graph, 540  
 Generation of synthetic signatures, 933  
 Generic Fourier descriptor, 530, 533  
 Generic method, 661–663, 667, 672  
 Genetic algorithms, 536, 542  
 Geometric  
   features, 496  
   information, 181, 783  
   knowledge, 207  
   relationships, 783, 798  
   tree, 206, 208, 209  
 Gestural Shortcuts, 957  
 Gesture  
   mode, 950, 955  
   recognition, 952, 953, 955, 956, 958–960, 962, 975–977  
   vocabulary, 958

- Ghega datasheets, 218  
Ghost characters, 438, 439  
Ghost character theory, 438  
Gill, 440  
GLCM, 313, 319, 320  
Global  
    approach, 313  
    features, 920, 926, 937, 940  
    information, 901  
    level, 926, 941  
    shape, 901, 904  
Global-based vision classifiers, 446  
Global thresholding, 78–86, 88, 91, 93  
Global thresholding techniques, 78–86  
Glyph, 8, 10  
Gobbledoc, 180, 182  
Google book search, 198  
Gradient  
    features, 342–344, 468, 470, 471, 480  
    orientation, 819, 824  
Gradient, structural, and concavity (GSC), 480–481  
Graffiti, 896, 899, 900, 911  
Grammars  
    inference, 536  
    scale factor, 411  
Grammatical, 574, 577  
Graph  
    descriptors, 532, 536, 538  
    grammars, 691, 693, 695, 765  
    hierarchy, 767  
    prototypes, 536  
Graph-based  
    models, 187  
    representations, 479  
    techniques, 759  
Graph cut textures, 635  
Graph-edit distance, 536  
Grapheme, 302–304, 731, 739, 740  
Graphic(s)  
    documents, 553–587  
    document understanding, 964–966, 976  
    object, 951–953, 958, 962–964, 975  
    primitives, 781, 783  
    recognition, 68, 750, 759, 952, 962–964, 989, 992, 996–999  
    symbols, 302, 303  
Graphical  
    documents, 490, 491, 494–504, 512, 513, 516  
    entities, 638  
    primitives, 752–754, 760–762, 764, 767, 768  
Graphic-rich documents, 504  
Graphics recognition workshop (GREC), 491, 492, 511  
Graphism multiplication, 436, 437  
Grass script, 908  
Gravure, 19, 24, 30, 31  
GREC. *See* Graphics recognition workshop (GREC)  
Ground truth, 985–990, 992–996, 1012–1016, 1018–1024, 1026–1031, 1034  
Ground-truth datasets, 217, 219  
Ground-truthing tools, 1028, 1030  
Growth, 266, 268–269, 271, 273, 287  
GSC. *See* Gradient, structural, and concavity (GSC)  
Guide, 567, 582, 583  
Guidelines, 736  
Gujarati, 461  
Gurmukhi, 461, 475
- ## H
- Haar wavelet transform, 445  
Halftoning, 35, 36  
Hallucination, 855  
Han, 460, 462  
Han-based script, 313  
Handheld devices, 923, 937  
Handprinted, 359–385  
Hand-sketched, 527, 543, 546  
Handwriting, 23, 51, 52, 986, 990, 991, 1002  
Handwriting recognition, 317  
Handwritten  
    addresses, 710, 722, 723  
    imaged documents, 813, 814  
    mathematical symbols, 688  
    page, 136, 172  
    scores, 758, 764, 766, 768  
    text recognition, 990  
    word representation, 809  
Handwritten document  
    images, 809, 812, 820, 829–832, 835  
    retrieval, 808, 814, 835  
Handwritten music  
    scores, 751, 753, 754, 759, 762, 766–768, 770, 771  
    symbols, 754  
Handwritten word recognition (HWR), 812, 820  
Hangul, 460  
Hanja, 460, 462  
Harmonic mean, 1017, 1019, 1023  
Hashing, 544  
Hausdorff–Besikovich dimension, 316  
Heading, bar units, ending, 752

- Health conditions, 941  
 Hebrew, 314, 318, 320, 428, 430, 434, 439–441, 445, 450, 452  
 Hebrew alphabet, 439, 440  
 Hectograph, 24, 29  
 Helvetica, 323  
 Heuristics, 561–563, 567, 579, 607, 618–621, 761, 768, 769  
 Heuristics rules, 725, 737, 738  
 Hidden Markov models (HMMs), 85, 93, 335, 336, 339, 348–353, 362, 371, 378, 380–382, 384, 393, 405–410, 412, 415–419, 421, 447–451, 482, 688, 689, 718, 725, 727, 738, 739, 762, 763, 768, 770, 771, 820, 821, 824, 835, 837, 897–900, 902–905, 907–912  
 Hidden structures, 781, 783  
 Hierarchical composition, 904  
 Hierarchical X-Y tree, 616  
 High speed scanners, 716  
 High-stability regions, 929  
 Hindi, 296, 301, 304, 314, 316, 321  
 Hiragana, 460, 466  
 Histogram  
     descriptors, 534  
     features, 342–344  
 Historical, 259, 260, 262, 264, 276, 282, 283, 287  
     documents, 461, 476  
     documents images, 809  
     manuscripts, 814, 829, 831  
 History of automated document analysis, 555  
 HMMs. *See* Hidden Markov models (HMMs)  
 Holes, 316, 321, 325  
 Holistic  
     approach, 446  
     features, 740  
 Holistic word  
     recognition methods, 727  
     representation, 809, 811, 812, 818, 828, 835  
     word recognition, 725  
 Hook, 894  
 Horizontal belt, 150  
 Horizontal projection histogram, 814  
 Horizontal text layout, 466  
 Hough, 530  
 Hough transform, 74, 75, 77, 113, 117, 149–151, 445, 498, 508, 511, 512, 514  
 Human document analysis, 555  
 Human interaction, 582  
