A Document Page Layout Analysis Technique:

Enhanced Scientific Document Understanding Through the Combination of Salient Features into a Support Vector Machine Trained for Equation Detection

in Partial Fulfillment of the Requirements for the Degree
M.S. in Computer Engineering

Presented to the Faculty of ECE
of Virginia Tech by

Jake Bruce

in July 2013

Supervisor: Prof. Lynn Abbott, Ph.D.
Co-Supervisors: Prof. Jason Xuan, Ph.D. and Prof. Jules White, Ph.D.

Introductory remarks

Version <number>, published in <month> <year>

The importance of the glossary. It is strongly recommended to read the glossary in full before starting with the first chapter.

Hints for screen use. This work is optimized for both screen and paper use. It is recommended to use the digital version where applicable. It is a file in Portable Document Format (PDF) with hyperlinks for convenient navigation. All hyperlinks are marked with link flags (►). Hyperlinks in diagrams might be marked with colored borders instead.

Navigation aid for bibliographic references. Bibliographic references to works which are publicly available as PDF files mention the logical page number and an offset (if non-zero) to calculate the physical page number. For example, to look up [Example :a01, p. 100_{-80}] jump to physical page 20 in your PDF viewer.

Declaration

Hereby I declare that I wrote this thesis myself with the help of no more than the mentioned literature and auxiliary means.

Up to now, this thesis was not published or presented to another examinations office in the same or similar shape.

<place>, <date>

place and date

signature (<first name> <last name>)

Abstract

<first paragraph title>. <The abstract of the diploma thesis is meta information resp. “management information”. Therefore it should cover at most two pages, sum up the thesis' essentials and contain the idea behind the thesis.>

Acknowledgments

<Here is one page to thank everybody who helped and supported the author to write this thesis.>

Table of contents

1 Introduction

Basically, our goal is to organize the world's information and to make it universally accessible and useful.

Larry Page – Co-founder of Google

1.1 Enhancing Information Accessibility through Document Analysis and Recognition

Never, since the invention of the printing press, has society seen such a radical change in its means of information distribution. Armed with powerful search engines roaming the vast expanse of the World Wide Web, nearly everyone in the world has, at their very fingertips, access to archives full of information. This enhanced information accessibility is having profound implications for society and could lead to a fruitful age of enlightenment.

The global effects of high speed Internet access are seen daily as hundreds of millions browse for information/multimedia, look up map directions, interact through email/social networks/video games, shop remotely, video chat, etc. Corporations like Google, Microsoft, Facebook, eBay, and Amazon continue building and extending the capacity of their server farms as the growth of user demand shows no signs of slowing down. By mid-2012, it was reported that nearly an eighth of the world's population was on the popular social networking site, Facebook [1]. As such figures continue to grow, studies are showing that technology is even affecting the manner in which we think and behave at the most fundamental levels. Whether or not the long-term effects of this relatively nascent medium of interaction prove to be largely positive or negative remains to be seen. One remaining certainty, however, is that continuing innovation is, for better or for worse, altering the manner in which we live out our daily lives.

It was Benjamin Franklin who once said that “genius without education is like silver in the mine.” One would be hard-pressed in arguing that, throughout history, all people have been able to realize their full potential to succeed and make a difference in the world. If that were true, many would argue that our knowledge would, by now, have long since surpassed its current state. In fact it was just under five hundred years ago, that Europeans were finally emerging from an age of intellectual darkness which had lasted for roughly a millennium. If we look back to the spread of knowledge throughout written history, starting from the earliest true writing systems developed in

ancient Egypt/Mesopotamia circa 3000 BC to the origins of philosophy, math, science, and theater in ancient Greece, all the way to the birth of the “modern era” which culminated itself in the scientific revolution of the sixteenth century AD, we notice a general trend of small bursts of knowledge spreading repeatedly, each time with greater strength than before, each one improving upon on its predecessor. Sir Isaac Newton exemplified this trait of humanity with his statement that “if I have seen further, it is by standing on the shoulders of giants.”

Although much of what defines us from a cultural perspective may indeed be passed from generation to generation through word of mouth, our tremendous advancements in math, science, art, and literature since the dawn of the modern era can be largely attributed to Johannes Gutenberg's invention of the printing press, which made mass distribution of books possible in Late-Medieval Europe. Prior to this key event in history, the stage was set in Europe for an age of scientific inquiry and revelation when the religious leader, Thomas Aquinas, embraced the separation between the purely theological and purely scientific schools of thought. Also of vital importance was the translation and recurrence of ancient Greek writings which had been studied and further developed by Arabic scholars. The first universities built in Medieval Europe were initially centered around classical Greek and religious studies and helped to lead Europe out of its age of darkness. This collaborative environment of scholastic endeavor helped set the framework for an age of enlightenment which would move humanity a step forward. Archaic ideas such as bloodletting were soon supplanted by discoveries leading to modern medicine and the commonly held geocentric model of our earth was replaced by a heliocentric one. Major breakthroughs were made in every field to foster the spread of knowledge which took society to where it is today. Without this ideal of scientific thinking combined with the means to distribute information, society would have never seen such tremendous improvements.

Moving forward to the present day, society has recently made technological breakthroughs which make the world's knowledge and information more accessible than ever before. In fact, many have suggested that the widely used search engine, Google, will go down in history as rivaling in importance with Gutenberg's printing press. It was only about a decade and a half ago that two Stanford Ph.D. students decided that they would like to take a shot at downloading and categorizing the entire internet. These two graduate students are of course the founders of Google [2], a now successful multinational corporation which, during the late nineties, left its search

engine competitors far behind. Google is unique in that its employees facilitate a diverse range of interesting projects ranging from cataloging the human genome, building autonomous vehicles, developing smart homes of the future, to developing augmented reality eye glasses, among many others. It is, however, in Google's core mission of finding ways to make the world's information “more universally accessible and useful,” that the company has had its greatest impact on the world at large. It was in keeping with this mission that, in 2005, in collaboration with HP Labs and the Information Science and Research Institute at UNLV, Google revived and open sourced an optical character recognition engine that had been developed as a Ph.D. project for HP Labs between 1985 to 1995. Although optical character recognition (OCR), the autonomous conversion of printed documents into digital formats, is a very mature area of research [3], development in this area continues in order to increase recognition support for the broad spectrum of languages, formats, and subject matter of printed documents. HP's OCR engine, named “Tesseract,” had proven itself as one of the industry's leading engines during UNLV's Fourth Annual Test of OCR Accuracy [4]. Eventually, however, HP subsequently went out of the OCR business, leaving the software to basically collect dust for about a decade.

Meanwhile, by around 2004, Google had begun its Google Books Initiative [5], a large-scale library digitization project. This initiative began with the lofty goal of digitizing all of the world's printed documents such that they may be indexed and searched online. By around 2005, Google hired Ray Smith, the former lead developer of Tesseract, to return to his long-abandoned, yet ground-breaking, Ph.D. work and also brought Tesseract into the open source domain. In so doing, Google helped to spur further research interest into efficient and accurate document recognition[1]. In the roughly eight years since the project was revived, support has been added for recognition of over fifty languages. Advanced page layout analysis techniques have been implemented in order to detect various types of documents ranging from novels, magazines, newspapers, images, textbooks, sheet music, etc. Language and script detection modules have also been implemented in order to autonomously determine what processing should be carried out for any given world document [6]. If Google's endeavor is successful, then the resulting implications to society will be extraordinary, possibly similar to the impact that Arabic scholars had on Europe when sharing and translating ancient Greek literature. If Google is successful in the autonomous

[1] The term, recognition, is herein used to describe a machine's extraction of a document's contents. This requires both the document page layout analysis as well as algorithms which subsequently convert the page layout contents into a machine-understandable form. The field of document layout analysis is further discussed in Section 1.2.

digitization and recognition of any printed document regardless of its origin, then it will not be long before information from all of the world's documents become instantly accessible in every language and to everyone around the world. Such a development would certainly speed up the world's already significant progress toward an era of far greater enlightenment and wisdom than has yet been seen.

The autonomous recognition of all printed documents would not only expedite the global advancement of knowledge and wisdom, but would also have tremendous implications toward every individual in society. Such a breakthrough would be especially significant toward the endeavor of Assistive Technology. With many devices being developed and studies being carried out on ways to enhance human computer interaction (HCI) for visually or physically handicapped individuals, digital access to all printed documents could make finding information, not only more convenient, but also possible for many who would not otherwise have access. Global autonomous document recognition could also help open the doors toward breaking down language barriers in information accessibility.

As research and development continues to enhance the accurate translation of discourse between various languages [7], the successful recognition of printed documents could eventually allow them to be machine-translated according to the language preference of a given user. With instant access to all of the world's information, regardless of its language or origin, at one's disposal, collaboration and learning among individuals across the world will be significantly enhanced. All people in the world regardless of their language preference, geographical location, and physical ability will have access to the world's stores of knowledge, and the opportunity to have a profound impact on society through the medium of the World Wide Web. Enhanced document analysis and recognition capabilities will make a significant contribution toward this end. The following section will discuss the background as well as some of the fundamental problems faced in the fields of document analysis and recognition.

1.2 Introduction to OCR and Document Analysis: A Brief History

From Herbert Shantz's *History of Optical Character Recognition (OCR)* [3], it is clear that the OCR of printed documents has been studied extensively over the last century. In one of the earliest OCR patents [8] (Figure 1), a mechanical apparatus was used to measure the incidence of light reflected back from a printed character when illuminated through a set of character templates. A character detection would occur when the light emitted from the template overlapped the character (assumed to be in dark print) sufficiently to prevent light from being reflected upon the medium. Despite

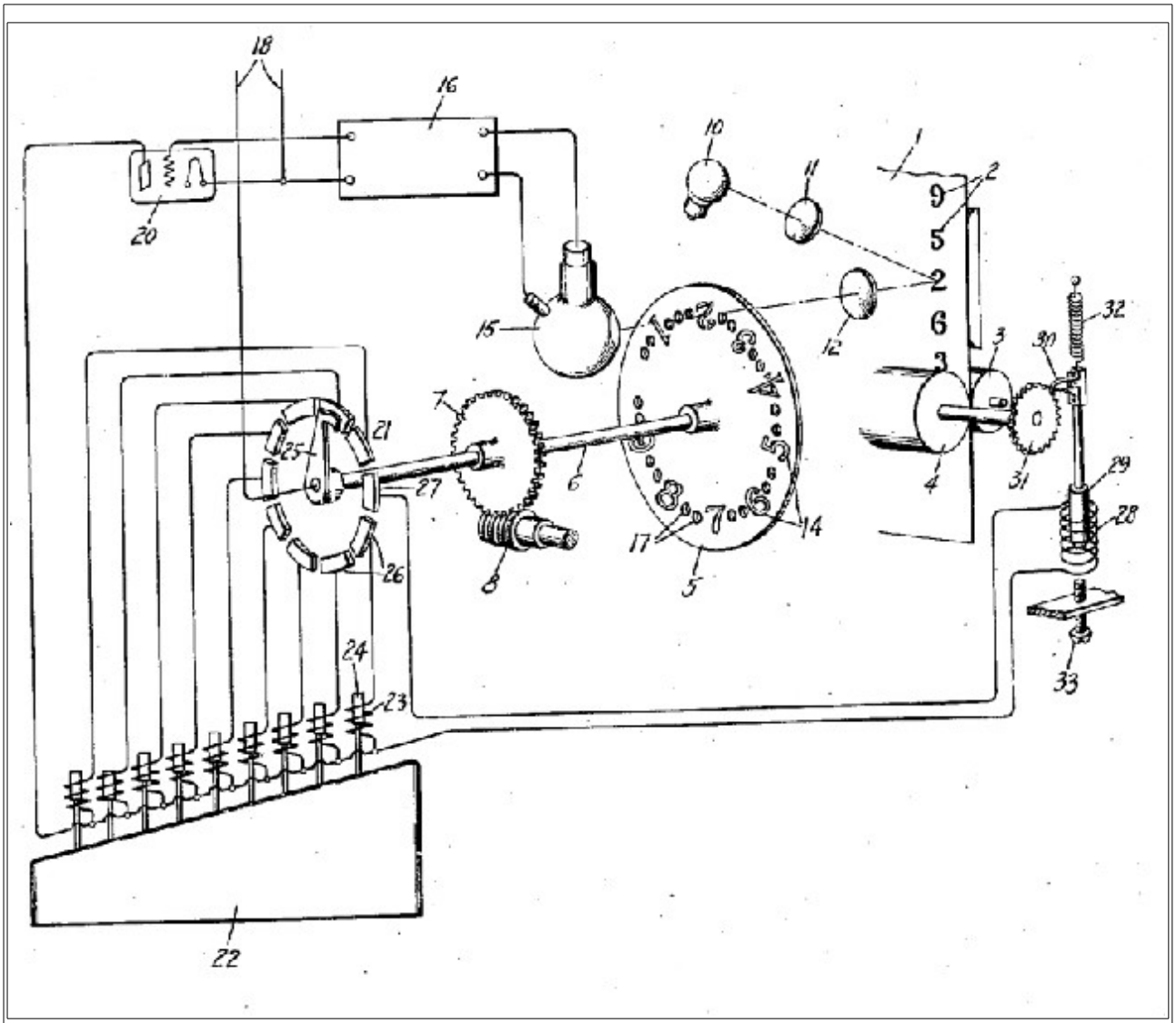


Figure 1: An illustration taken from the 1933 patent from Paul W. Handel, a former employee of General Electric, entitled “Statistical Machine” [8]. This is one of the earliest OCR devices ever invented.

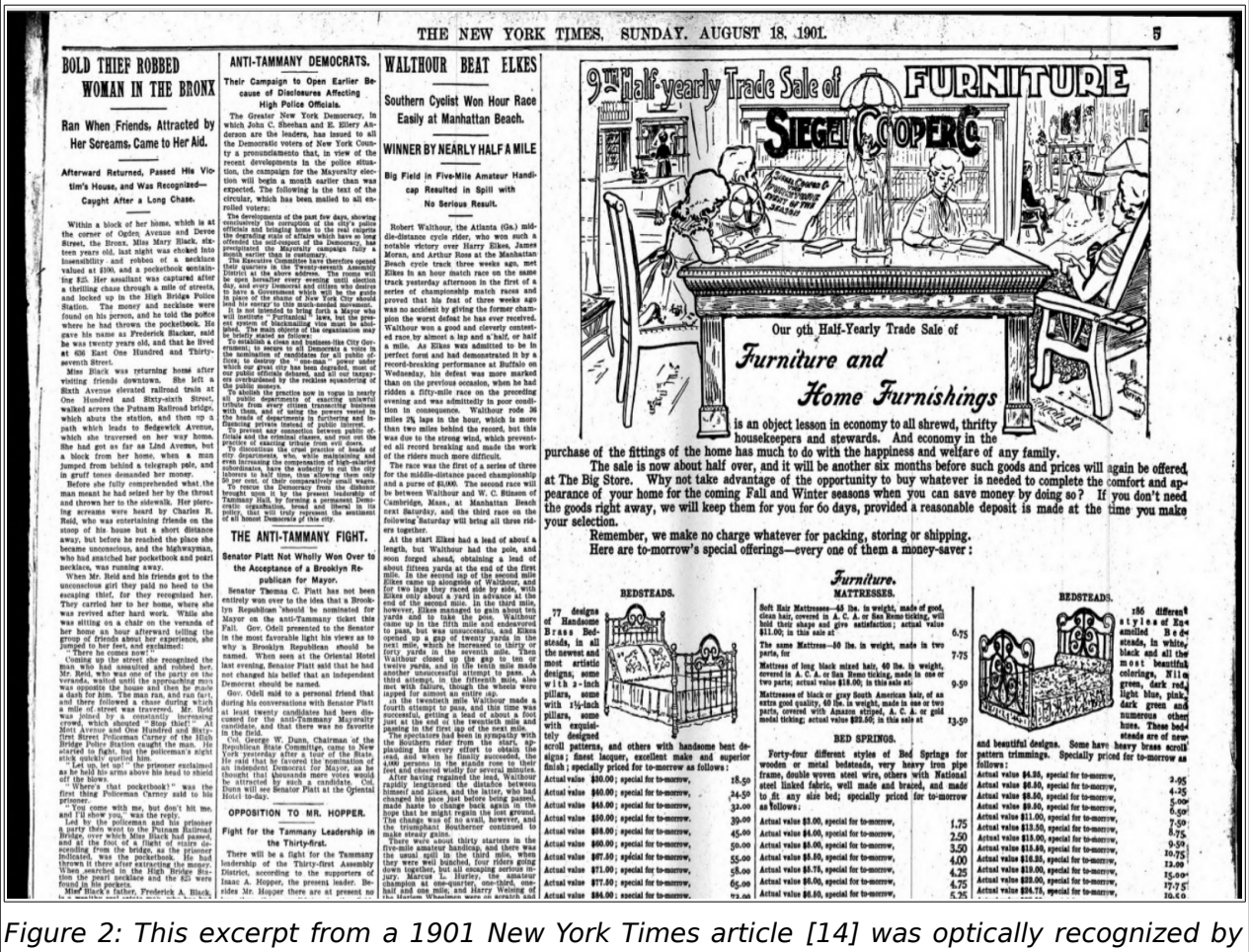
requiring a significant amount of human intervention to ensure proper alignment and being largely inefficient at best, the fundamental ideas which motivated this early initiative are seen repeatedly throughout the century, and even now, albeit on a much larger scale.