 Human perception, 195, 196, 211, 215  
 Hu moment invariants, 601, 602  
 HWDB/OLHWDB, 465  
 HWR. *See* Handwrittenword recognition (HWR)  
 Hybrid  
     approaches, 209, 447, 448, 781  
     methods, 78  
     thresholding techniques, 90–91  
 Hybrid-level classifiers, 446, 447
- I**
- IFN/INIT, 448  
 Illumination, 618, 623, 626, 628, 635, 638  
 Image  
     acquisition, 707, 714, 716, 732, 733, 736  
     databases, 707, 708, 730, 731, 742  
     enhancement, 716  
     features, 806–809, 811, 820, 832  
     preprocessing, 1014–1017, 1029  
     quality, 68  
     search, 846  
     segmentation, 779, 787  
 Image-based documents, 651–667, 674  
 Imaged document retrieval, 807, 823, 827, 836, 838  
 Image features, 811  
 Image processing algorithms, 74–75, 127  
 Impact printing, 23, 24, 30, 52  
 Implementation of an analysis system, 559  
 Implicit segmentation, 725, 736, 738, 739  
 Indexing, 600, 606, 614, 638, 639  
 Indic scripts, 460–465, 475–483  
 Individual dissimilarity values, 937  
 Industrial  
     applications, 707–714  
     systems, 707, 725, 727  
 INEX book corpus, 202  
 Information  
     extraction, 1017  
     fusion, 606  
     spotting, 556, 584, 586  
 Information retrieval (IR), 198, 216, 806, 809, 812, 823, 826, 829, 831, 832, 836, 1017  
 Information retrieval (IR) systems, 606  
 INFORMys, 182, 205, 217–218  
 INFORMys invoice dataset, 217–218  
 Inkjet, 18, 20, 32–34, 51  
 Inkjet printer, 718  
 Inks, 15–20, 22, 36, 37, 47, 52  
 Inpainting, 625, 633–636  
 Integral images, 627  
 Integration strategy, 713, 716

- Interactive whiteboard, 838  
Interest points, 545, 618  
Interpretation  
    context, 556, 559, 564, 567, 581–585, 587  
    strategies, 560–564, 572, 581  
Intersession variability, 925, 930  
Intra-class variations, 619  
Invariant features, 871, 873  
Inverse document frequency (IDF), 825, 832, 835  
Invoice  
    processing, 203–205  
    reading system, 182, 203  
IPSAR, 450  
Isolated character recognition, 900  
Iterative thresholding methods, 83
- J**  
Japanese, 305, 306, 313, 314, 316, 318, 320, 321, 325
- K**  
Kalman filtering, 630  
Kanji, 460, 462, 466  
Kannada, 304, 306, 316, 320, 321, 461, 477, 478  
Karhunen-Loëve transform, 366, 375  
Kashti, 438  
Katakana, 460  
Kernel  
    function, 540, 542, 543  
    trick, 542  
Kerning, 274, 287, 288  
Keypoint, 860, 863, 868  
Keypoint correspondence, 627–628  
Keyword  
    recognition, 719  
    spotting, 805–838  
Keyword-based strategies, 727, 728  
*k*-Nearest neighbors (*k*-NN), 146, 149, 154, 155, 161, 167, 168, 762, 768, 769, 771  
Knowledge, 178, 179, 182–185, 196, 197, 199, 201–205, 207, 208, 210, 212, 214–216, 219  
Knowledge based  
    approaches, 197  
    systems, 563  
Knowledge modeling, 554, 560–563, 571–573, 582, 587  
Konkani, 304  
Koranic addition, 432  
Korean, 313, 314, 318, 319, 321  
KOREN, 440  
Kullback–Leibler (KL) divergence, 539  
Kurdish Sorani vowels, 436
- L**  
Labeled graphs, attributed graphs, 536  
Labeling algorithms, 188, 216  
Language  
    differences, 921, 930  
    identification, 293–296, 300–302, 307, 325, 326  
    independent, 811, 817, 824, 828  
Language acquisition device (LAD), 295  
Language models (LMs), 336, 348, 351–352, 463, 465, 473, 474, 479, 482–484, 909  
Large vocabularies, 722  
Large-volumes lexicons, 720, 730  
Laser printer, 20, 32, 34  
Late fusion, 602, 604, 607  
Latin, 305, 313, 314, 316–321  
Latin based scripts, 313  
Lattices, 738, 741  
Layout  
    analysis, 136–139, 144, 150, 615, 616, 683, 685, 687, 690–696, 710, 711, 715, 717–718, 720, 731  
    trees, 683–687, 691–698  
Learning-based approaches, 196  
Left (right) reservoir, 310  
Left-to-right, 343, 344, 350, 351  
Legal amount recognition, 731, 733–735, 739–742  
Legal amounts, 711, 717, 719, 731, 733–737, 739, 741  
Legendre moments, 533  
Legibility, 278  
Letterpress, 19, 24–26, 29  
Letters have position-dependent shapes, 428  
Level-building algorithm, 898  
Level set, 266, 271–272, 282, 283, 287  
Lexicon, 66, 67  
Ligatures, 360, 367, 368, 372, 428, 432, 436–438, 442, 443, 449, 452, 894, 898, 899, 902, 903, 907, 908, 910  
Ligatures model, 899, 907  
Lightface, 322  
Line  
    drawings, 554  
    extraction, 814  
    tracking, 858  
Line adjacency graph (LAG), 758, 767

- Line-fitting, 506, 508  
 Line segmentation, 259–260, 263–271, 287, 475, 477, 478, 653, 657, 990, 996, 998  
 Linguistics, 66–68, 294, 295, 297–304  
 Lithography, 24, 28, 29  
 LMs. *See* Language models (LMs)  
 Local  
     approach, 315, 324  
     decisions, 896  
     level, 926, 941  
     minima, 442–444  
     neighborhood structure, 837  
     ones, 926  
     texture, 863  