Although some of the first commercial OCR systems were released during the 1950's, their applicability was limited in that, by and large, they were only capable of handling a single font type with very strict rules on character spacing. It was not until the mid-late 1970's, with the invention of both the charge-coupled device (CCD) flatbed scanner and the “Kurzweil Reading Machine” [9] that it became possible for a computer to read a variety of documents with reasonable accuracy. Although the training process for a particular font would take several hours and multi-column page layouts or images had to be specified by the user manually, Kurzweil's software showed significant improvement over the state-of-the-art technologies of the time.

In the 1980's, a company called Calera Recognition Systems [10] introduced an omnifont system that could read pages containing a mixture of fonts while also locating pictures and columns of text without any user intervention or extra training. The progress of the state-of-the-art in document recognition will be further discussed in the Chapter 2 Literature Review. More recent commercial OCR systems such as ABBYY FineReader [11], OmniPage Professional [12], and Readiris [13], are all quite accurate, not only in recognizing individual words or characters, but also in understanding and reproducing document layout structure. A magazine or newspaper page may, for instance, contain an intricate heading structure followed by multiple columns of text, pull-out quotes, in-set images, and/or graphs as demonstrated in the historical New York Times article shown in Figure 2 [14].

In order to understand and recognize content of such a document, it is essential to first carry out document layout analysis techniques which will determine how the document is partitioned. The text will be recognized with an understanding of where the columns of text are, which portions of text indicate headings or quotes, and which segments correspond to images, tables, captions, etc. If the text is not partitioned appropriately prior to recognition then the textual output will become unpredictable. With columns, paragraphs, or other structures merged together incorrectly, the text will lose much of its intended meaning and become far less readable to the human eye. For these reasons, sophisticated page layout analysis algorithms are of the utmost importance, not only for document recognition accuracy, but also in ensuring that the generated output is formatted correctly.

1 Introduction



THE NEW YORK TIMES, SUNDAY, AUGUST 18, 1901. 5

BOLD THIEF ROBBED WOMAN IN THE BRONX

Ran When Friends, Attracted by Her Screams, Came to Her Aid.

Afterward Returned, Passed His Victim's House, and Was Recognized—Caught After a Long Chase.

Within a block of her home, which is at the corner of Ogden Avenue and Devoe Street, the Bronx, Miss Mary Black, sixteen years old, last night was choked into insensibility and robbed of a necklace valued at $100, and a pocketbook containing $25. Her assailant was captured after a thrilling chase through a mile of streets, and locked up in the High Bridge Police Station. The money and necklace were found on his person, and he told the police where he had thrown the pocketbook. He gave his name as Frederick Slacker, and he was twenty years old, and that he lived at 626 East One Hundred and Thirty-seventh Street.

Miss Black was returning home after visiting friends downtown. She left a Sixth Avenue elevated railroad train at One Hundred and Sixty-sixth Street, walked across the Putnam Railroad bridge, which abuts the station, and then up a path which leads to Sedgewick Avenue, which she traversed on her way home. She had gone as far as Lind Avenue, but a block from her home, when a man jumped from behind a telegraph pole, and in gruff tones demanded her money.

Before she fully comprehended what the man meant he had seized her by the throat and thrown her to the sidewalk. Her piercing screams were heard by Charles B. Reid, who was entertaining friends on the stoop of his house but a short distance away, but before he reached the place she became unconscious, and the highwayman, who had snatched her pocketbook and pearl necklace, was running away.

When Mr. Reid and his friends got to the unconscious girl they paid no heed to the escaping thief, for they recognized her. They carried her to her home, where she was revived after hard work. While she was sitting on a chair on the veranda of her home an hour afterward telling the group of friends about her experience, she jumped to her feet, and exclaimed:

"There he comes now!"

Coming up the street she recognized the man who had assaulted and robbed her, Mr. Reid, who was one of the party on the veranda, waited until the approaching man was opposite the house and then he made a dash for him. The man ran, and ran fast, and there followed a chase during which a mile of street was traversed. Mr. Reid was joined by a constantly increasing crowd, which shouted "Stop thief!" At Mott Avenue and One Hundred and Sixty-first Street Policeman Carney of the High Bridge Police Station caught the man. He started to fight, but the policeman's night stick quickly quelled him.

"Let up, let up!" the prisoner exclaimed as he held his arms above his head to shield off the blows.

"Where's that pocketbook?" was the first thing Policeman Carney said to his prisoner.

"You come with me, but don't hit me, and I'll show you," was the reply.

Led by the policeman and his prisoner a party then went to the Putnam Railroad Bridge, over which Miss Black had passed, and at the foot of a flight of stairs descending from the bridge, as the prisoner indicated, was the pocketbook. He had thrown it there after extracting the money. When searched in the High Bridge Station the pearl necklace and the $25 were found in his pockets.

Miss Black's father, Frederick A. Black,

ANTI-TAMMANY DEMOCRATS.

Their Campaign to Open Earlier Because of Disclosures Affecting High Police Officials.

The Greater New York Democracy, in which John C. Sheehan and E. Ellery Anderson are the leaders, has issued to all the Democratic voters of New York County a pronunciamento that, in view of the recent developments in the police situation, the campaign for the Mayoralty election will begin a month earlier than was expected. The following is the text of the circular, which has been mailed to all enrolled voters:

The developments of the past few days, showing conclusively the corruption of the city's police officials and bringing home to the real culprits the degrading state of affairs which have so long offended the self-respect of the Democracy, has precipitated the Mayoralty campaign fully a month earlier than was expected.

The Executive Committee has therefore opened their quarters in the Twenty-seventh Assembly District at the above address. The rooms will be open hereafter every evening until election day, and every Democrat and citizen who desires to have a Government which will be the gainsay in place of the shame of New York City should lend his energy to this much-needed movement.

It is not intended to bring forth a Mayor who will enforce Sabbatarian laws, but the prevention of Sabbatarianism in vice must be abolished. The main object of the organization may be briefly stated as follows:

To establish a clean and business-like City Government, to secure to all Democrats a voice in the nomination of candidates for all public offices, to destroy the "one-man" power under which our great city has been degraded, most of our public officials debased, and all our taxpayers overburdened by the reckless squandering of the public moneys.

To abolish the practice now in vogue in nearly all public departments of exacting unlawful tribute from every citizen transacting business with them, and of using the powers vested in the heads of departments in furthering and financing private instead of public interest.

To prevent any connection between public officials and the criminal classes, and root out the practice of exacting tribute from evil doers.

To discontinue the cruel practice of heads of city departments, who, while maintaining and even increasing the compensation of high-salaried subordinates, reduce the wages of the poor laborers to half time, thus allowing them only 50 per cent. of their comparatively small wages.

To rescue the Democracy from the dishonor brought upon it by the present leadership of Tammany Hall, by forming a permanent Democratic organization, broad and liberal in its policy, that will truly represent the sentiment of all honest Democrats of this city.

THE ANTI-TAMMANY FIGHT.

Senator Platt Not Wholly Won Over to the Acceptance of a Brooklyn Republican for Mayor.

Senator Thomas C. Platt has not been entirely won over to the idea that a Brooklyn Republican should be nominated for Mayor on the anti-Tammany ticket this Fall. Gov. Odell presented to the Senator in the most favorable light his views as to why a Brooklyn Republican should be named. When seen at the Oriental Hotel last evening, Senator Platt said that he had not changed his belief that an independent Democrat should be named.

Gov. Odell said to a personal friend that during his conversations with Senator Platt at least twenty candidates had been discussed for the anti-Tammany Mayoralty candidate, and that there was no favorite in the field.

Col. George W. Dunn, Chairman of the Republican State Committee, came to New York yesterday after a tour of the State, and said that he favored the nomination of an independent Democrat for Mayor, as he thought that thousands more voters would be attracted by such a candidate. Col. Dunn will see Senator Platt at the Oriental Hotel to-day.

OPPOSITION TO MR. HOPPER.

Fight for the Tammany Leadership in the Thirty-first.

There will be a fight for the Tammany leadership of the Thirty-first Assembly District, according to the supporters of Isaac A. Hopper, the present leader. Besides Mr. Hopper there are at present no

WALTHOUR BEAT ELKES

Southern Cyclist Won Hour Race Easily at Manhattan Beach.

WINNER BY NEARLY HALF A MILE

Big Field in Five-Mile Amateur Handicap Resulted in Spill with No Serious Result.

Robert Walthour, the Atlanta (Ga.) middle-distance cycle rider, who won such a notable victory over Harry Elkes, James Moran, and Arthur Rose at the Manhattan Beach cycle track three weeks ago, met Elkes in an hour match race on the same track yesterday afternoon in the first of a series of championship match races and proved that his feat of three weeks ago was no accident by giving the former champion the worst defeat he has ever received. Walthour won a good and cleverly contested race by almost a lap and a half, or half a mile. As Elkes was admitted to be in perfect form and had demonstrated it by a record-breaking performance at Buffalo on Wednesday, his defeat was more marked than on the previous occasion, when he had ridden a fifty-mile race on the preceding evening and was admittedly in poor condition in consequence. Walthour rode 36 miles 924 laps in the hour, which is more than two miles behind the record, but this was due to the strong wind, which prevented all record-breaking performances at Buffalo on Wednesday.

The race was the first of a series of three for the middle-distance paced championship and a purse of $5,000. The second race will be between Walthour and W. C. Stinson of Cambridge, Mass., at Manhattan Beach next Saturday, and the third race on the following Saturday will bring all three riders together.

At the start Elkes had a lead of about a length, but Walthour had the pole, and soon forged ahead, obtaining a lead of about fifteen yards at the end of the first mile. In the second lap of the second mile Elkes came up alongside of Walthour, and for two laps they raced side by side, with Elkes only about a yard in advance at the end of the second mile. In the third mile, however, Elkes managed to gain about ten yards and to take the pole. Walthour came up in the fifth mile and endeavored to pass, but was unsuccessful, and Elkes opened up a gap of twenty yards in the next mile, which he increased to thirty or forty yards in the seventh mile. Then Walthour closed up the gap to ten or twelve yards, and in the tenth mile made another unsuccessful attempt to pass. A third attempt in the fifteenth mile, also met with failure, though the wheels were lapped for almost an entire mile.

In the twentieth mile Walthour made a fourth attempt to pass, and this time was successful, getting a lead of about a foot just at the end of the twentieth mile and passing in the first lap of the next mile.

The spectators had been in sympathy with the Southern rider from the start, applauding his every effort to obtain the lead, and when he finally succeeded, the 4,000 persons in the stands rose to their feet and cheered wildly for several minutes.

After having regained the lead, Walthour rapidly lengthened the distance between himself and Elkes, and the latter, who had changed his pace just before being passed, made haste to change back again in the hope that he might regain the lost ground. The change of no avail, however, and the triumphant Southerner continued to make steady gains.

There were about thirty starters in the five-mile amateur handicap, and there was the usual spill in the third mile, when about twenty were well bunched, four riders going down together, but all escaping serious injury. Marcus L. Hurley, the amateur champion at one-quarter, one-third, one-half and one mile, and Harry Wojcieszyn of the Morton Wheelmen were on scratch and

Figure 2: This excerpt from a 1901 New York Times article [14] was optically recognized by ABBY Fine Reader 8. The "New York Times" heading at the very top, the "Furniture and Home Furnishings" label embedded in the illustration, and the layout of the three columns at the bottom right were all incorrectly recognized by the commercial system.

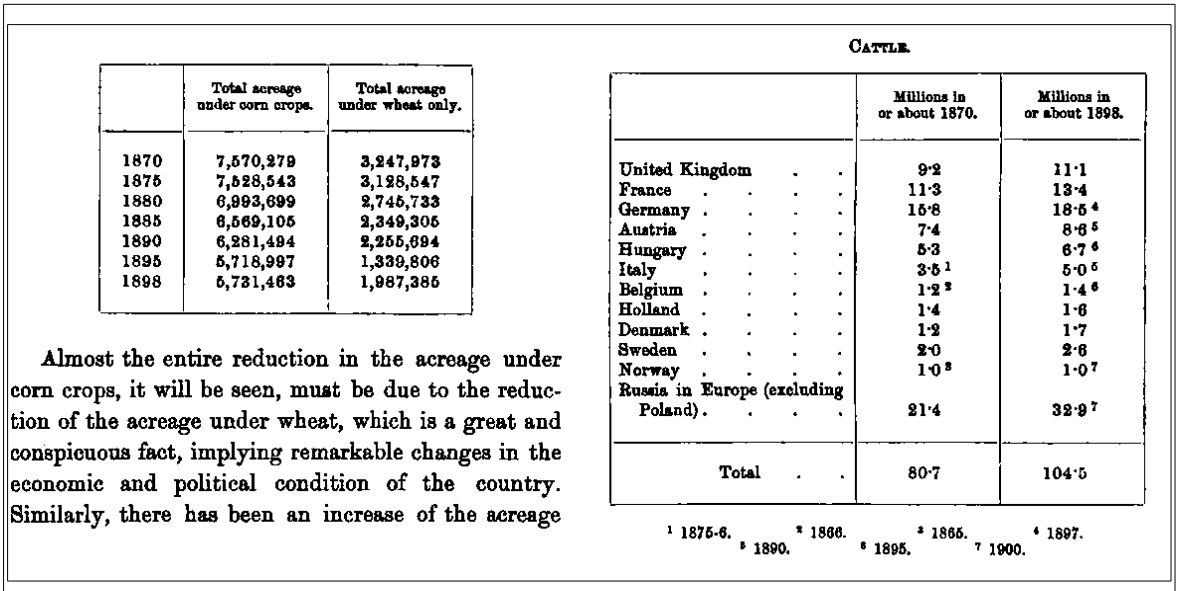
Although most publishers keep digital copies of their more recent documents, there is also great demand for older documents which, unless they are digitized, will largely become forgotten by society. This would be unfortunate in that it is often surprising how pertinent older information and ideas can be. For companies such as Google who would like to make the world's information more readily available and accessible as well as to the Assistive Technology community, this is of the utmost importance. For this reason, a standard OCR output format called hOCR, which embeds OCR output within well-defined and widely available HTML and CSS structures has been put into place [15]. In order to ensure the quality of textual output generated by OCR for the wide variety of possible document layout structures, sophisticated document layout techniques are critical.

1.3 Google Books Initiative

There are various languages, dialects, and page layout formats for which Google's Tesseract software is being developed. Among them are mathematical equations, tables, graphs, and other figures which can be found in any standard science or math text book. While Smith's original work was optimized solely for the recognition of English newspaper formats, Google's continued efforts are aimed at recognizing page formats from a much broader scope [5]. Much of Google's ideas regarding document recognition are essentially in their infancy, and have a long way to go before being fully realized. Although an experimental equation detector has been added to the Tesseract software, its results, although showing significant promise, have been tested to have fairly limited accuracy. A table detector implemented by Google has also been tested on some sample images [16] (Figure 3) to show that, it too, could use significant improvement (Figure 4). Notice that, in the left-most table in Figure 4, the software failed to indicate the years as either belonging to the table or the normal text. They were simply disregarded. Also, the software was unable to determine where exactly the table boundaries are (which should be labeled green). In the right-most table, notice that although a better job was done, while the bottom portion of the text consists of footnotes, it is therein incorrectly labeled as part of the table. Also, the second line of all column labels are not recognized as part of the table when they clearly should be.

The problem of efficiently and accurately detecting equations, tables, graphs, and other figures for the broad spectrum of possible document types is certainly no easy one to solve. Although from a human's perspective, this problem may seem trivial, programming a machine to sum up a document with the same accuracy as the human eye proves to be a daunting task, as will be further discussed in the literature review chapter of this paper. As the inventors of Google continue to work toward their dream of creating an online “Library of Alexandria,” there is significant progress to be made before such a large-scale endeavor can be fully realized. The Google Book Search initiative has opened up many avenues for future research in document understanding and recognition, of which, this project is certainly one of the many to come.