     thresholding techniques, 86–90  
 Local-based approaches, 447  
 Local-based vision classifiers, 446  
 Local gradient histogram (LGH), 821  
 Localization, 614, 621, 623, 626, 627, 636, 638  
 Local (adaptive) thresholding, 78  
 Logical  
     component, 778, 779, 781, 782, 784, 792, 799  
     labeling, 180, 182, 184, 187, 188, 193–197, 204, 208, 210, 212, 216–220  
     language, 295  
     objects, 178, 180, 181, 184, 186, 197, 204, 206–209, 213, 214, 217  
 Lognormal  
     impulse response, 921  
     primitives, 921  
     velocity profile, 921  
 Logo  
     database, 620, 640, 641  
     detection, 596, 604, 614, 615, 621–641  
     models, 615, 618, 619, 623, 629, 635, 638  
     recognition, 595, 596, 614–622, 625, 629, 639  
     removal, 596, 622–641  
     spotting, 614, 615, 618, 639  
 Logograms, 307  
 Logographic, 467  
 Logographic script, 429  
 Logosyllabic, 892  
 Log response time, 921  
 Log time delay, 921  
 Long range correlation, 901, 904  
 Long-term modifications, 930  
 Loosely-structured documents, 186  
 Low-force forgery, 932  
 Low resolution, 623, 844, 850, 853–856, 870, 873  
 Luminance histogram, 94  
 Lyrics, 750, 751
- M**  
 Maatras, 462  
 Machine  
     learning, 196, 197, 210, 211, 213, 219, 293, 303, 696, 861, 862, 864–866, 874, 875  
     printed, 360, 367, 368  
 Madrigals, 766, 767  
 Magazines, 181–183, 199, 200, 206, 209–214  
 Mailroom solutions, 744  
 Mail sorting, 183, 184, 216, 333, 335, 337  
 Majority voting, 196  
 Malayalam, 314, 316, 319, 320, 461  
 Manhattan layouts, 137, 140, 141, 188  
 Manipuri, 301, 304  
 MAP-MRF framework, 468  
 MAP rule, 541, 542  
 Maps, 491–496, 502, 511, 513, 514, 516, 517  
 Marathi, 301, 304, 316  
 Markov chain, 350, 351, 405  
 Markov random fields (MRFs), 167, 170, 172, 369, 468, 537, 633, 636, 862, 869, 870  
 Markov source model, 909  
 MAT. *See* Medial axis transform (MAT)  
 Matched wavelet, 167, 170  
 Matching, 527, 536, 537, 541, 545  
 Matching algorithms, 527, 528, 535, 536, 541, 819, 820, 829, 836, 837  
 MatchScore, 1022–1024  
 Math dialect, 695  
 Mathematical  
     expressions, 974, 990, 992, 993, 999, 1000  
     information retrieval, 682, 687  
     morphology, 75, 76, 92  
     notation, 679–698  
 Math recognition system, 681–685, 695–698  
 Matrix recognition, 691, 694–695  
 Maximal empty rectangles, 146, 147, 149, 160  
 Maximally stable extremal region (MSER), 867, 868  
 M-band wavelet feature, 170  
 Mean square error (MSE), 1016, 1019  
 Mechanical engineering, 974  
 Medial axis, 370  
 Medial axis transform (MAT), 525, 529, 537  
 Media team Documents Dataset, 217  
 Medical Article Records Ground-truth dataset, 21, 211, 212  
 Medical forms, 363

- Medical invoices, 193, 203, 205  
MEDLINE, 209  
Merged regions, 1024  
Metadata, 781, 783–785, 793, 930,  
    931, 941  
Metric, 528, 539, 546  
Microfiche, 32, 34, 47  
Microfilm, 32, 34, 47  
Middle eastern  
    character recognition, 427–453  
    scripts, 428, 450  
    writing systems, 429, 430  
Million Book project, 198  
Mimeograph, 24, 31  
Minimum spanning tree (MST), 146, 149, 154,  
    156, 164, 689, 692, 694  
Misclassification Penalty Metric (MPM), 1021  
Missed regions, 1024, 1025  
Mixed languages, 910  
Mobile devices, 744  
Model(s), 12, 27, 36, 40, 41, 46, 48, 51–57, 60  
Model-based classification techniques, 937  
Model-based techniques, 928, 938, 941  
Model decoding, 898  
Model-driven, 214  
Model-driven (top-down), data-driven  
    (bottom-up), 781  
Modeling radicals, 467, 468  
Modified fractal signature (MFS), 315  
Modified quadratic discriminant function  
    (MQDF), 335, 345–347, 471,  
    472, 482  
Modified quadratic discriminant function  
    (MQDF) classifier, 472  
Moiré, 36  
Moments, 313, 318, 368, 370, 372–376,  
    378, 384  
Monogenetic theory, 298  
Mono-scriptor applications, 711  
Morphological  
    analysis, 200, 717, 723  
    operations, 74, 75, 96, 97, 99, 101, 761,  
        764, 767  
    operators, 75  
    processing, 633, 717, 736  
    structure, 448  
Morphology  
    analysis, 496–498, 502, 503  
    based method, 153  
Motion blurring, 850  
MPEG-7 standard, 526, 534  
MQDF. *See* Modified quadratic discriminant  
    function (MQDF)  
MRFs. *See* Markov random fields (MRFs)  
MS Courier New, 323  
MS Georgia, 323  
MST. *See* Minimum spanning tree (MST)  
MS trebuchet, 323  
Multichannel Gabor filter, 166, 167  
Multi-expert  
    approach, 928, 941  
    decisions, 719  
Multifont, 65  
Multi-grey level images, 707, 715, 716  
Multi-language, 714  
Multilayer perceptron, 769  
Multilevel verification systems, 929  
Multilingual  
    content, 836  
    documents, 838  
Multimodal, 606, 607, 617–618, 639  
Multimodal biometrics, 933, 941  
Multipage documents, 192  
Multiple-frame processing, 853–855  
Multi-recognition engine strategies, 740  
Multi-resolution  
    morphology, 153  
    representations, 494, 500–501  
Multi-resolution analysis (MRA), 535  
Multi-scale, 717  
    decomposition, 534–535  
    layout analysis, 717  
Multi-script  
    document, 294, 305–307, 326  
OCR, 306–308  
Multi-scriptor, 711  
Multi-touch gesture, 958  
Music interpretation, 998  
Music symbols, 750, 752–754, 756, 757,  
    759–765, 767, 769, 770
- N**
- Named entities, 337  
Naskh, 437–439, 447, 449, 452  
Nastaliq, 437–439, 442, 444, 447, 449  
Natural images, 622, 627, 640  
Natural language processing (NLP), 206, 448  
Navigation, 847  
*n*-best list, 411  
Nearest neighbors, 379, 471–472, 542  
Nearest neighbors classifiers, 471–472  
Near-vertical strokes, 119–120  
Negative Rate Metric (NRM), 1020  
Nepali, 301  
Neural nets, 718, 725, 737, 739  
Neural networks, 196, 211, 213–215, 762,  
    768–770

- Neuromuscular system, 918, 919, 921  
 Newspapers, 182, 183, 206, 209–214, 217  
 N-gram language model, 482  
 N-gram models, 474, 483  
 N-grams, 686–688, 690, 697, 828, 909  
 Niblack, 335, 338  
*NLP.* *See* Natural Language Processing  
 Noise, 596, 602, 616–618, 621, 622, 630, 633, 634, 653, 656, 660, 662, 986–988, 990–992, 995, 1002  
 Noise removal, 189–190  
 Noisy background, 95, 97, 99, 101  
 Non-flat characters, 852  
 Non-impact printing, 23, 31–33  
 Non-invasive, 940  
 Non-Manhattan layout, 137, 138, 141, 142  
 Nonterminals, 528  
 Non-textual components, 495  
 Non-threatening process, 940  
 Non-uniform lighting, 860–861, 868  
 Normalization, 277, 283  
 Normalize, 276, 278, 283  
 Notes and rests, 751, 761  
 Numeric fields, 719
- O**  
 Objective assessment, 1032  
 Occlusion, 602, 621, 623, 627, 628, 635  
 OCR. *See* Optical character recognition (OCR)  
 OCR’ed, 808, 829, 832, 836  
 Off-line, 615, 623, 629  
 Offset mathematical expressions, 683, 686–688  
 Offset printing, 19, 24, 29, 30  
 Old scores, 750  
 Old scores, handwritten music scores, 766, 770  
 Omnifont, 65  
 Omni-scriptor applications, 711  
 OMR. *See* Optical music recognition (OMR)  
 On-line, 600  
 Online approaches, 770  
 Online graphics recognition, 952, 962, 963, 975, 976  
 Onlinemusic recognition, 769  
 Ontology, 562, 580  
 ΠΟΔΑ, 208  
 Opaque, 625, 630, 633, 636, 637  
 Open content alliance, 198  
 Opening, closing, 153  
 Operational strokes, 923  
 Operator dominance, 686, 687, 691–693, 695  
 Operator-driven decomposition, 691–693, 695  
 Operator trees, 683–685, 691, 693, 695, 697, 698  
 Optical character recognition (OCR), 7, 9, 10, 64–68, 259, 260, 262, 263, 266, 272–273, 276, 278, 293, 301, 303, 306–308, 315, 323, 325, 332, 333, 335–338, 341, 342, 345, 348–350, 352–355, 524, 657, 663–667, 673, 750, 752, 780, 795, 798, 806–814, 817, 820, 828, 829, 832, 834, 835  
 Optical character recognition (OCR)-based techniques, 836  
 Optical character recognition (OCR) engines, 718  
 Optical music recognition (OMR), 750, 752–759, 767–771  
 Optimization, 865, 870, 872  
 Optophone, 333  
 Origin of language, 293, 296–299, 326  
 Oriya, 304, 316, 320, 461  
 Orthogonal, 374  
 Orthography, 302  
 Otsu, 338  
 Outlier rejection, 720  
 Overlapping  
     layout, 139, 141–142, 145, 147–149, 161, 165–172  
     text lines, 268  
 Over-segmentation, 335, 349
- P**  
 Page  
     analysis, 992, 994–996  
     component, 139, 144–148, 150, 151, 154, 155, 157, 158, 160, 161, 165, 166, 169, 172  
     decomposition, 187  
     hierarchy, 187  
     orientation, 104–112  
 Page segmentation, 135–173, 189–191, 212, 220, 985, 988, 995  
 Page segmentation competition, 1022  
 Pages tree model, 779  
 Pairwise contour matching methods, 506–508  
 Paleography, 302  
 Paleo-Hebrew, 439  
 Palimpsests, 14, 47  
 Palm leaf manuscripts, 476, 477  
 Paper  
     degradation, 766, 768, 771  
     fingerprint, 847  
 Papyrus, 13–14  
 Parallel calculations, 616  
 Parameter features, 924–926

- Parchment, 13–14, 16, 47  
Parsing techniques, 695  
Partial differential equations (PDE), 634, 636  
Partial graphic object recognition,  
    962–963, 976  
Partial missed region, 1024, 1025  
Part-of-speech (POS), 195, 200  
Patents dataset, 218  
Paths, 757–760  
Pattern, 599, 604, 617, 632  
Pattern recognition, 524, 527, 532, 537, 538,  
    540, 543, 545  
Pattern recognition applications, 743, 744  
PAW, 446, 447, 453  
PDE. *See* Partial differential equations (PDE)  
PDFBOX, 781–783, 785, 800  
PDF documents, 776–785, 799, 800  
PDF generation methods, 779  
PDL. *See* Picture Description Language (PDL)  