1 Introduction



	Total acreage under corn crops.	Total acreage under wheat only.
1870	7,570,279	3,247,973
1875	7,528,543	3,128,547
1880	6,993,699	2,745,733
1885	6,569,105	2,349,305
1890	6,281,494	2,255,694
1895	5,718,997	1,339,806
1898	5,731,463	1,987,385

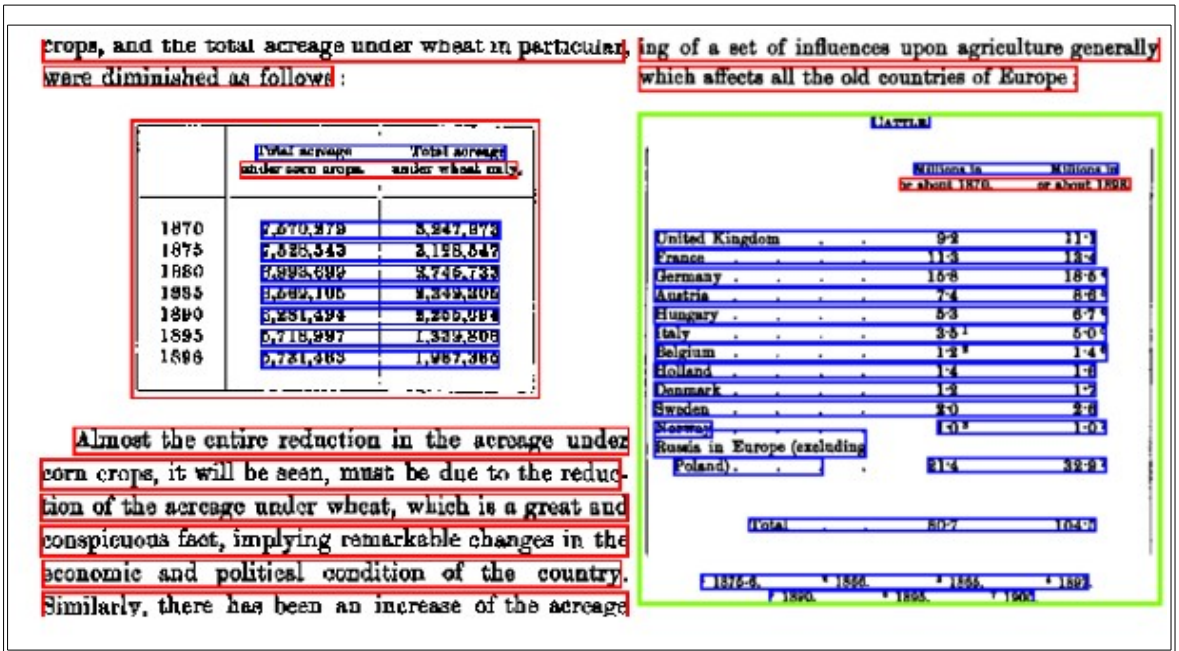
Almost the entire reduction in the acreage under corn crops, it will be seen, must be due to the reduction of the acreage under wheat, which is a great and conspicuous fact, implying remarkable changes in the economic and political condition of the country. Similarly, there has been an increase of the acreage

CATTLE.

	Millions in or about 1870.	Millions in or about 1898.
United Kingdom	9·2	11·1
France	11·3	13·4
Germany	15·8	18·5 [4]
Austria	7·4	8·6 [5]
Hungary	5·3	6·7 [6]
Italy	3·5 [1]	5·0 [5]
Belgium	1·2 [2]	1·4 [6]
Holland	1·4	1·6
Denmark	1·2	1·7
Sweden	2·0	2·6
Norway	1·0 [3]	1·0 [7]
Russia in Europe (excluding Poland)	21·4	32·9 [7]
Total	80·7	104·5

[1] 1875-6. [2] 1866. [3] 1865. [4] 1897. [5] 1890. [6] 1895. [7] 1900.

Figure 3: The above text includes excerpts from two different pages taken from a scan of Sir Robert Griffin's Stastics textbook (circa 1913) [16]. On the left is a table followed by a paragraph of text, while on the right is a larger table. These images were extracted from a PDF which was made available online by Google.



crops, and the total acreage under wheat in particular, were diminished as follows:

Almost the entire reduction in the acreage under corn crops, it will be seen, must be due to the reduction of the acreage under wheat, which is a great and conspicuous fact, implying remarkable changes in the economic and political condition of the country. Similarly, there has been an increase of the acreage

ing of a set of influences upon agriculture generally which affects all the old countries of Europe:

Figure 4: Above is the text from Figure 3 after having been labeled by Tesseract's table detection software. The text within the blue rectangles was identified as belonging to a table while the text within red rectangles was not. The green rectangle should encompass the entire table figure. As can be seen there are both false negatives and false positives.

1.4 Contributions of this Thesis

This thesis introduces a novel approach to detecting mathematical expressions during the document layout analysis stage of OCR. The focus of this thesis is toward enhancing the OCR quality of printed documents which may contain mathematical formulas. The motivation for mathematical expression detection is illustrated by Figure 5. From Figure 5, it is clear that, when presented with mathematical expressions as input, Google's OCR System, Tesseract, will fail. With reliable expression detection, it becomes possible to prevent this mangled output from occurring, and also allows existing equation recognition algorithms, which have been extensively studied in the literature, to be better utilized.

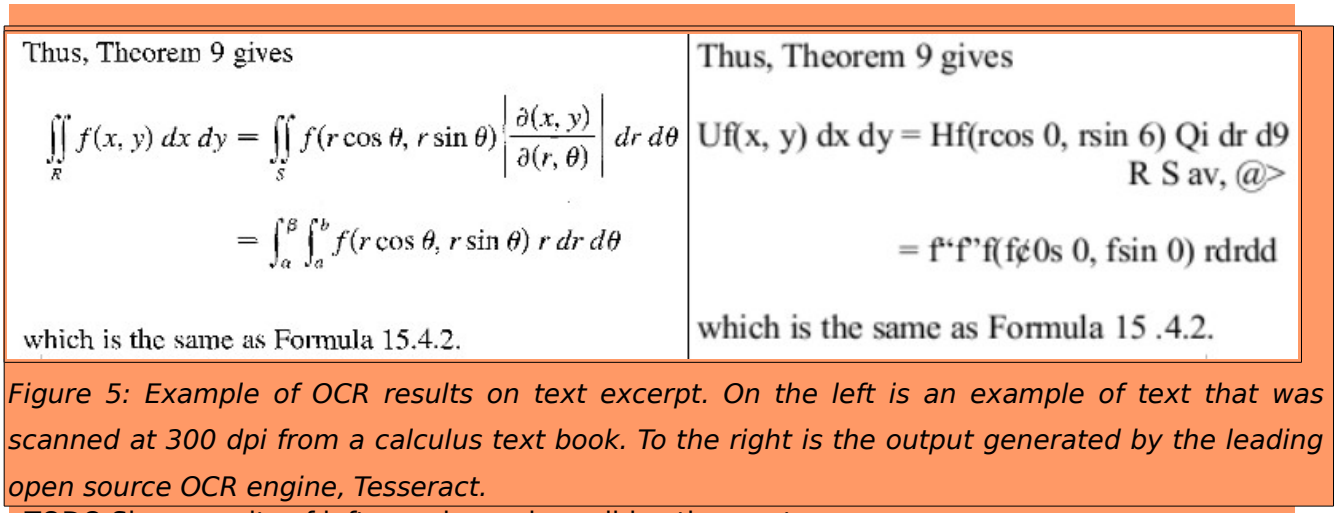


Figure 5: Example of OCR results on text excerpt. On the left is an example of text that was scanned at 300 dpi from a calculus text book. To the right is the output generated by the leading open source OCR engine, Tesseract.

TODO Show results of infty reader and possibly other systems

By utilizing and interfacing with the existing data structures and algorithms present within Google's open source OCR engine, Tesseract, much of the more well-studied areas of OCR / document analysis research are surpassed so that a study of the relevant problem of equation detection can be explored in much greater detail than would be possible otherwise. As the Tesseract software, much like commercial state-of-the-art systems, is capable of partitioning a document into columns, paragraphs, headings, etc., the software implemented in this work searches Tesseract's resulting partitions in order to detect regions of interest. Greater document understanding is accomplished through recognition of a variety of relevant features, many of which have yet to have been explored in existing research. Relevant features are subsequently combined with a binary support vector machine (SVM) classifier.

The feature recognition and classification steps in the proposed system are carried out in two separate passes: the first of which detects areas of interest at the

individual character level while the second uses results from the first to combine the equation regions into their full partitions. The contributions of this work are summarized in the list below. All of the subjects for which the author could not find an in-depth previous study involving equation detection are marked with an asterisk (*).

- (*) Generation of an extensive ground-truth training set of equation test data, taken from scientific text books and articles. These publications are all in the public domain and thus will be freely available online for future research endeavors.
- Recognition of the following combination of features on equation detection. Although many of these features have indeed been studied to some extent, they have yet to all be used within a single framework as of this date to the author's knowledge.
  - (*) Measurement of the number of horizontally adjacent characters to the right of a given character within some vertical threshold (see chapter x.x).
  - Sub-script/super-script recognition
  - (*) Running a Tesseract classifier trained using Infty Reader's database of mathematical expressions and comparing the result to normal language output to detect math expressions (possibly subsequent recognition?) (see chapter x.x)
  - Use of n-grams to locate expressions embedded within text (see chapter x.x)
  - Testing whether or not a character's language[2] classification result falls under a category of potential math symbols such as <,>,_,+,-,/,%, etc.
  - Vertical distance to nearest character neighbor above and below (within a horizontal threshold), horizontal distance to nearest neighbor left and right (within some vertical threshold).
  - Ratio of horizontal and vertical distance from nearest neighbor to the left and above to nearest neighbor to the right and below respectively. ????
  - Height of characters as compared to average character height within a page
  - Detection of Italicized text
  - Horizontal bar detection
  - Features extracted from PDF if available

[2] Here the language classification result indicates the result of a classifier that was trained for a particular language. Although in the context of this work English is all that is tested, testing of existing techniques in various languages is encouraged for future work.

  - Detection of indentation or centering of text
  - Measurement of character vertical distance from a row of text's baseline
- (*) Use of a Support Vector Machine (SVM) classifier for equation detection
- Thorough evaluation of the proposed system's results, includes a comparison with Google's system.

1.5 Organization of Thesis

The work to be discussed in this thesis is aimed toward moving the world a step closer to realizing some of the lofty goals set by Google's engineers and scientists. Chapter 2 presents a review of existing document analysis techniques with extra emphasis on those involving mathematical/scientific documents. Although there are a wide variety of problems which need to be tackled in the area of document recognition, the primary focus is on enhancing equation detection accuracy through the use of feature recognition and a support vector machine (SVM) classifier. The remainder of this thesis is organized as follows: Chapter 3 consists of a theory section discussing the image processing and classification techniques employed as well as software optimization techniques utilized by Google's open source OCR engine, Tesseract. Chapter 4, the method section, discusses the ground truth generation procedure, feature recognition algorithms, classification technique, and result evaluation. Chapter 5, the results section, will involve a discussion of all results and their significance. Chapter 6, the conclusion, summarizes important points and discusses recommendations for future work.

2 Literature Review

“We are like dwarfs sitting on the shoulders of giants. We see more, and things that are more distant, than they did, not because our sight is superior or because we are taller than they, but because they raise us up, and by their great stature add to ours.”

John Salisbury

2.1 The Beginnings of OCR

2.1.1 Fixed-font

Over the past century, Assistive Technology has been a primary incentive for OCR research and development. While machine understanding was initially the most commercially viable domain for OCR, reading devices for the blind have been commonly implemented over the years. In 1914, one of the earliest reading devices, the Optophone [17], could allow blind individuals to understand printed text without relying on braille. The device projected light upon a character of interest, focusing the light's reflection upon a selenium photosensor. A sound with a frequency corresponding to the reflected light would then be emitted to alert the reader of the current character. A blind individual trained to use such a device, however, could only expect to read at a mere one word per minute.

While there were some OCR patents released in subsequent years [18][8], it was not until the late forties and early fifties that there was any commercial development in the OCR industry. In 1949, RCA engineers were working on an OCR system which used an early text-to-speech synthesis technology to read individual characters out loud [19]. This system required the user to move a “eye” (a cathode ray tube) across the letters of interest. The rays were then reflected upon a photosensor connected to a complex processing unit (Figure 5). The project, however, was discontinued prior to completion due to its nonviable commercial-ability.

In 1953, David Shepard patented an OCR system, “Gismo”, which could read all 26 fixed-font letters of the English alphabet, understand musical notation, and comprehend Morse Code [20]. Shepard founded Intelligent Machines Research Corporation (IMRC) and released the world's very first commercial OCR systems. Credit card reading, although now carried out through magnetic strip recognition, was one of the first commercially successful applications of OCR. The Farrington B numeric font,

still widely used on the front of credit cards to this day, was invented by Shepard in order to minimize recognition errors.

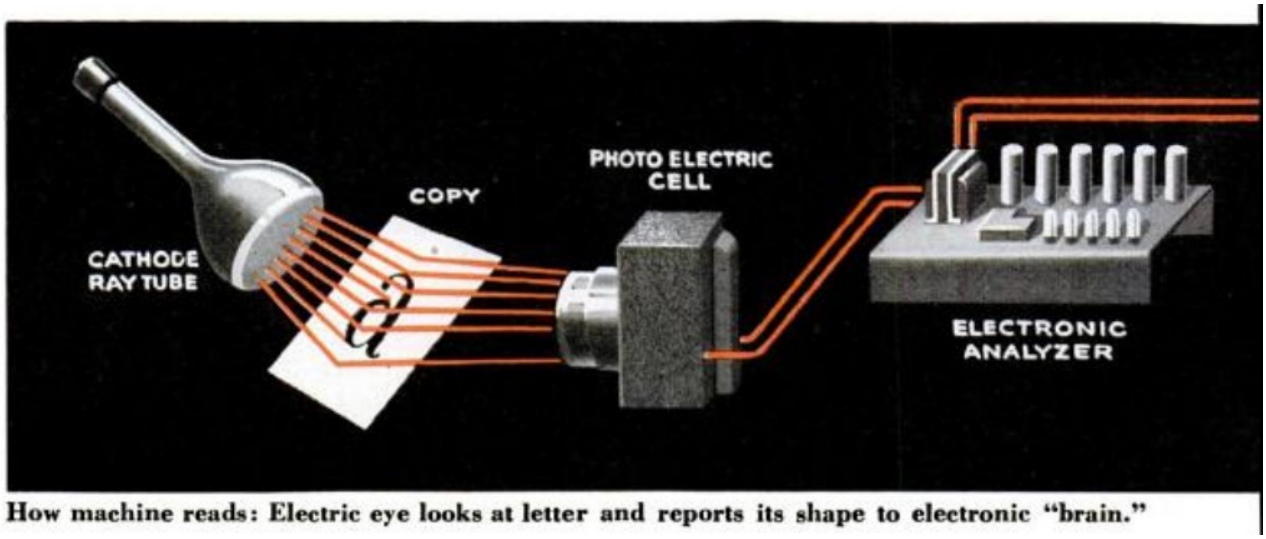


Figure 6: An image of RCA's 1949 OCR system taken from an issue of Popular Science [19]. The system was discontinued prior to completion due to its high costs .

IBM utilized Shepard's patents over subsequent years while also improving upon the accuracy of fixed-font OCR. The IBM 1408 Optical Character Reader [21] was packaged with the IBM 1401 Data Processing System (Figure 6) in 1960. The entire system, which included printer, optical reader, central processing unit, magnetic storage, etc. was sold for $146,600 [22], a price tag which, if sold by today's standards, would amount to over a million dollars. The IBM 1418 Optical Character Reader could only handle the ten numeric characters, the dash symbol, and the lozenge symbol. A later model, however, the IBM 1428, was alphanumeric. The alphanumeric reader could be programmed to read several document layout types assuming that they were printed in the correct font and format. Recognizable documents included premium notices, charge sales invoices, operations and route slips, payroll and dividend checks, and mail orders [23]. Throughout the 1960's, fixed-font OCR continued to be utilized and improved upon due to its usefulness in a variety of industrial applications. Some of the devices from this era are, in fact, still used even to this day for applications such as mail sorting and banking.

Although commercial OCR systems from the 1960's and early 1970's were primitive by today's standards, they were quite successful during their time. Maintenance costs for word processing, a then expensive resource, could be reduced significantly with ordinary typewriters used for drafting and their OCR results used for

final editing [24]. Fixed font OCR, although primitive, indeed proved to meet most of the requirements set by industry. For purposes of Assistive Technology, however, it was of little to no use. The blind or visually impaired community needed an optical reader to understand, not only OCR-specific fonts and layouts, but a wide variety of printed documents including books, newspapers, magazines, text books, etc. just as the idea of OCR originated primarily for the purpose of Assistive Technology, some of it's most important breakthroughs were driven by this same incentive.

Figure 7 The IBM 1401 System (Optical Character Reader not shown here). From left to right, the punch card reader/writer, mainframe, printer, and magnetic tape units.

2.1.2 Omnifont

A major commercial breakthrough in the field of OCR came with the introduction of Ray Kurzweil's Reading Machine in 1976 [9]. Up until this time, all OCR systems were tailored to a specific font, or perhaps a specific set of fonts. This font limitation can be attributed to the template matching algorithms commonly used at the time, which would compare each incoming character image to a library of bit-mapped images. Although recognition of a larger set of fonts can be made possible through the addition of more templates into the library, too many templates would cause the processing speed of each character to decrease significantly. Although it would be ideal to have a set of fonts which could encompass all possibilities in the template

library, this would prove unfeasible as there would be such a wide range of possibilities.

Omnifont recognition is characterized primarily by its use of sophisticated feature extraction techniques. As opposed to the brute force character-by-character template matching algorithms utilized in earlier systems, feature extraction enables recognition of characters irrespective of the font or typeset they are in. These techniques find properties which are relatively invariant for the same character with respect to the kinds of changes that occur across different typestyles. These properties can often include line segments (vectors), concavities, and loops. For example, the properties of a standard capital "B" include two loops on top of one another. Although the number “8” has this same feature, it does not have a straight edge on the left side as does the “B”. Furthermore, it is often that the two characters can also be disambiguated based on contextual analysis. For instance, if a character with the two vertically adjacent loops is detected at the beginning of a word, this character is far more likely to be a letter than a number.