Peak Signal-to-Noise Ratio (PSNR), 1016,  
    1017, 1019–1020  
Penalty graph minimization, 694  
Pencil, 19, 20, 22, 23, 28  
Pen-computing technology, 326  
Pens, 13, 16, 18–23, 25, 28, 37, 52  
Pen trajectory, 890, 893  
Percentile features, 344–345  
Perceptually important points, 923, 927  
Perceptual organization (PO), 513, 598–601,  
    610, 614, 639  
Performance  
    evaluation, 546, 673, 674, 984–986, 990,  
        993, 997, 1022, 1028, 1032  
    measures, 216–217  
    metrics, 696  
Peripheral features, 468  
Persian, 314, 319, 320  
Personal entropy, 924  
Personalized threshold, 937  
Perspective distortion, 850, 853, 857–860,  
    873, 875  
Perturbation(s), 716, 717, 719, 739, 740  
Perturbation(s) techniques, 717  
Phoenician alphabet, 428, 432  
Phonetics, 299  
Photocopier, 32, 49  
Photocopying, 16, 33, 49, 55  
Phrase recognition, 382  
Picture description language (PDL), 526, 537  
Pixel based detection, 615–616  
Pixels, 36, 38–40, 44, 50, 51, 53–56  
Pixel-to-contour distances, 1021  
Planographic printing, 28–30  
Platform-and application-independent, 777  
Point Voronoi diagram, 156, 157  
Polar, 529–531  
Polar coordinates, 530, 533  
Polygenetic theory, 298  
Polygonal approximation, 494, 504, 508,  
    510, 511  
Polyline primitives, 545  
Poor-quality printing, 718  
Portable documents, 777, 779  
Postal applications, 705–745  
Postal automation, 67  
Postal items, 707, 714, 716, 720  
Postal addresses, 362, 363, 366  
Postal addresses recognition, 709, 710, 714,  
    720, 721, 732  
Post processing, 463, 465, 473–474, 479,  
    481–484  
Practical, 581–583  
Precision, 201, 202, 206, 210, 211, 215–217,  
    826, 838, 1017, 1019, 1024, 1027  
Preclassification, 909  
Preprinted texts, 731  
Preprocessing, 711, 715, 716, 755, 756, 767,  
    769, 950–953, 959, 960, 975  
Pre-processing, 461, 466, 473, 475–477, 598,  
    615, 619, 629  
Preregistered template, 717  
Prevention of fraud, 744  
Price label scanners, 335  
Primitive(s), 137, 144–149, 154, 158, 164, 527,  
    528, 535, 537, 540, 541, 543–546,  
    752–754, 760–762, 764, 765, 767,  
    768, 770  
Primitive(s) extraction, 490  
Primitive shape fitting, 954–955  
Printed  
    addresses, 723  
    document, 807, 814, 817, 832, 834  
Printing, 12, 15–20, 23–37, 44, 48, 49,  
    51, 52, 55, 56  
Prior knowledge, 619  
Privacy issues, 931  
Probabilistic annotation model, 831, 835  
Probabilistic language models, 482  
Probabilistic modeling, 205, 215, 216  
Probability  
    density, 266, 269–271, 287  
    model, 823, 825  
Profiles, 620, 629, 631  
Progressive refinement strategy, 722–725  
Projection(s)  
    distortions, 623  
    histograms, 372–373  
Projection based method, 148–151

- Projection profile(s), 74, 75, 77, 99, 100, 102, 103, 105, 106, 112–116, 120–121, 858, 859  
 Projection profile(s) cutting, 687, 691, 692  
 Projection, region growth, 287  
 Properties of handprinted, 367–369, 377, 382, 384, 385  
 Prototype-based descriptors, 535, 536  
 Pruning, 820  
 Pseudo-polar transform, 530  
 Psycho-visual approach, 213  
 Public datasets, 596, 639  
 Public signature databases, 932  
 Punctuation, 5, 9  
 Pyramidal decomposition, 535, 545
- Q**  
 Quality, 11–60  
 Query  
     expansion, 826  
     by expression, 682, 698  
     image, 819, 821  
 QuickDiagram, 951, 963, 970–971, 976
- R**  
 Radicals, 467, 468, 470, 892, 903, 904, 906  
 Radon transforms, 529, 530, 533  
 RAG. *See* Region adjacency graphs (RAG)  
 Rajasthani, 301  
 Random forgeries, 926, 932, 935, 938  
 Raster-to-vector  
     algorithms, 545  
     conversion, 494, 498, 503–505, 510, 516  
 Ratio, 279, 284  
 Reading machine, 332–334  
 Reading order, 778, 783, 799  
 Real-life  
     applications, 707  
     problems, 744  
 Real-scenes, 596, 604, 623, 627, 638, 639  
 Real-scenes images, 623, 638  
 Real-time  
     computing, 709, 710, 717  
     recognition, 710, 714  
 Real-world  
     applications, 711, 744  
     data, 1032  
 Recall, 201, 202, 206, 210, 211, 215–217, 826, 838, 1017, 1019, 1024  
 Receptive fields, 629, 631  
 Reciprocal rank, 607  
 Recognition  
     errors, 722, 723  
     lattice, 411  
     rates, 708, 712, 730, 734, 735, 741  
     strategy, 718, 721, 722, 724, 727, 733, 741  
     systems, 1013, 1025, 1032, 1033  
 Recognition and retrieval, 698  
 Recognition-based localization, 865–867, 873  
 Recognition driven segmentation, 479  
 Recognition of amounts, 710, 711, 714  
 Recognition of legal amounts, 711  
 Recognition of logos, 710, 711, 744  
 Rectangular layout, 138, 140, 144  
 Recurrent Neural Network (RNN), 821–824  
 Recursive decomposition, 691–693, 695  
 Recursive layout analysis, 692  
 Recursive XY cut, X-cut, Y-cut, model-based method, 150  
 Redundancy, 680  
 Reference signature model, 930  
 Reflow, 138  
 Region adjacency graphs (RAG), 527, 528, 537  
 Region based detection, 616  
 Region growth, 266, 268–269, 273, 287  
 Regions, 528, 529, 533, 535, 537, 544, 545  
 Regression, 529, 533  
 Regular expressions, 738  
 Regularities, 443, 444  
 Rejection scheme, 741  
 Reject rate, 710, 741  
 Rejects, 713, 715, 716, 719, 720, 741  
 Relevance feedback, 596, 607–610, 613, 628, 632, 826  
 Relief printing, 23–28  
 Representativeness, 596, 612, 627, 635  
 Resampling, 894, 895  
 Reservoir base-line, 311  
 Resolution, 32, 34, 36, 38, 39, 44, 46, 48–50, 53, 54, 343  
 Retrieval, 594–615, 617, 622, 623, 625–626, 629, 631, 632, 638–640  
 Retrieval models, 823–826  
 Right pages, 180, 182, 200  
 RLSA. *See* Run-length smearing algorithm (RLSA); Run-length smoothing algorithm (RLSA)  
 Rocchio algorithm, 609  
 ROC curve, 931  
 Roman script, 301, 303, 304, 314  
 Rotation invariant, 146, 161  
 Routing, indexing, or translation, 307  
 Rule base, 194–195, 197, 209, 219  
 Rule-based  
     approaches, 194–195  
     system, 762

- Rules, 750, 753, 754, 758, 760, 763–765, 768, 769  
Rulings, 650, 652–660, 662–665, 667, 669, 670  
Run, 308, 309, 311–315, 318  
Run-based encoding, 509  
Run-length, 149, 151–152, 309, 311, 314, 315, 318, 756–759  
Run-length smearing algorithm (RLSA), 149, 151–153, 213  
Run-length smoothing, 74–76, 106  
Run-length smoothing algorithm (RLSA), 67, 76, 106, 151, 498, 502  
Russian, 314, 319
- S**  
Sakhr, 450, 452  
Sanskrit, 301, 304  
Santhali, 304  
Sauvola, 335, 339  
Sayre dilemma, 446  
Scalability, 596, 612, 615, 621–622, 638, 639  
Scanned bitmap image, 780  
Scanners, 716  
Scanning, 32, 36, 37, 39, 40, 48, 50, 53–56  
Scene  
  images, 846, 849–852, 861, 869, 874, 875  
  texts, 847–849, 851, 852, 854, 862  
SCHOCKEN, 440  
Scoring principles, 739  
Scripts, 291–327, 888–897, 900, 901, 903–906  
Scripts identification, 293, 294, 301–305, 307–318, 320, 323, 325, 326  
Seedpoint, 268, 269, 287  
Segment, 257–288  
Segmentation  
  based, 380–382  
  error, 1025  
Segmentation-by-recognition, 725, 726, 729, 738, 739  
Segmentation-driven OCR, 478  
Segmentation free, 366  
Segmentation free methods, 807, 828  
Segmentation Metric (SM), 1023  
Segmentation of characters, 360, 361, 372, 382  
Segmentation-recognition, 719  
Segmented, 524–526, 531, 543  
Selective averaging, 165  
Selforganizing map (SOM), 817, 818, 824  
Self-related attributes, 194, 195  
Semantics  
  ambiguities, 555, 559  
  analysis, 559  
categories, 598, 604–607, 610  
gap, 547  
Semiformal visual language, 681  
Semi-global based vision classifiers, 446  
Semiotics, 571  
Semi-structured documents, 186  
Semi-transparent, 625, 630, 636, 637  
Sequence of strokes, 919, 920  
Serif, 272–274, 276, 277  
Server-based OCR, 336  
SFSA. *See* Stochastic finite state automaton (SFSA)  
Shadow-through effects, 95, 97, 99  
Shape  
  context, 601, 603, 604, 611, 618  
  descriptors, 600–604, 610  
  recognition, 603  
Shape code, 807, 810–812, 817, 826  
Shape context representation, 837  
Shape directed covers, 149, 158–160  
Shapeme, 603, 611  
Shining-through, 95, 97, 99  
Shirorekha, 316, 475, 478  
Short-term modifications, 30  
Show through, 48–50, 52, 56–57  
SIFT  
  descriptors, 535, 627, 629, 631, 632  
  features, 618  
Sigma-lognormal  
  model, 920–922  
  parameters, 933  
Signal processing, 494, 496, 501, 502  
Signatures  
  comparison, 926, 928  
  complexity of, 929  
  components, 919, 923  
  segmentation, 923, 936, 940  
  repeatability, 930  
  stability, 929  
  verification, 984, 993, 999–1001  
Signature verification competition, 917–942  
Silkscreen, 24, 30, 31  
Similarity, 807, 811, 824, 825, 828, 829, 832, 834–837  
Similarity measure, 538, 539, 545  
Similarity measure method, 837  
Simple forgeries, 932, 935, 938  
Simple operations, 761, 764  
Single word recognition, 393, 398, 405, 411  
Singularities, 443, 444  
Skeleton, 362, 364, 369–371, 384, 525, 529  
Skeleton-basedmethods, 506, 509  
Skeletonization, 74–76  
Sketching interfaces, 949–977

- Skew, 40, 48, 53, 54, 816  
 Skew detection, 190  
 Skilled forgeries, 926, 931, 932, 935, 938  
 Slant  
     correction, 814, 816, 894  
     estimation, 119–123  
 Slanted, 262, 281, 287  
 Slanted line, 316  
 Sliding windows, 525, 545, 625–627  
 Slope, 317, 323, 325  
 Slurs, 751, 753  
 Smart desktop systems, 846  
 Smart FIX, 204  
 Smartphones, 708, 744  
 Smearing based method, 149, 151, 164  
 Smoothing, 267, 893  
 Sobel, 342  
 Soft biometrics, 930  
 Software, 761, 770–771  