The Kurzweil Reading Machine, used feature extraction and could be trained on any number of fonts. Once the system was trained on a given font (a process taking several hours), the knowledge would be stored on disk so that retraining would no longer be required. The system could be trained to handle up to nine fonts simultaneously [10]. If the page contained pictures or multiple columns, the user would be required to specify their locations manually. While sophisticated techniques have been developed to address the problems of document analysis, the following subsection section will focus on work which has been done to prevent any retraining from being required on new fonts. With the enhancements in processing speed and more abundant memory attributed to the advent of microprocessors, it became possible to implement much more intelligent systems utilizing complex pattern recognition approaches, as will be discussed in the following section.

2.2 Pattern Recognition Techniques in OCR

As with all pattern recognition applications, in OCR some combination of feature selection, extraction, and classification is essential. In general, a statistical classifier will observe the features of its input and, based upon those features, choose the optimal class label to which the input should be associated. For a given problem, there are often many combinations of features and classifiers from which acceptable results

may be obtained. The choice of classifier and feature set is largely application dependent, and, as of yet, no “one fits all solution” has been found. For OCR there are many such combinations which have been proven to yield near perfect results. This, of course, is to be expected, in that OCR is one of the most historically well-studied areas in the field of Pattern Recognition. Not only are pattern recognition techniques fundamental to character-by-character classification, but they are also essential for the detection of merged or broken characters, text lines, word recognition and linguistic analysis, and, as will be discussed in Section 2.3, document layout analysis. While a broad overview of all techniques utilized for OCR would be outside of the scope of this thesis, some of the most fundamental and important ones will briefly be discussed.

2.2.1 Text Line Finding

Character and word classification algorithms typically operate under the assumption that the unknown text to be recognized is already in the fully upright position. This is an unrealistic assumption given the many possible angles of skew with which the text may have been scanned. Skewed text, as illustrated in Figure 8 [25], is commonly encountered by most OCR systems. Assuming that page layout analysis has already extracted all of the columns and text blocks, it is then necessary to recognize angle of skew for each block. This is essential, not only so that characters may be rotated to their upright positions prior to classification, but also to prevent words and characters in vertically adjacent rows from becoming mangled inappropriately. Individual character classification algorithms will often utilize a character's positional information within a row as a distinguishing feature. As illustrated by Figure 9 [26], there is much information about a character which can be derived from where it's top, middle, and bottom portions reside within a row. In order to have access to such information, accurate text line finding algorithms are essential. Some of the most important techniques are briefly discussed.

Horizontal Projection Profile. One of the most straightforward methods for determining the skew angle of a document image uses horizontal projection profiles. When the horizontal projection profile is applied to an M x N pixel image, a column vector of size M x 1 is obtained. Elements of this column vector are the sum of pixel values in each row of the image [27]. The contents of this vector are at maximum amplitude and frequency when the text is skewed at zero degrees since the number of

co-linear black pixels is maximized in this condition. One way in which the horizontal projection profile can be utilized is by rotating the input image through a range of angles while calculating the projection profile for each one [28]. Each projection profile is then compared to determine which one has the maximum amplitude and frequency. Although much work has been done in order to optimize this approach, there are still more efficient and accurate methods which can be utilized [25].

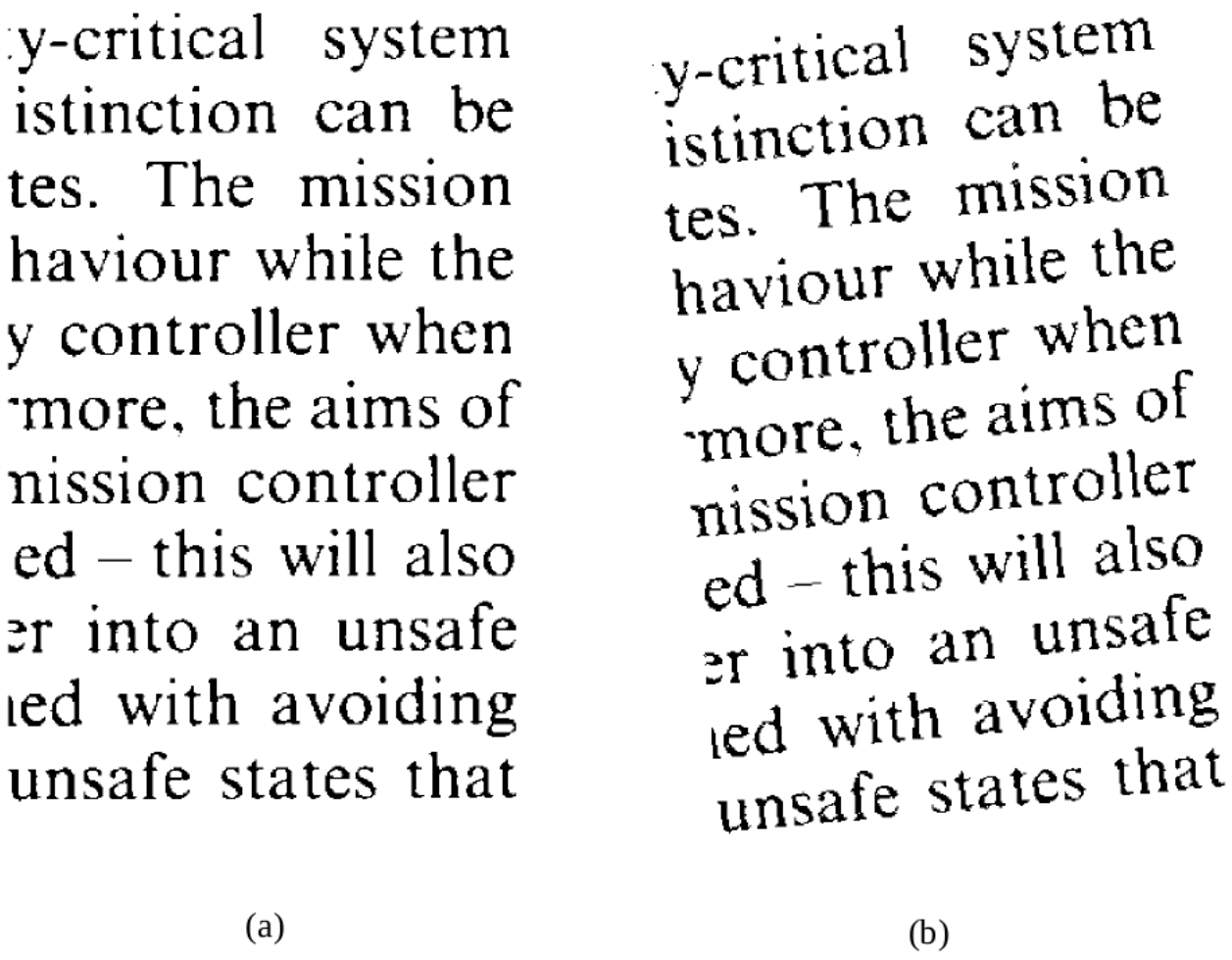


Figure 8: Original image in correct alignment (a) and skewed by 5 degrees (b). Image borrowed from [25].

Hough Transform. The Hough transform, a well known feature extraction technique in computer vision, can be utilized in order to detect, not only the skew angle of a document image, but any mathematically tractable shape of interest. This technique, when applied to 2D images, will take a series of (x, y) coordinates (for the case of document images this will likely correspond to groups of connected pixels) and

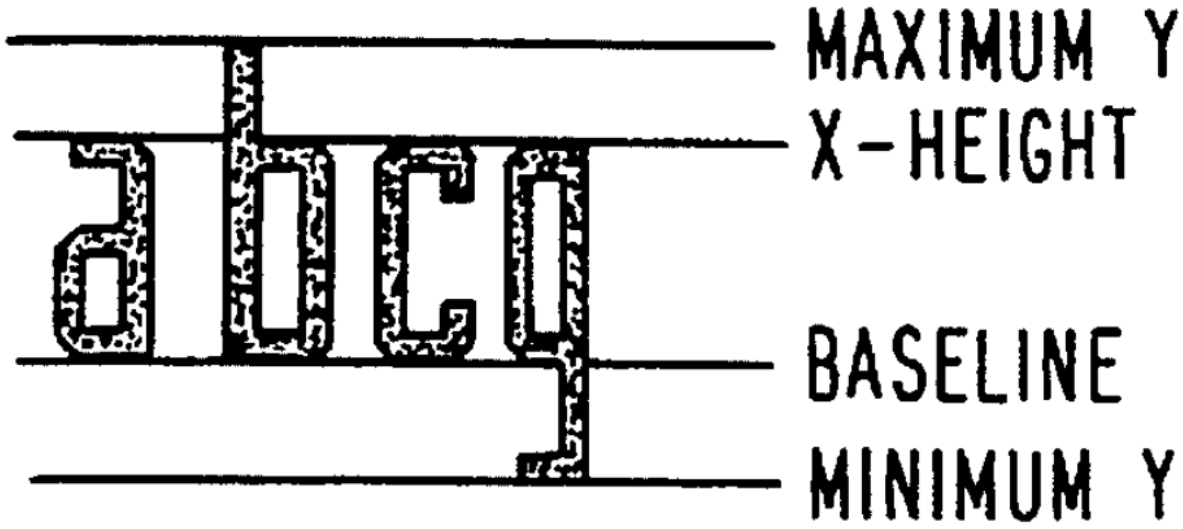


Figure 9: For printed text, a given character often has a precise position within the text line which can be useful for classification purposes [26].

transform them into a new coordinate space. While the coordinate space will vary depending upon the desired shape to be detected, for straight lines the x and y coordinates will be converted to the (ρ, θ) coordinate space using the following equation:

$$\rho = xcos(\theta) + ysin(\theta)$$

Where ρ is the distance of the (x, y) point from the origin (usually at the upper left-hand corner of the image), and θ varies between -90° and 90°. The (ρ, θ) parameter uniquely represents a given line in the image by specifying its perpendicular angle and distance with respect to the origin as shown in Figure 9.

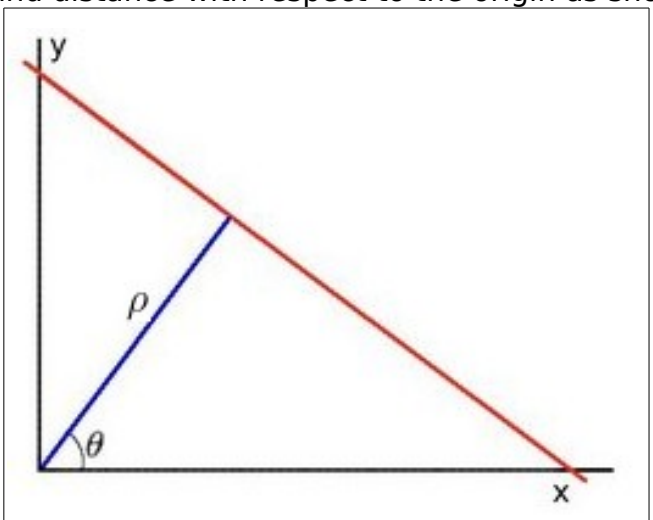


Figure 10: Mapping from (x,y) to Hugh space

2 Literature Review

For each chosen (x, y) coordinate within the image, the Hough transform algorithm will calculate the ρ values corresponding to some subset of the possible θ values between -90° and 90°. There can be an infinite set of lines going through a given point, thus the amount of lines required per point depends upon the desired accuracy of the system as well as desired overall computational speed. The set of chosen lines per point, each represented in Hough parameter space (ρ, θ), is represented by an accumulator array [29], each entry of which corresponds to a unique line in the image. Each time that a line is found to go through an (x, y) point of interest, it's corresponding entry in the accumulator array is incremented. When the process is completed, the accumulator array entries with the highest increments will correspond to the lines which intersect the most points in the image.

For OCR purposes, lines of text may be found within the image based upon this operation. The (x, y) coordinates of interest typically correspond to the centroids of connected components (groups of connected foreground pixels which often correspond to individual characters). When several parallel lines are found to have very high entries in the accumulator array, this will often mean that the page was scanned at the skew angle corresponding to these lines, and that they are likely to represent individual lines of text within the document.

Geometric Distribution of Connected Components. The Hough Transform has been utilized in various techniques to achieve accurate skew results. For a more complete survey of past techniques the reader is referred to [25]. These techniques, for the most part, vary, not by their use of the Hough Transform, but by their method for determining connected components which are of interest and most likely to correspond to rows of text. In [30], Smith utilizes an efficient and simple algorithm which, unlike previous methods, finds lines of text independently of the page's skew. The connected components of the image are extracted and filtered such that the remaining components are most likely to represent a body of text. The connected components are then sorted based on their positions in ascending order from left to right and iterates through them. Each connected component is added to a row of text to which it is most likely to belong based on vertical overlap. If no such row exists then it is created. Based upon which connected components are added to which rows, a running average is kept on the slope of the text rows. This process is continued in an iterative fashion until all connected components have been associated with rows. This algorithm has been found to achieve accurate results while proving to be more efficient than corresponding Hough Transform based algorithms.

Curved Text Line Detection. Even when text lines are accurately found, it is often the case that the lines will need to be fitted to the text more precisely due to scanning artifacts which may give the text a curved appearance as depicted in Figure 10. Among the techniques utilized for this problem are quadratic or cubic spline modeling via least square fitting techniques [31] as well as active contour tracing via snakes [32]. Smoothing techniques are often applied in order to simplify the input for curved line detection. The optimal technique to be applied largely depends upon the type of document fed into the OCR system. Thus document understanding at early stages in the OCR process is of great importance in achieving accurate results. For a more complete account of text line detection in various documents, the reader is referred to [33].

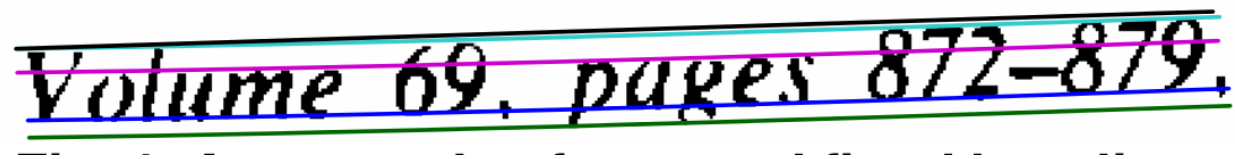


Figure 11: An example of skewed and slightly curved text borrowed from [31]. Close inspection shows that the cyan/gray line is curved relative to the straight black line above it.

2.2.2 Character Feature Extraction

The problem of feature extraction for optical character recognition, although a difficult task, has been extensively studied in the literature. Techniques vary based upon their application, with handwritten recognition often requiring different techniques from printed character recognition. As with the rest of this thesis, the focus here will be on techniques pertaining to printed character recognition. Techniques utilized by Google's OCR System, Tesseract, will be emphasized and discussed primarily since they are used within this thesis project.

Edge Extraction. After text lines have been located as discussed in the previous section, the next task for a typical OCR system will be to perform some image processing operations on the input in order to make features more easily and efficiently recognizable. In Tesseract [26], utilizes a novel edge operator which can take advantage of grayscale values if they are available to achieve robust character segmentation results. Text and non-text can often also be distinguished based on contextual evidence as well as using basic height/width filters. Furthermore, the edge

extraction algorithm will inherently filter out a significant amount of noise since it will disregard any portions of the image which do not form closed loops. The term “closed loop” is used here to describe a contour which, after being followed a certain amount of time will return to its starting position.

Also of importance is preserving the relationships between the inner and outer portions of characters. Take, for instance, the character “o” depicted at the left on Figure 12 [34]. Since the edge detector will find the inner and outer portions of this character as separate, simple data structures must be implemented which store the relationships among overlapping edges. In Tesseract, a 2D bucket sorting technique is utilized in order to store all of the inner portions of characters as enclosures or “holes” within them. The results of edge extraction are stored in chain code format as illustrated by Figure 13 [35].

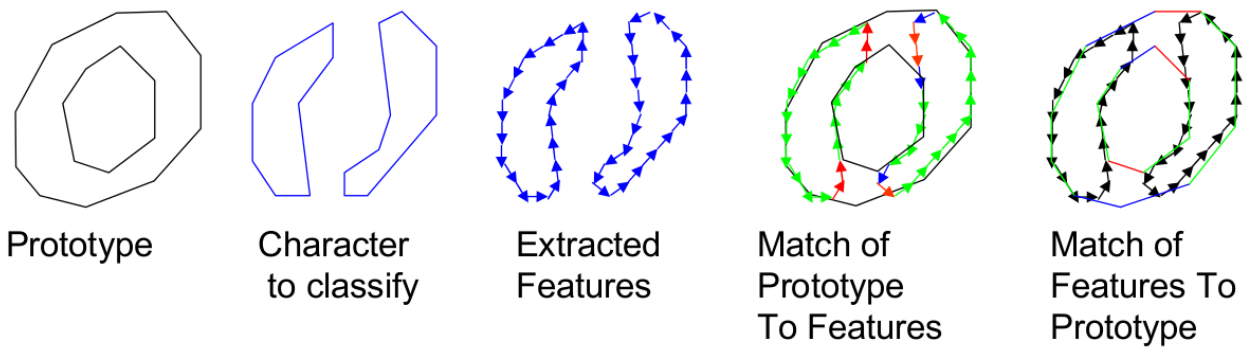


Figure 12: Comparison of a prototype for “o” to an unknown character by template matching [34].

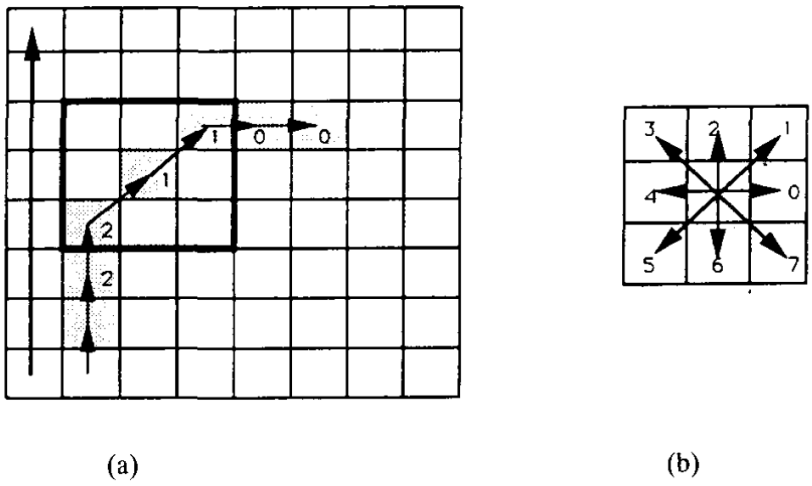


Figure 13: (a) Example of chain-code (b) Chain-code directions. [35]

Polygonal Approximation. In Tesseract, the process of polygonal approximation is required in order to optimize the efficiency and accuracy of subsequent feature extraction techniques. Polygonal approximation of a character image, if done effectively, results in an output whose data is neither too fine or course for purposes of feature extraction [36]. It becomes easier to detect global convexes and concavities as well as character enclosures, which are very important features.

The process of polygonal approximation utilized by Tesseract analyzes the chain code output of the edge extractor in order to locate simplifications which can be made, which will enhance the robustness of subsequent feature extraction techniques. The process begins by first breaking up the character into directional segments, separated by 90° or two subsequent 40° transitions [26]. The second stage involves further analysis of these segments and subsequent approximations being made between the end points of each segment. The process is repeated iteratively until certain criterion are met.

Normalization and Template Matching. After polygonal approximation and prior to feature extraction, normalization is applied to the input in order to eliminate some of the complexities which may come about from various font differences. For an in depth discussion on normalization techniques the reader is referred to Chapter 3 of [36]. Normalization is very important in accounting for character font transformations which may occur in terms of size, perspective, and rotation with respect to the features of prototypes used in training the system. Normalization techniques can, in general, be separated into categories using either linear or nonlinear methods. While linear methods account for affine transformations often found in printed characters, nonlinear techniques are generally geared more towards handwritten character recognition wherein much more drastic variation is to be expected.

Normalization can be performed either before or after feature extraction. If done after feature extraction, then the process is carried out within the feature space rather than directly on the character's pixels. In the case of Tesseract, normalization is carried out on the feature space of the character's polygonal approximation, which can be viewed as a vector of 3D features, the dimensions of which are simply x position, y position, and direction within the range of [0, 2pi] [37]. Figure 13 gives an example of how Tesseract will normalize the features of unknown characters while matching them to those of character prototypes.

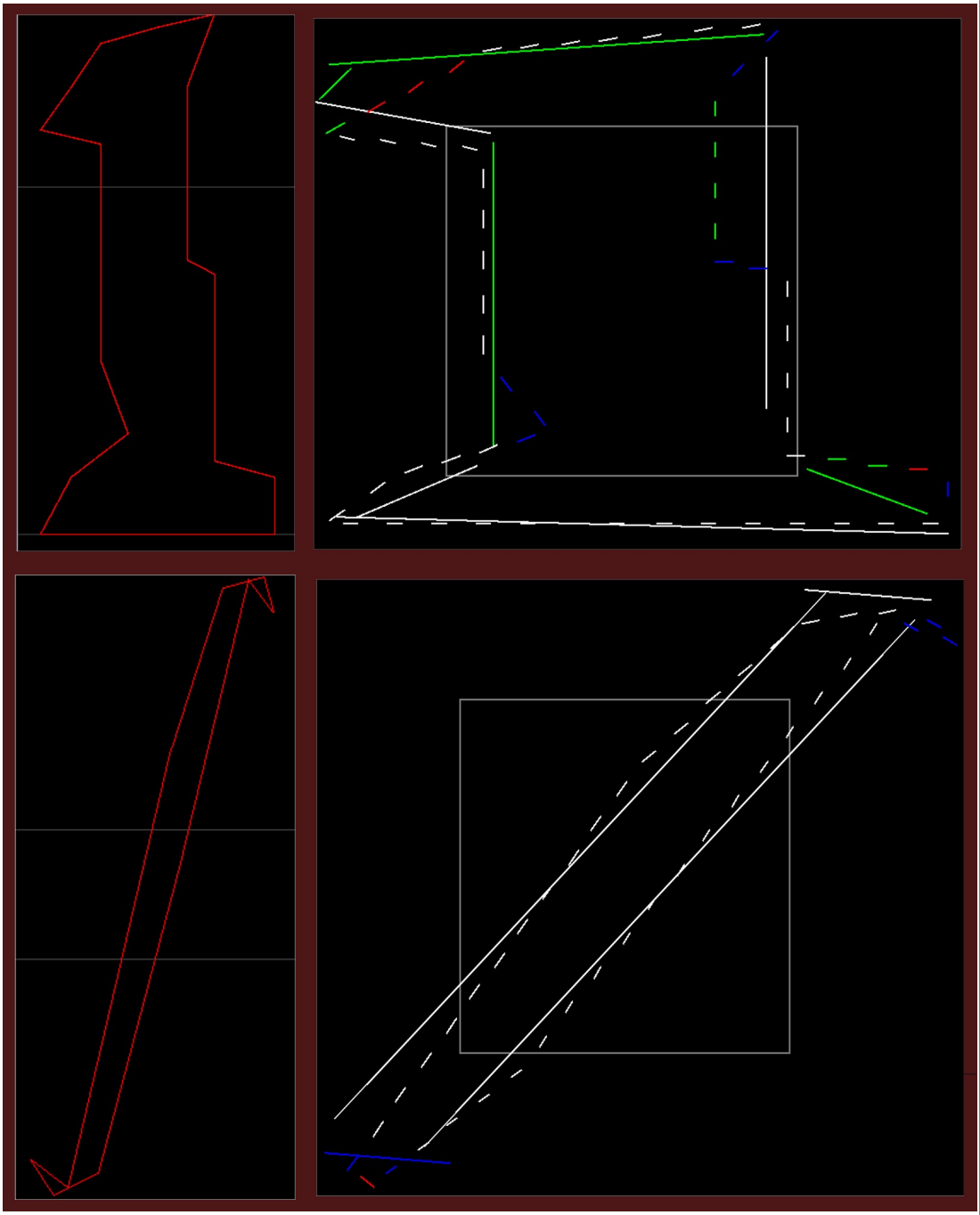


Figure 14: (top) The polygonal approximation features of a “1” followed by those same features after normalization with respect to the prototype of “1” to the right. (bottom) Features of an “integral,” a character for which there is currently no valid template in Tesseract. To the right is the integral after normalization with respect to the prototype of “/”. For both normalized pictures, the solid lines represent the prototypes while the dotted lines represent the normalized unknown character. Lines are colored from best to worst match: white, green, red, blue. These images were taken from Tesseract's debugger.

As illustrated by Figure 13, Tesseract normalizes a feature vector by each character prototype to which it is compared. For instance, assume that the character “8” is fed into the system. Based upon a coarse shape analysis of the character a subset of the the total prototypes may be chosen as potential candidates. For instance “B” may be chosen since it has two enclosures, and “0” may also be chosen based upon its convex top and bottom regions. Assuming that only these characters are chosen as candidates, the feature vector for “8” will be subsequently normalized based upon both of these prototypes prior to the respective template matching. The process of normalization begins by isotropically scaling the bounding box to a fixed height and width. The feature vector is then centered and scaled anisotropically based upon the second moments of the prototype to which it is being compared [37]. Moment-based character normalization has been studied extensively in the literature dating back to even before the advent of microprocessors. For some examples of in-depth studies the reader is referred to [38] [39].

During the classifier training, the training data is grouped into clusters based upon certain important features. These feature clusters are then utilized during classification to reduce computation time with very little loss in accuracy. The five most important features utilized by Tesseract will be herein briefly discussed.

Concavities. One of the most important features in character recognition are concavities. By definition, a concavity is part of an outline which does not lie on it's convex hull (the smallest convex region enclosing the outline as illustrated by Figure 14 [40]). In Tesseract, a concavities are characterized by the direction of their hull line, their centroid [26], shape, skew, and area.

Figure 15: Red outlines represent convex hulls [40].

Functional Closures. Character closures are common features which can be useful in distinguishing characters regardless of their font. For instance the “e” and “o” characters will both always consist of a single closure no matter what. The term functional closure is useful when a character's closure may be slightly degraded somehow, such that there there may be an unintended opening. In Tesseract [26], each concavity is tested for functional closure. Based open the location of the concavity within the character (i.e. upward facing, downward facing, etc.) a threshold is assigned for the maximum character to concavity width ratio expected for a functional closure. If the ratio is below the appropriate threshold then a functional closure will be detected.

Axes. Tesseract defines axis features only on characters for which no concavities or closures are detected. Characters including commas, periods, quotations, etc. fall under this category. The axis feature measures a character's length to width ratio. The length of a character is determined by finding a point on the outline whose distance from the character's centroid is maximum. The vector going from the point to the centroid is said to be the character's major axis. The character's width is then calculated as the sum of the maximum perpendicular distances from the major axis to the character outline on either side of the axis. The major axis length to character width ratio can be useful in disambiguating commas, periods, quotes, etc.

Lines. As illustrated by Figure 13, lines are useful features in template matching. Line features are only used by Tesseract for unknown characters which closely match more than one of the prototypes, as measured with concavities, closures, and axes [26]. The degree to which a line in the unknown character matches a line in a prototype is measured based upon the normalized position of the center of the line, its quantized direction, and its scaled length.

Symmetry and Detection of Italicized Characters. Vertical as well as horizontal symmetry can be a very useful measure in discriminating certain characters. For instance, the character “C” and “G”, “j” and “/”, “j” and “]”, “T” and “1”, etc. can often be disambiguated through their respective measurements of symmetry. The main difficulty in symmetry measurement is not in measuring the degree of symmetry about an axis, but rather in locating the axis of interest. While the problem is trivial for vertical text (simply drawing a vertical line through the center will suffice), italicized text is much more difficult since the axis is rotated slightly and may be difficult to locate. Tesseract utilizes two methods to determine a character's axis of

symmetry. Once this axis is found it is then easy to determine whether or not the character is italicized.

The first method used by Tesseract searches the character's outline for a vector which passes from the bottom to the top half of the character's bounding box. The direction of this vector may be a good indication for the direction of the axis of symmetry. For round characters such as “o” and “e” and those which contain vertical lines such as “H” and “p”, this method is useful. However, for angular character such as “X” or “8”, a valid result is not produced. The second method finds the rightmost point on the outline then calculates the most clockwise line which can be drawn through this point, without intersecting any other point on the outline. This operation is repeated on the leftmost point of the outline as well. The line which was least clockwise from the vertical becomes the axis of symmetry.

After the axis of symmetry found, the outline is searched around the axis for points of reflection. Symmetry testing is commenced at a point where the axis intersects the outline and works in opposite directions simultaneously. The points are tested for being in the same locality of the point on the opposite side of the axis. Symmetry is only measured for certain character candidates and typically only in one direction (either vertical or horizontal).

2.2.3 Character Classification

The line finding, edge extraction, polygonal approximation, and feature extraction techniques discussed thus far would be of little to no value without an effective classifier. In pattern recognition, a classifier will take a set of feature measurements as input and, using these measurements, choose from a finite set of classes, the class to which the unknown input is mostly likely to belong. In the case of OCR, the classes will often correspond to individual characters. Tesseract employs two separate classifiers, one is termed the static classifier while the other is the adaptive classifier. In order to save computational time, a class pruner is utilized first to narrow down the number of candidate classes for an unknown character.

Tesseract Class Pruner. In the first stage of classification, Tesseract will employ its class pruner in order to reduce the number of potential candidates to which an unknown character is to be compared. The class pruner uses a fixed quantized version of the 3D feature space wherein each of the 3 dimensions (x, y, theta) are quantized into 24 cells. After the unknown character's features are quantized, they are

indexed to the quantized feature space in order to obtain a set of classes which allow the given features. The number of feature hits for each class is summed and the best few matching classes are then fed into the next stage of classification [37].

Tesseract Static Classifier. Both Tesseract's adaptive and static classifiers are unique when compared to more standard techniques in that they operate on a variable number of features. While standard classifiers such as neural networks, support vector machines, etc will work in a feature space of fixed dimension, Tesseract has a variable number of features for each class of interest. The classifer can be regarded as an optimized K-Nearest-Neighbor (KNN) classifier where the character class, k, with minimum distance from the unknown character is computed as follows:

$$argmin(k)\frac{1}{M+J_k}\left(\sum_{l,i}\left(x_{il}-\mu_{ijk}\right)^2+\sum_{j,i}\left(x_{il}-\mu_{ijk}\right)^2\right)$$

Where the variables are as follows: i is the current feature dimension (either x position, y position, or θ); j is the cluster; k is the character class; and l is the unknown's feature. x_{il} is the feature dimension (either x position, y position, or θ) of the unknown character x's feature at index l. M is the total number of feature vectors in the unknown character, x (this varies depending upon the character of interest). J_k is the total number of character clusters which the training set was divided into. μ_{ijk} is the mean feature value for the i^{th} feature, j^{th} cluster, and k^{th}class calculated during training.

While the left-most summation measures the distance between each feature dimension and its corresponding average clustered prototype value, the right-most summation measures the distance between the average value in each cluster to the corresponding feature dimension. The result of these summations is then divided by the total number of features in the unknown character and training set. A key advantage to this approach is it's symmetry. The nearest matching features between both the unknown and prototype and the prototype and the unknown are effectively found. Say, for instance that the unknown character is “e” and the prototype to which it is compared is “c”. Since most of the features in “c” are allowed by “e”, it becomes possible that the “e” will be misclassified as “c” if only the distance between the “e” and “c” is computed. When the distance between the “c” and “e” is added into the

classification, the lack of crossbar in the “c” will incur a penalty, thus lowering the risk of misclassification.

Tesseract Adaptive Classifier. After word recognition, as will be briefly discussed in section 2.2.5, a second pass is made by Tesseract's classifier. This time the classification is considered to be adaptive in that it utilizes the extra information obtained after word recognition in order to better train the classifier to the current font. After word recognition is carried out, there may be several characters which can be disambiguated and thus used to better train the classifier on the second pass and increase accuracy. The adaptive classifier is essentially the same as the static one except that it applies a different type of normalization to the unknown character prior to comparing it to the prototype. While, for the static classifier, the centroid of the unknown character is centered in the feature space and then scaled anisotropically to normalize the second moments of the outlines, the adaptive classifier will normalize the unknown by centering the horizontal centroid of the outline and scaling isotropically to normalize the x-height of the character. This normalization retains font differences, which, at this stage of OCR, is very important [37].

2.2.4 Detection of Merged or Broken Characters

While some of the first OCR systems would only recognize each individual character independently, more sophisticated systems such as Tesseract, Omnipage, and Abby Fine Reader, ReadIris, etc. analyze inter-character relationships in order to increase their systems' robustness in the presence of noise. In Tesseract, while the results of word recognition (described in Section 2.2.5) are unsatisfactory, a character merger/segmenter module is utilized in order to test the word on new potential character candidates in areas with low character recognition confidence. The merger/segmenter module will locate concave vertices of a questionable character's polygonal approximation and attempt to separate the character in those locations to test for a possible merged character as illustrated by Figure 16. Likewise, potentially broken characters are attached to their neighbor and tested if their combined width is within an acceptable range.