 SOM. *See* Selforganizing map (SOM)  
 Sparse-pixel, 494, 506, 508–509  
 Spatial  
     information, 904  
     relations, 891, 892, 896, 902, 903, 906  
     relationships, 681, 685, 690, 691, 693, 695  
     structure, 901–904, 906  
 Spatio-temporal persistency, 630  
 Specialized hardware, 709, 716  
 Spectral  
     estimation, 634  
     saliency, 627  
 Spline  
     curves, 322  
     function, 322  
     knots, 322  
 Spot noise, 621  
 Spotting  
     methods, 524, 544–547  
     system, 543, 544  
 Staff  
     lines, 752, 753, 755–761, 767, 768  
     removal, 753–759, 767, 769, 770  
 Stamp recognition, 708  
 Statistical  
     classifiers, 531, 538, 541–543  
     descriptors, 528, 530, 546  
     features, 468–470, 480  
     method, 668–669  
 Statistical language models, 473, 909  
 Stencil duplicating, 31  
 Stochastic, 693, 697  
 Stochastic context-free grammar, 691, 693, 694  
 Stochastic finite state automaton (SFSA), 482  
 Stochastic language models, 697  
 Street name recognition, 722, 726–729  
 Street number recognition, 720, 722, 728–729  
 Stripe noise, 621  
 Stroke(s), 274–276, 282, 283  
     density features, 468–470  
     filter, 868  
     map, 282  
     order, 891, 892, 901, 902, 904–906, 908  
     relation, 902, 903, 906  
     segmentation, 950, 953, 954, 959, 975  
     shape, 906  
     width, 311, 322  
 Stroke based approach, 468  
 Stroke cavity map (SCM), 282, 283  
 Stroke width transform (SWT), 867, 868  
 Structural  
     descriptors, 525, 528, 530, 534–542, 544, 546  
     element, 153  
     elements of characters, 377  
     features, 343, 362, 365, 377–378, 384  
     or geometric features, 480  
     information, 904  
     method, 655, 668  
     representation, 544  
 Structure components, 784  
 Structured documents, 180, 186, 212  
 Style recognition, 293, 294, 317–322, 325, 326  
 Stylus containing a small CCD camera, 922  
 Subgraph  
     isomorphism, 536, 545  
     matching, 541, 545  
 Subregion, 268, 269, 287  
 Superimposed logos, 623, 625, 630  
 Superpixel, 145, 147, 868  
 Super-resolution, 853–856, 870  
 Supervised learning, 538, 582, 583  
 Support vector machines (SVM), 196, 213, 342, 366, 379, 384, 540, 542, 543, 687, 688, 762, 768, 769, 862, 865, 866  
 SURF keypoints, 627  
 Syllabaries, 891  
 Syllabic, 304  
 Syllabic system, 429  
 Symbols  
     identity, 687, 688, 690, 694  
     location, 684, 690  
     normalization, 531  
     recognition, 680, 683, 685, 688–694, 696, 698, 754, 755, 760, 763, 765, 769, 770, 961–964, 969, 973, 976, 984, 989, 993, 996, 997, 999

- relationships, 690  
spotting, 556, 568, 583, 997, 999
- Synergy, 921
- Syntactic  
constraints, 688, 691, 694  
descriptors, 535  
grouping, 200  
methods, 691  
models, 514
- Syntactical, 361, 362, 377, 378
- Syntactic logical labeling, 195
- Syntactic pattern recognition, 680, 693–694
- Syntax, 571
- Synthetic  
data, 984–988, 990–992, 1015, 1032  
imagery, 1029  
query, 635
- Synthetic individual generation, 933
- Syriac, 428, 430–434, 447, 452  
language, 430, 433  
words, 432, 433
- System interoperability, 934
- Systems, 593, 595–614, 616–618, 621–623,  
627, 629, 632, 638–640
- T**
- Table(s), 67, 647–674  
processing, 1000  
recognition, 984, 989, 993  
spotting, 203
- Tamil, 316, 320, 461, 477
- Technical  
documents, 491, 512, 524, 527,  
544, 546  
journals, 180, 184, 197, 206,  
209–214, 217
- Technology Development for Indian  
Languages (TDIL), 465
- Telugu, 315–317, 319, 461
- Template matching, 285, 286,  
759–761, 764
- Temporal  
clue, 893  
persistency, 633
- Term frequency, 825, 835
- Terminal, 528, 537
- Terminal punctuation, 957, 958
- Text detection, 814
- Text extraction tools, 782
- Text-graphics separation, 491, 493–497,  
500–503, 512, 515–517
- Text image acquisition, 849, 850
- Text line, 810, 814, 820, 822, 824, 837
- Text line orientation detection, 466
- Text localization, 843–875, 999, 1000
- Text/non-text discrimination, 861, 862, 865,  
867, 869, 870, 872
- TextNon-text classification, 190
- Text recognition, 848, 849, 851–852, 862,  
865, 869–875, 986, 988–990, 992,  
995–998, 1000, 1025–1026, 1033
- Text-to-speech conversion, 303, 333
- Text touching graphics, 491, 493, 498,  
501–504
- Texture analysis, 166
- Texture-Based Graphical Primitives, 494,  
513–514
- Texture features, 313–315, 317, 319,  
320, 325
- Thaana, 428, 430, 434
- Thai, 312, 314, 318, 321
- Thermal printers, 30
- Thinning, 75, 370
- Threshold(ing)  
method, 368  
reduction, 153
- Time delay neural network, 910
- Time series, 889, 896
- Times new roman, 322, 323
- Tobacco-800, 620, 640, 641
- TOCDAS, 200
- TOC page, 198–202
- Tokenization, 200