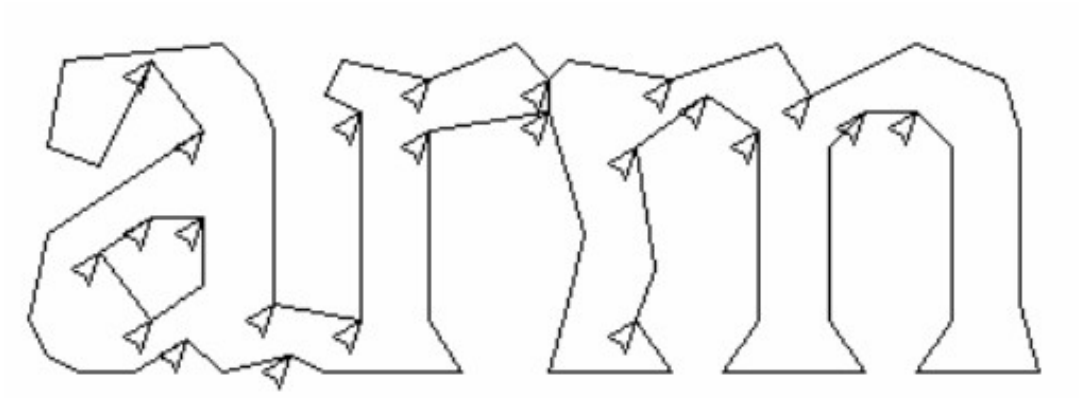


Figure 16: Example of merged letters with candidate chop points (denoted by triangles) [34].

2.2.5 Word Recognition and Linguistic Analysis

Individual words within Tesseract are detected based upon the distribution of space between characters found on a text line. Characters which are within an appropriate horizontal distance parallel to the text line are considered to be within the same words, while groups of such characters are considered to be separate words. The word recognition module looks up recognized words in a dictionary to make sure they are valid. This information is also vital to detecting broken or merged characters and training the adaptive character classifier.

Some basic linguistic information can be important for increasing accuracy. For instance, in Tesseract's English word recognition module, numeric characters are not allowed to exist in alphabetic words, uppercase characters cannot follow lower case ones, and the only punctuation allowed within a word are apostrophes. Markov Methods are also very useful in OCR due to spelling conventions (such as u following q) and the need for words to be pronounceable (i.e. g is unlikely to follow j). By modeling each individual character as a possible state and each character occurence as the next element in a markov chain, it is possible to use a transition matrix (whose width and height are 26, the number of characters in the English alphabet) to help in selecting a word's next character [26]. While it is possible to make choices based upon multiple characters, the transition matrix for making a choice based upon the previous m-1 characters would require a transition matrix of size $26^{(m-1)}x26^{(m-1)}$. Rather than using large values for m, Tesseract employs dictionary methods.

Strings of characters can be reduced into words which are either in a dictionary or can be generated through the use of various production rules [41]. For each string

of characters a set of candidate words are derived using the dictionary. The word which has the highest overall rating based upon the recognition confidence of its individual characters is chosen. The word recognition result can then be utilized in order to boost the adaptive character classifier's accuracy since certain characters which had low confidence in the static classifier may now be confirmed.

2.3 Document Layout Analysis Techniques

The improvements made in the field of commercial OCR throughout the 80's and early 90's are primarily attributed to enhanced processor and digitizer technologies rather than to improved classification techniques for individual patterns [42]. By the early 90's there had been significant progress already made toward the study of OCR and pattern recognition techniques which are still largely in use to this day. A significant amount of the more recent progress made in the state-of-the-art has been due to improvements in document layout analysis and understanding as opposed to the much more mature character-by-character feature extraction and classification algorithms.

While inflexible hardwired classification engines once dominated the market for OCR, the computational advancements of the 70's and 80's allowed for more intelligent systems take hold. While systems became robust against multiple fonts, merged/broken characters, and document skew as discussed in the previous section, the need also arose for systems which could recognize pages from a wide variety of document types. While an OCR system may be predominantly exposed to documents like newspapers, magazines, letters, etc., it is also often necessary to process such “special” documents as electronic circuit diagrams, envelopes, checks, tax return forms, music notations, etc. [43].

The importance of document layout analysis techniques is made apparent in the presence of both the former and latter document types as they may contain complex backgrounds, lines with drop-caps, mathematical formulas, various symbols, imagery, tables, graphs, multiple columns, titles, headings/subheadings, etc. Therefore, it becomes important, not only to recognize the individual words and characters, but to also interpret and preserve the layout and spatial context of a document's components. Such details as spatial context and document structure are vital in conveying a document's message as it is intended to be perceived, as well as for understanding how exactly the document needs to be processed in order to achieve

the optimal recognition accuracy. Figure 17 [44] shows two examples of document images with complex layouts.

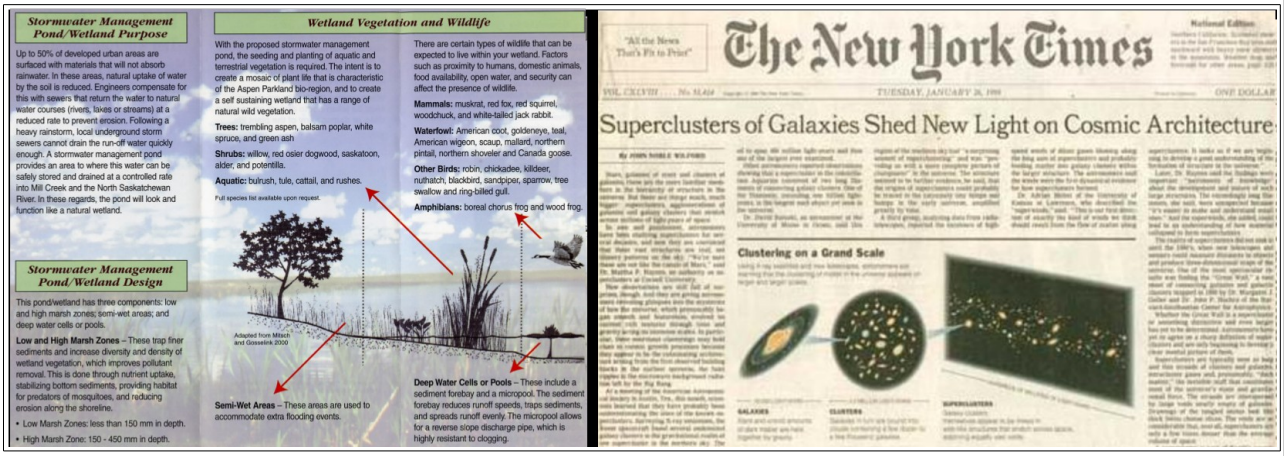


Figure 17: Two document images with complex layouts [44]. On the left, an intricate background as well as complex column structure is observed. On the right is a newspaper article with a complex layout of titles and columns along with imagery and captions. Both document images will require sophisticated image processing and layout analysis techniques in order to achieve both OCR accuracy as well as efficient data storage and indexing within computer systems.

Document layout analysis is a very important design component for any OCR system and has been extensively studied in previous literature [43][44][45][46][47][48][49][50]. Not only is document layout analysis often essential for obtaining correct OCR results, it can also provide the means for computer systems to use logical information such as titles, footers, authors, captions, abstracts, page numbers, etc. to more efficiently store and index a document image's information [51]. This contextual information is also essential for Assistive Technology purposes, in enabling blind individuals to have an understanding of the same spatial and logical cues afforded by the document's visual layout [52]. This section will discuss how the field of document analysis is divided into various sub-problems by existing literature and then compare and contrast various techniques which address these problems. The problem of document analysis can be broadly divided into its five most important interdependent components: image preprocessing, document structure analysis, document content representation, training set development, and finally performance evaluation as illustrated by Figure 18 [44]. After a brief overview of what each stage entails along

with some introduction of terminology, various techniques found in the literature for each stage will be discussed.

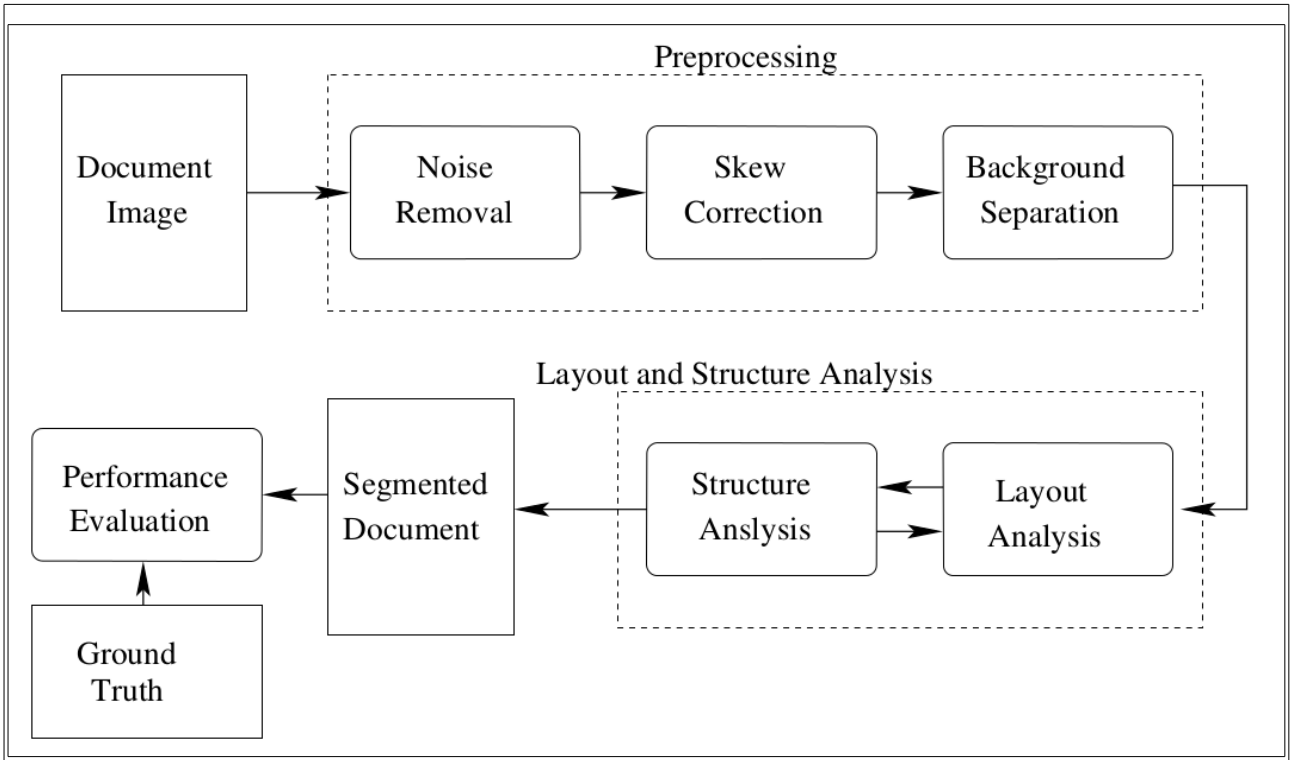


Figure 18: The five most important inter-dependent components of document analysis involve preprocessing, document structure analysis, document content representation (not illustrated here), training set development (ground truth), and performance evaluation. Each module is described as interdependent because the performance of the overall system really depends on each component. For instance, if preprocessing is not effective, then structure analysis will likely fail. If the document content representation is not consistent, then ground truth and performance evaluation will yield insignificant results. Diagram borrowed from [44].

Firstly, the most common problems addressed by image preprocessing (Section 2.3.1) in document analysis involve noise removal, separation of background and foreground regions, and skew correction. Secondly, after any necessary preprocessing is carried out on the document image, the modified image is fed into the system's document structure analysis module. From a broad perspective, document structure analysis involves first extracting the document's geometric structure and then mapping that structure into a valid logical one which can be understood by computer systems. A document is thus considered as having both a physical (geometric-based) structure and a logical (content-based) structure. Thus document structure analysis is commonly divided into two distinct phases: physical layout analysis and logical layout

analysis. Each distinct phase of document structure analysis will be further discussed in Section 2.3.2.

Thirdly, an important question which must be asked prior to the design of any structural layout system, is how exactly a given document should be represented by a computer internally. This brings about the problem of document content representation (Section 2.3.3) which also has been extensively addressed by the literature. A standard format for all documents is of course desired, however the problem of finding a standard format which could accommodate all possible document image layouts has been no trivial one. A variant of SGML or HTML is commonly employed, however the tradeoffs between different representations as well as grammars will be briefly reviewed. Fourthly, given the knowledge of how a document should be represented internally, an important question is what training sets and groundtruths are available for evaluating performance (Section 2.3.4). Since building a training set manually is an extremely expensive and time consuming process, the ideas of creating synthetic training data and/or training a system to automatically generate training data from documen[64]ts while requiring little to no human correction, have been explored. Finally the issue of performance evaluation has been explored through various techniques which will be discussed in Section 2.3.5.

2.3.1 Preprocessing

The most common problems addressed by image processing involve noise removal, separation of background and foreground regions, and skew correction. Since skew correction was already covered in great detail by Section 2.2.1, it will not be further discussed in this section. Noise removal in image processing is a well studied field and advanced techniques have been developed to cope with white noise, salt and pepper noise, quantization artifacts, etc. Such noise sources are often compensated for by using techniques such as median filtering, dithering, low pass filtering, etc. [49]. An in depth overview of noise reduction methods in image processing can be found in [53]. For purposes of document layout analysis, one of the more important noise removal tasks involves the detection and filtering of half-tones. This discussion will be followed by a brief overview of preprocessing tasks for background and foreground separation.

Noise Removal: Dealing with Half-tones

Halftones, as illustrated by Figure 19 [54], utilize variably sized or space dots in order to create the optical illusion of an infinite range of colors while, in actuality, only printing a limited amount. Half-tones are utilized by color and grayscale printers in order to reproduce imagery while requiring few colors of ink. Figure 19, for instance, creates the illusion of grayscale while only requiring black dots. When scanned at high resolution, the halftones in a document become a significant noise artifact, as an image's connect components clearly should not be divided into such small dots for document analysis purposes. Halftones can be detected through the use of various filtering techniques [55], whose accuracy often depends upon the dot sizes and spaces of the halftone in question. Once detected, the halftones can be converted into continuous grayscale by applying an appropriate low-pass filter to smooth out all of the dots, followed by a sharpening technique which will reduce the blur.

Figure 19: An example of a halftone image [54]. Notice that, when looking at the image from a distance, the illusion is created that the image is in grayscale, when, in fact, it is actually printed with only black dots of varying sizes.

Background and Foreground Separation

Although the problem of foreground detection is often very simple in the case of the most typical black text on white background, the problem becomes much more complex when faced with intricate backgrounds which are overlayed with text in varying color, size, and font as depicted in Figure 20. In the former case it is possible to use thresholding techniques like Otsu's Method [56]. An alternative method which could work for varying background and foreground color schemes would be to find the outline of characters through edge detection [26]. In the presence of complex backgrounds, however, more sophisticated background and foreground separation techniques may be required. A common approach is to compute statistical properties of image patches and assign them as either foreground or background using a trained classifier such as a neural network [44]. Through a combination of edge detection and a trained classifier it becomes possible to detect foreground text of varying colors on a complex background with a certain degree of confidence as demonstrated in Figure 20.

Figure 20: (Left) Part of a document image with complex background. (Right) The same image with foreground separated from background [44].

2.3.2 Document Structure Analysis

A primary component of any document analysis system is the document structure analysis stage itself. As previously indicated in Figure 18, however, the steps of preprocessing, document content representation, training set development, and performance evaluation also play a crucial role. In this section the term “document structure analysis” is used to refer to the broad class of both physical and logical document structure analysis methods which will be explored in this section. In general, physical layout analysis techniques are one of the first steps of an OCR system and will initially divide the document image into areas perceived as text and non-text, as well as splitting multi-column text into columns [57]. In this literature review an important distinction between physical and logical layout analysis techniques is made such that, while logical layout analysis techniques make final classification decisions on blocks, physical layout analysis techniques extract and evaluate the geometry of blocks without necessarily reaching any final conclusions on their syntactic meaning. While the physical layout analysis stage looks for geometric patterns, the logical layout analysis stage will utilize this and other information in order to infer a document's meaning from a syntactic perspective (i.e. the type of document and the location and functional purpose of its “zones” which may include titles, headers, footers, math equations, imagery, etc). This logical understanding is very important both for indexing and storage purposes as well as for Assistive Technology applications as previously mentioned.

A document analysis system must be able to understand, not only how a document can best be partitioned into its logical sections, but also the role that physical geometry plays in conveying information effectively. It is important for a document recognition system to move back and forth between physical and logical analysis in an intelligent manner which may vary significantly depending upon the aspects of what is being recognized. This concept is illustrated by the bidirectional arrows seen in Figure 18. Although it is possible for a very specific physical layout to match to only a single logical structure (i.e. in the case of a very complex and unique form), there is never a guaranteed one-to-one mapping between any physical and logical layout or vice versa. In creating a system that can generalize to a wide variety of document structures while minimizing overfit, it is thus important not to make assumptions too early based solely upon geometric information. A system may require to make “fuzzy” decisions which, in later steps, can be further refined to reach an

appropriate solution. For instance, if text is found to be centered within a column this could open up many distinct possibilities based upon the contents of the text itself as well as its context within the entire page. It could, for instance, be the title of a new subsection, new chapter, a mathematical formula, a quote, image caption, or any number of other possibilities. Thus, while an understanding of the geometric structure of a block of text is important, there is more information required in order to understand the block's logical structure. If an OCR algorithm yields results with low enough confidence then various alternatives can be tested (i.e. for mathematical formulas, musical notation, other languages, etc.).

Document Physical Structure Analysis

Physical layout analysis, an essential step for all OCR and document analysis systems, localizes individual blocks of text and imagery while leaving assignment of logical meaning of these blocks as well as final classification of text/nontext regions to later stages in processing which will be discussed in the Document Logical Structure Analysis Section. Methods for physical layout analysis fall into roughly three categories: top-down, bottom-up, and hybrid, each of which will be discussed in turn by this section. An important distinction between algorithms involves the types of physical layouts which they can handle. The following three types of physical layout patterns are commonly defined: these include Manhattan, rectangular, and arbitrary layouts [58]. A document's Manhattan layout can be viewed as the document divided into a grid, which may be horizontally or vertically split recursively into smaller components in any given region. For a Manhattan layout, if a region overlaps another then it must be entirely covered by that region (i.e. there is no partial overlap). Rectangular layouts consist of several rectangles arbitrarily spaced apart or which could be partially overlapping. Arbitrary layouts, on the other hand, are formed by unconstrained polygonal shapes as demonstrated by Figure 21 [59].

Top-down physical layout analysis techniques recursively segment the document into smaller rectangles which are expected to correspond with image, column, paragraph, or other text block boundaries [51]. Bottom-up techniques, on the other hand, analyze individual pixels or connected components, recursively merging them together into larger regions. While bottom-up techniques can handle arbitrary physical layouts, top-down methods are constrained to only handling rectangular regions. A disadvantage of bottom-up techniques, however, is that they may result in over-fragmented regions. For instance, a bottom-up technique will be more likely to

properly segment small structures like individual paragraphs of text than to properly segment entire columns. Due to these trade-offs it is often that hybrid techniques, which combine top-down and bottom-up ideas, are employed [60]. Starting with top-down methods, variants of each broad category of physical layout analysis will be reviewed by this section.

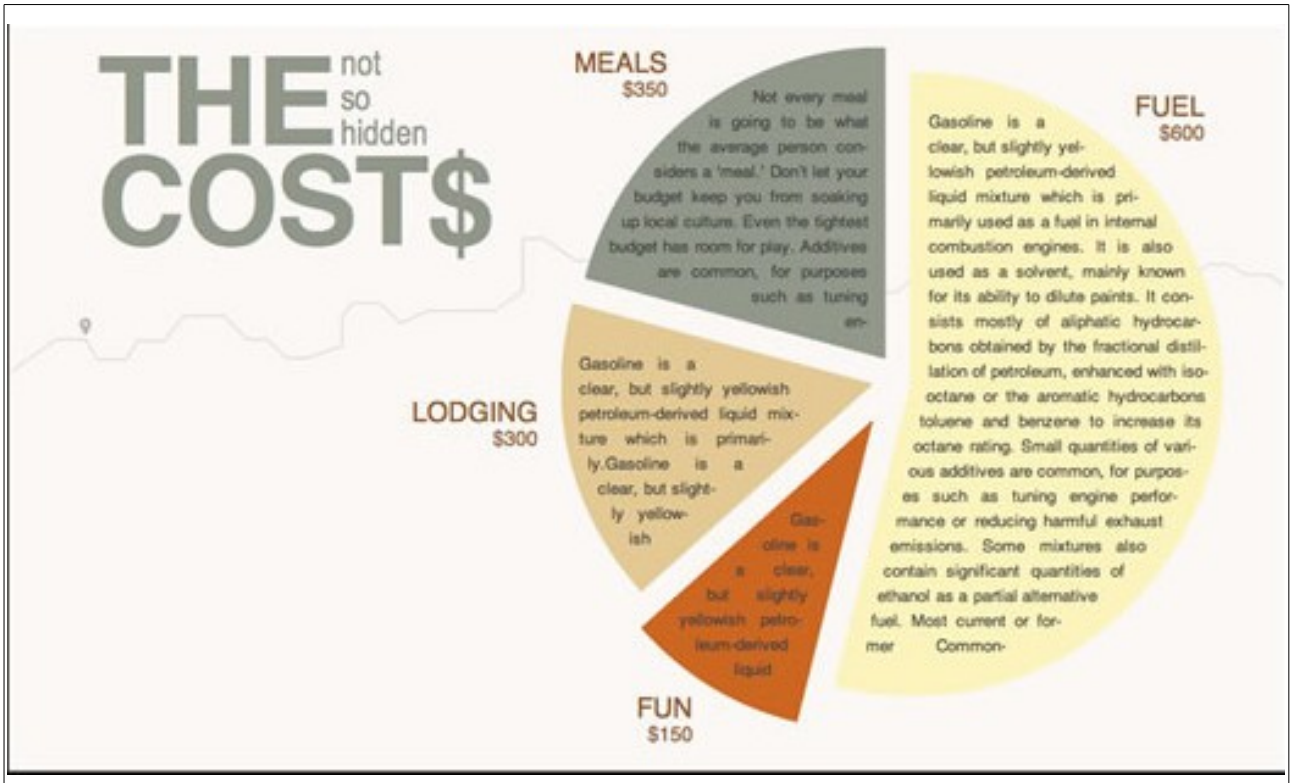


Figure 21: An example of an arbitrary physical layout wherein blocks of text are fit to the shape of a pie chart. A layout analysis system should ideally be able to segment text blocks into the appropriate shape, which sometimes may be more complicated than simple rectangular layouts. For this figure, a document layout analysis system which can only handle rectangles would be insufficient, and would likely result in a mangled output. Image borrowed from [59].

Top-Down Physical Structure Analysis

The first physical layout analysis technique to be reviewed here is the “top-down” method. Top-down strategies segment blocks based upon interpretations of the document from a high level (i.e. by first looking at a representation of the entire document and recursively splitting it into smaller components). Top-down strategies will then typically attempt to verify each segmentation by visiting each node down the

to the terminals (the lowest levels, corresponding to individual connected components or pixels) [61]. For documents having a complex layout, top-down methods are often more robust but slower than bottom-up ones. Typical bottom-up algorithms are faster, but can be less reliable since they may greedily over-segment blocks without regard to all of the available contextual information.

X-Y Cut Algorithm. The X-Y cut algorithm [62] is a top-down approach which has been utilized extensively over the past several decades. The technique analyzes vertical and horizontal projection profiles of the image to find regions of low pixel density, often termed as “valleys” [43][44]. Assuming that the document has a white background and Manhattan layout, its X and Y valleys are likely to correspond with horizontal and vertical text block boundaries respectively. For instance, these could be divisions between paragraphs and columns. The X-Y cut algorithm will start with the horizontal and vertical projection profiles of the entire image and use the largest valley (or valleys) in either direction as the first splitting point. After having made the first split(s), the algorithm will then recursively make further splits within each sub-region using the same methodology. The document's physical layout is represented by an X-Y tree data structure wherein each node represents a split region. If the algorithm is correct, then the terminal nodes of the tree will correspond to the individual text blocks. Once the terminal nodes have been located, the algorithm will backtrack through the tree structure to ensure that the physical structure is appropriate based upon some preconceived notions of expected document structure. A possible result of the algorithm is illustrated by Figure 22. In order for the X-Y cut algorithm to work correctly, it is vital that the document first has its skew corrected. If, for instance, the horizontal projection profile is taken for a document that has been rotated by several degrees, then many of the “valleys” will not be found correctly and thus the algorithm will fail.

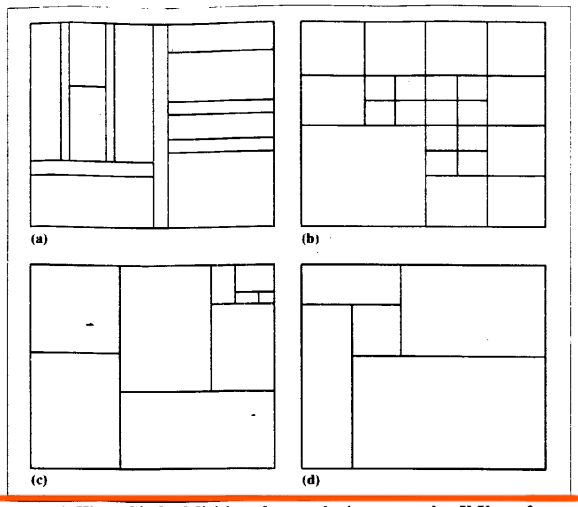


Figure 2. Hierarchical subdivision of rectangles into rectangles: X-Y tree for page segmentation (a); Quad tree for comparing or combining several images (b); K-D tree for fast search (c); example of tiling with rectangular blocks that cannot be obtained by successive horizontal and vertical subdivisions only (d).

to the various components to construe a valid graphic "sentence."

Although we use compiler tools developed primarily for formal languages, the syntactic analysis of document images exhibits many of the difficulties of parsing natural language. Layout conventions may be insufficient to identify *every* document component. For instance, text lines with equations buried in them may radically alter the expected line spacing. We must therefore ensure that minor deviations have only local effects. Furthermore, the grammar for a modern programming language is established from the start, while document grammars must be inferred indirectly, as later discussed. (A never-ending task: Journals frequently put on new faces.)

Block grammars. The document grammar for a specific journal consists of a set of block grammars. Each block grammar subdivides a block horizontally or vertically into a set of subblocks. The net result of applying the entire document grammar is therefore a subdivision of the page into *nested rectangular blocks*. Such a subdivision can be represented efficiently in a data structure called the X-Y tree[6] (Figure 2). The block grammars themselves are also organized in the form of a tree: The block grammar to be used to subdivide each block is determined recursively by the results of the parse at the level above.

Syntactic attributes

A (horizontal/vertical) *block profile* is a binary string that contains a zero for each horizontal or vertical scanline that contains only white pixels; otherwise it is a one.

A *black atom* is a maximal all-one substring. It is the smallest indivisible partition of the current block profile. A *white atom* is an all-zero substring.

A *black molecule* is a sequence of black and white atoms followed by a black atom. A *white molecule* is a white atom that separates two black molecules.

An *entity* is a molecule that has been assigned a *class label* (title, authors, figure caption). It may depend on an ordering relationship.

This approach effectively transforms the difficult two-dimensional segmentation into a set of manageable one-dimensional segmentation problems.

The syntactic formalism is theoretically well understood, and sophisticated software is available for lexical analysis and parsing of strings of symbols. Each block grammar is therefore implemented as a conventional string grammar that operates on a binary string called a *block profile*. The block profile is the thresholded vertical or horizontal projection of the black areas within the block. Zeros in the block profile correspond to white spaces that extend all the way across the block and are therefore good candidates for the locations of subdivisions.

Representing the structure of an entire page in terms of block grammars simplifies matters considerably. But each block grammar itself is a complex structure. It must accommodate many alternative configurations. For instance, to divide the title block from the byline block, the block grammar must provide for a varying number of title lines and bylines, and for changes in spacing caused by the ascenders and descenders of the letters. To simplify the design process, each block grammar is constructed in several stages, in terms of syntactic attributes extracted from profile features.[7]

Syntactic attributes. The first stage of a block grammar operates on the ones and zeros of the block profile. Strings of ones or zeros are called *atoms*. Atoms are divided into classes according to their length. A string of alternating black and white atoms is a *molecule*. The class of a molecule depends on the number and kind of atoms it contains. Finally, molecules are transformed into *entities* depending on the order of their appearance. The words *atom*, *molecule*, and *entity* were chosen because they are not specific to a particular publication or subdivision. (See the sidebar on syntactic attributes.)

The syntactic attributes that determine the parse are the size and number of atoms within an entity, and the number and order of permissible occurrences of entities on a page. Table 1 shows the expected variation in the horizontal profile of a page fragment that includes the title and byline. The assignment of symbols into larger units is accomplished by rewriting rules or *productions*. These

Figure 22: A possible result of the X-Y Cut algorithm on a page taken from [51]. Here the entire page is cut vertically (red) and then each sub-region is cut horizontally (green). The splitting order from this point becomes rather complex but is color coded as follows: orange, yellow, blue, and pink. Notice that a single node may have more than two children, which is the case for sections with multiple paragraphs, columns, etc.

Run-Length Smoothing Algorithm (RLSA). It is typically unnecessary to perform processing on all pixels of the document image. For the top-down algorithms previously described, which use either maximal white space rectangles or projection profiles, the document image is usually reduced in size during a preprocessing stage. By reducing the size and complexity of the input image, both the efficiency and accuracy can be enhanced assuming that only insignificant data is reduced. For instance, when detecting entire columns of text, the spacing between individual characters, words, and lines is unnecessary. One way to reduce the amount of data is to use a run-length smoothing algorithm (RLSA) [63] which will be discussed further in the Bottom-up Physical Structure Analysis section. This method can merge characters into words, words into text lines, and text lines into paragraphs by "smearing" the text to join characters into blobs. This is done by inspecting white spaces between foreground pixels and, if their width is below some threshold, setting them to black.

Template Techniques. "Template" techniques which have been observed in the literature [64][65], are labeled as top-down even though they often rely on a combination of both logical and physical document structure analysis [47]. These methods require a significant amount of knowledge about the expected document structure on which they are trained and may not generalize well to new types of documents. An effective way in which document structure can be described is through the use of a Form Description Language (FDL) [64]. The basic concept of FDL is that both the logical and physical structures of a document can be described in terms of a set of rectangular regions. The FDL specifies how a document should be processed based upon various aspects of its physical layout. Systems which utilize an FDL typically operate on a limited assortment of document types, thus its use is very application specific.

Dengel et al. present a technique which they call "Discriminating Attribute Values in uncertain Object Sets (DAVOS) [65]. By "object sets", Dengel et al. are referring to sets of regions on a document image along with their appropriate logical labels. The attributes (geometric features) of these objects may not be limited to single values but could cover a range of possible values and are thus considered as "uncertain." The DAVOS system analyzes business letters and builds a decision tree where each level corresponds with an increasing level of document type specificity. The terminals on the tree specify the entire logical layout of the document. Just as

with FDL, the ability of the DAVOS system to generalize to new document types is limited. DAVOS was only tested on business letters and was evaluated against a bottom-up technique (which utilized merging of connected components) and shown to have similar but “more balanced” results (i.e. logical labeling errors were more distributed among the various labels).

Bottom-Up Physical Structure Analysis

While top-down approaches start with the complete document image, repeatedly splitting it into smaller regions, bottom-up approaches carry out the inverse operation. Starting with the document image's primitives (i.e. individual pixels, connected components, words, etc. depending upon the application) bottom-up techniques repeatedly merge smaller regions into larger ones. While allowing more flexibility over top-down techniques, bottom-up techniques often result in greedy over-segmentation of regions. Bottom-up physical layout analysis techniques all utilize connected component analysis and may also make use of Veronoi diagrams, run length smoothing, mathematical morphology, neural networks [66], as well as communication theory (Document Image Decoding) [67]. This section will briefly review work that has been done for bottom-up techniques, starting with a discussion of connected component analysis.

Connected Component Analysis. As discussed previously, connected components are sets of foreground pixels such that a four or eight-connected path exists between every pixel pair in the the set. While text usually consists of connected components with a relatively consistent size and spacing, graphics generally tend to consist of larger connected components with more sparsely distributed positions. By analyzing these spatial properties of connected components, it becomes possible to identify and group text and graphics separately. Connected component generation involves grouping all four or eight-connected foreground pixels together in the document image. The components are then grouped based upon their bounding box location. The output of a connected component (cc) generation algorithm is a list of cc's where each entry contains the bounding box coordinates, shape of the region, number of black pixels, an image of the region itself, etc. The cc's are typically sorted by their bounding box position, and can be then filtered based upon height and width to determine regions more likely to be text vs those which are more likely to be graphics [43].

2 Literature Review

An example of a bottom-up physical analysis technique is Bixler et al.'s text extraction algorithm [68]. Bixler demonstrates his algorithm by extracting and recognizing the text from a map as shown in Figure 23. His technique first uses a standard recursive (stack-based) flood fill algorithm in order to find the connected components [69]. After finding an initial starting foreground pixel, the flood fill algorithm can be described simply as follows: (1) If the current pixel is not foreground then return, (2) Set the current pixel to a replacement color in order to mark it as processed, store it in memory (3) Recurse to the function in each direction in turn (4) Return from the function. The aforementioned algorithm is then repeated for each unmarked foreground pixel of the image in turn, until all connected components are found and all marked pixels grouped into their constituent connected components are stored in memory, along with their bounding boxes, and any other relevant information.