- Token passing model, 407
- Top-down  
architecture, 765  
recognition strategies, 718  
verification strategies, 928
- Topology, 601–603, 608, 614
- Top (Bottom) reservoir, 310
- Total variation regularization, 96, 99
- Touching characters, 260, 263, 265–268,  
273–281, 284–288, 371, 372,  
479, 725
- Touch screen, 923
- Trademark  
descriptors, 596, 598, 604, 613  
registrations, 593–595, 604, 605, 639  
watch, 594, 595, 597, 614
- Trademark logos, 530
- Trademark retrieval, 617, 623, 626
- Trademark retrieval systems, 595–614, 622,  
638–640
- Trainable sequential maximum  
a posteriori, 170
- Training-free, 615, 618
- Transducer lexicon, 200

- Translator, 846  
 Transliteration, 303  
 Transparent, 625, 630, 632, 636, 637  
 Tree-based classifiers, 480  
 Tree-based document models, 187  
 Tree classifier, 316  
 Tree organizations of lexicons, 726  
 Tree-organized lexicons, 727  
 Tri-lingual (triplet) documents, 315  
 TV broadcasts, 622, 639  
 TV Logo detection, 622–625, 630, 633, 635, 636, 640  
 Typefaces, 6, 7, 65, 274, 276–278, 281, 283, 287, 288, 303, 317, 322, 323, 325  
 Type I error rate, 931  
 Type II error rate, 931  
 Typesetting, 778  
 Typesetting conventions, 194, 197  
 Typewriters, 19, 24, 26–28, 31, 52  
 Typewritten addresses, 709  
 Typographic signs, 317
- U**  
 UML diagramming, 957, 958  
 Under-segmentation, 691, 694  
 UNLV database, 207, 208  
 UPV-BHMM, 450  
 UPV-PRHLTREC1, 451  
 UPV-PRHLT-REC2, 450  
 UPV-BHMM, 450  
 Urdu styles, 439  
 User interface, 613, 630, 698  
 User interface design, 966, 971  
 US National Library of Medicine, 209  
 UW datasets, 217
- V**  
 Vectorial representations, 498–500, 505  
 Vectorization, 490–494, 503–506, 511, 515, 516  
 Vector space (VS), 823, 825, 835  
 Vellum, 14, 17, 47  
 Vendors, 707, 708, 710, 731, 742–744  
 Vendors of postal recognition software and systems, 732–733  
 Verification, 284, 285  
 Verification process, 919, 926, 930, 931, 937, 941  
 Vertical line, 316  
 Vertical projection histograms, 814  
 VHTender, 214
- Video, 44, 46  
 Video-based text documents, 326  
 Videos  
   coding, 707, 708, 732, 733  
   inpainting, 633–636  
   retrieval, 848, 849  
   sequences, 596, 622, 623, 628, 629, 635, 637  
 Vienna classification, 597, 603, 605–607, 613  
 Visible watermark, 623  
 Visual  
   saliency, 864  
   similarity, 595–598, 601–605, 608, 609, 613, 617  
   vocabularies, 717  
 Vocalization, 435  
 Voronoi  
   diagram, 146, 147, 149, 156, 157, 161, 163, 339  
   edge, 149, 156, 157, 161  
   region, 156, 157  
 Voting  
   scheme, 626, 627  
   strategies, 545, 547  
   technologies, 711  
 Vowel modifiers, 462, 463, 475, 477, 478, 481  
 Vowels, 428, 430–432, 435, 436, 440, 462, 463, 478
- W**  
 Wabot-2 robot, 759, 760  
 Water flow level, 311, 312  
 Watermarks, 15, 47  
 Water reservoir, 310–312, 315, 316  
 Water reservoir area, 311  
 Wavelet  
   co-occurrence signatures, 314  
   decomposition, 113–115  
   energy features, 314  
   log mean deviation features, 314  
   packet, 167, 169, 170  
   scale co-occurrence signatures, 314  
   transform, Radon transform, 529  
 Weak classifier, 543  
 Web-based OCR, 336  
 Weighting scheme, 825  
 Western-style signatures, 930  
 Whiteboard writing recognition, 837  
 White tile(s), 146, 147, 149, 160–162  
 Width normalization, 398  
 Wiener filter, 90, 92, 95  
 Wigner-ville distribution, 120, 121

- Wild dots, 893  
Windowing approaches, 616  
Wired-glove device, 922  
WISDOM++, 212  
Word error rate (WER), 1026–1027  
Word features, 809, 829, 838  
Word image  
  coding, 811  
  descriptors, 819  
Word insertion penalty, 411  
Word recognition, 359–385, 445–448, 453,  
  465, 474, 480, 482, 707, 718–719,  
  725, 727, 731, 740, 741  
Word recognition rate (WRR), 1026–1027  
Word segmentation, 442–444, 446, 447,  
  466, 478  
Word shape coding, 810  
Word spotting, 805–838  
Writer-dependent  
  approaches, 929  
  threshold, 934  
Writer identification, 996, 998, 999  
Writer-independent approach, 929  
Writing  
  media, 5, 8–9  
  styles, 8, 9, 905, 906, 911  
  systems, 3–10, 429–441, 453  
  zones, 397–399
- X**  
Xerography, 20, 32  
X-line, 810–813  
XML, 987, 988, 992, 995  
XPDF, 781–783  
X-Y cut, 180, 209  
X-Y cut algorithm, 180  
Xylography, 23, 24  
X-Y tree, 67, 187, 195
- Y**  
Yiddish, Ladino, 439
- Z**  
Zernike, 374, 375  
Zernike moments, 526, 530, 533, 601, 610, 611  
Zernike polynomial, 533  
ZIP codes, 707, 709, 714, 719, 720, 722–726,  
  728, 730  
Zone classification, 995  
Zone segmentation, 984, 986, 988, 992, 994,  
  995, 997, 1000  
Zones of interest(ZOIs), 819, 824  
Zoning  
  descriptors, 533  
  vector, 817, 824