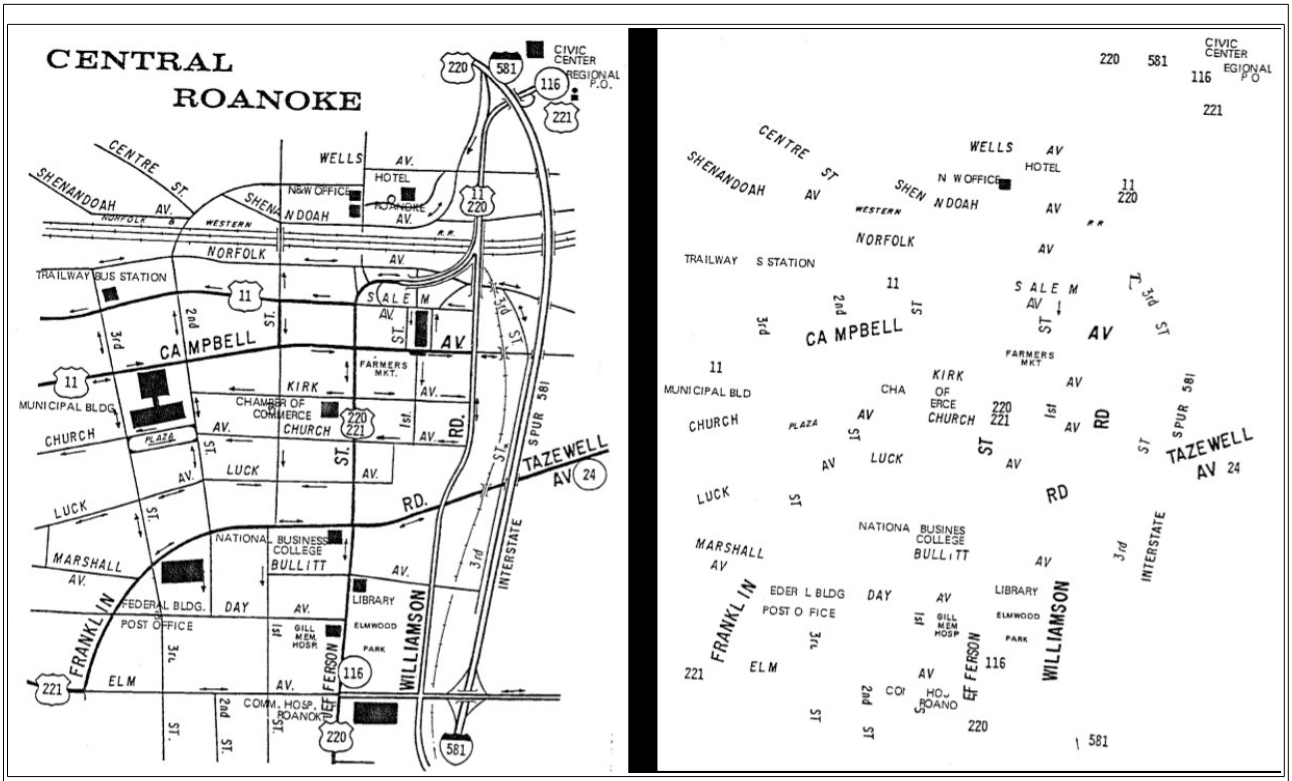


Figure 23: On the left is a map, and on the right is the map's extracted text. Notice there are some dependencies where the foreground text was confused with the imagery of the map. For instance one of the "f's" in the word "Post Office" is missing because it overlaps with a road [68].

With the connected components found, Bixler then determines which ones are text and which are graphics based on a simple height and width thresholding technique. Once the components have been segmented into text and graphics, those identified as graphics are subtracted from the image to leave only the text. The resolution of the image is then reduced based upon the size of the character components. A connected component tracking algorithm is then utilized in order to find words which could be potentially in any direction (i.e. vertical, diagonal, horizontal, etc). The algorithm scans the reduced document image from left to right, top to bottom looking for a starting connected component, then does a nearest neighbor search in each direction to find the closest character. Information about the spacing and direction between the first two characters is then utilized to track the location of the next character until entire words are detected. The procedure is repeated for each unique starting point until all words are found. The technique achieved near perfect results for a complex map, with only those words which significantly overlapped graphics being missed.

Document Spectrum Analysis ("Docstrum"). The Document Spectrum (Docstrum) proposed by O'Gorman [70], is a representation of a document which describes global structure features and can be useful for page analysis. The technique takes the document's connected components and utilizes a k-nearest-neighbor clustering technique in order to segment the document into words, text lines, paragraphs, etc. The algorithm recognizes five nearest neighbors for each connected component, where closeness is measured by Euclidean distance in the image. Each nearest neighbor pair is described by a 2-tuple, (d, θ), which is the distance and angle between the centroids of the two connected components. The "Docstrum" is the plot of (d, θ) for all nearest neighbor pairs in the image as illustrated by Figure 24. The text's spacing between characters and words as well as the line angles can be estimated by summing up the distance and angle values in the docstrum plot. The distances and angles are converted to respective histogram representations. The nearest neighbor angle histogram is smoothed and the peak found. The angle of the peak value gives a rough estimate of the text line orientation. This rough estimate is then used to determine intra-line and inter-line spacing by analyzing two histograms of the nearest-neighbor distance values. The first histogram is for intra-line spacing and filters out all distance values that are not within a tolerable range of the textline orientation estimate. This histogram thus represents the distribution of inter-character and word spacing within each text line. The second distance histogram filters out all

values outside of a tolerable range of the textline orientation estimate's perpendicular. This histogram, therefore, represents the distribution of the document's inter-line spacing.

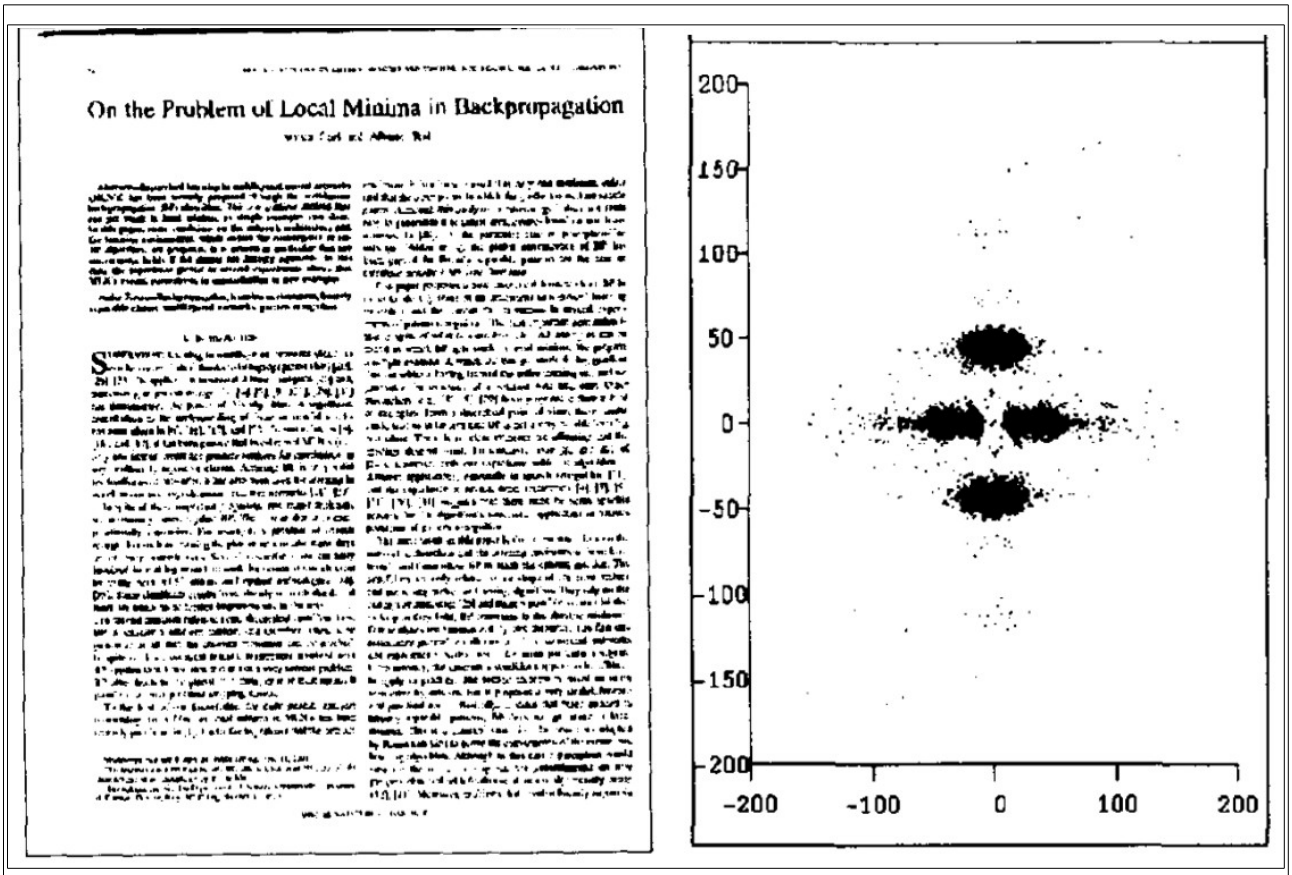


Figure 24: On the left is a document image and on the right is its corresponding Document Spectrum representation [70].

Nearest neighbors on each line are merged into words and then a regression fit is made to the centroids of the words in order to locate text lines. A straight line is fitted to the centroids in each group by minimizing the sum of square errors between centroids and the line. From these text lines a final estimate is made of the page's skew. An issue with this method is that text line descenders and noise could reduce the accuracy of the initial estimate and cause problems with reaching the right conclusions. It is important to have the correct threshold values and to smooth the histograms appropriately in order to get successful results. After the text lines are estimated, larger structures (like paragraphs or other text blocks) are then detected. The blocking technique examines pairs of text lines to determine whether or not they meet certain criteria to be considered part of the same text block. If the two lines are

approximately parallel, close enough in perpendicular distance, and/or horizontally overlap to some degree then they are said to meet the criteria of belonging to the same block.

One of the benefits of this algorithm is that it does not assume that each component of the document has the same skew angle. Thus it is possible to indiscriminately segment lines and/or blocks of text in any direction. This may be useful for a variety of circumstances including analysis of magazines or journals with sporadically appearing vertical text, scans of several credit cards or business cards each on the same page but at arbitrary angles, maps with text overlayed over imagery in arbitrary directions, etc. The technique was tested on hundreds of scanned journal pages, however no comprehensive performance evaluation is given.

Voronoi Diagram. Given a set of points and a subset of these points called sites (or generators), the Voronoi diagram is the partition of the entire set into convex cells, such that each cell is the region consisting of all points that are closer to a particular site than to any other. Voronoi diagrams are among the most fundamental and well-studied objects in computational geometry [71]. An *ordinary* Voronoi diagram, as illustrated by Figure 25 [72], is one which uses Euclidean distance as its metric and can be described as the set of Voronoi regions which correspond to the convex shapes created by the partition.

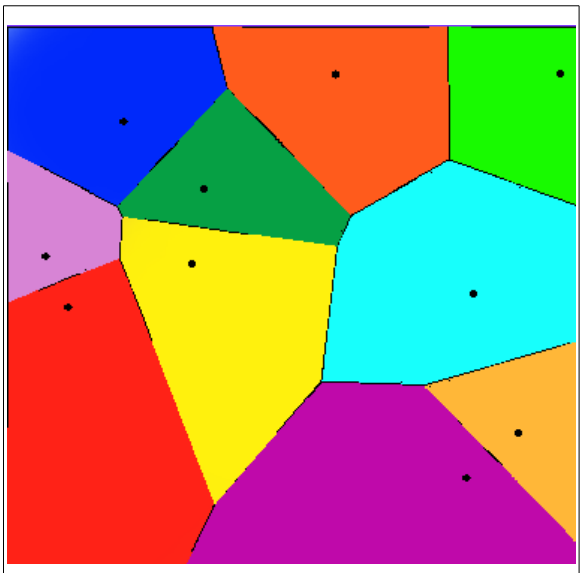


Figure 25: Illustration of an ordinary Voronoi diagram [72].

An *area* Voronoi diagram is a generalization of the *ordinary* Voronoi diagram depicted by Figure 25 which uses the Euclidean distance between the areas of connected components as a metric rather than the distance between points. The *area* Voronoi diagram for a document image can be found by the following procedure: (1) Sub-sample every connected component in the image such that all that remains is a subset of the points on the outter edge; (2) Generate an *ordinary* Voronoi diagram using this subset of points in the image; (3) Remove all edges of the Voronoi diagram which both belong to points of the same connected component. This process is illustrated by Figure 26 [73].

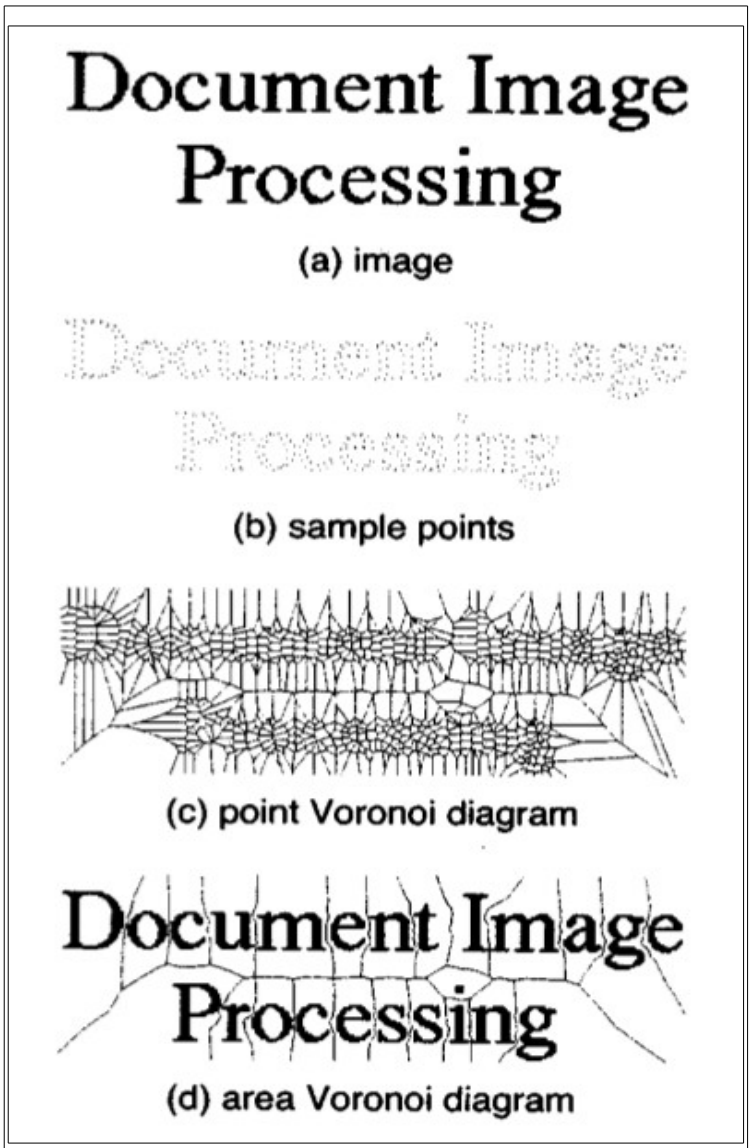


Figure 26: The process of segmenting an image into an area Voronoi diagram as illustrated by Kise et. al [73].

2 Literature Review

Kise et. al formulate the problem of physical page as that of determining which edges of a document's *area* Voronoi diagram best represent the boundaries of document components. By analyzing various features in the document image, superfluous edges of the area Voronoi diagram can be removed, thereby leaving only the edges corresponding to document boundaries. Superfluous edges would, for instance, correspond to the space between characters, words, text lines, etc. when a division of the page into separate paragraphs, columns, imagery, title, etc. is required. For each edge, all of its line segments are evaluated in order to determine the minimum distance between the two points on the connected component which were used to generate the Voronoi line segments in the first place. If this minimum distance is below a given threshold for any line segment of the edge, then the entire edge removed. Likewise, the area of connected components are divided by these edges are compared and if the distance between the connected components is small enough in relation to the area ratio of the two connected components, then the corresponding edge is removed.

Kise et. al evaluate their algorithm on 16 document images at two resolutions, 90 DPI and 300 DPI having a non-manhattan layout each at 4 different skew angles to test for robustness (thus a total of 128 with non-Manhattan layout when counting the resolutions and skew). In order to test the applicability with Manhattan layouts, the algorithm was also evaluated on 98 images from the University of Washington database (UW1) all at 300 DPI. In evaluating the algorithm on these datasets, the percentage of the “body” text, “auxiliary” text, and “non-text” document zones which were over and under fragmented is evaluated respectively. The algorithm performed best on the body text of the non-Manhattan documents scanned at higher resolution where only 2.1% of zones were over-fragmented and only .4% under-fragmented. The algorithm faired poorly for the segmentation of non-text zones of all document types, but especially poorly for Manhattan documents where it resulted in a 98% over-fragmentation rate.

Run Length Smearing Algorithm (RLSA). Proposed originally in 1974 by Johnston [74] in order to separate text blocks from graphics, the Run Length Smearing Algorithm (RLSA) has been frequently used to obtain basic features for document analysis [43]. RLSA, in its most basic form, transforms a binary image as follows: (1) For each background pixel, if the number of neighboring foreground pixels is above a certain threshold, then the pixel is changed to foreground; (2) All foreground pixels are left unchanged. When applied horizontally or vertically to the rows or columns of an

image respectively, RLSA has the effect of linking together neighboring background pixels that are separated by a number of pixels below the given threshold (illustrated by Figure 27 [63]). With an appropriate choice of threshold, it is possible for the linked areas to correspond with separate document zones. The threshold is typically set based upon the character height, gap between words, and interline spacing [43].

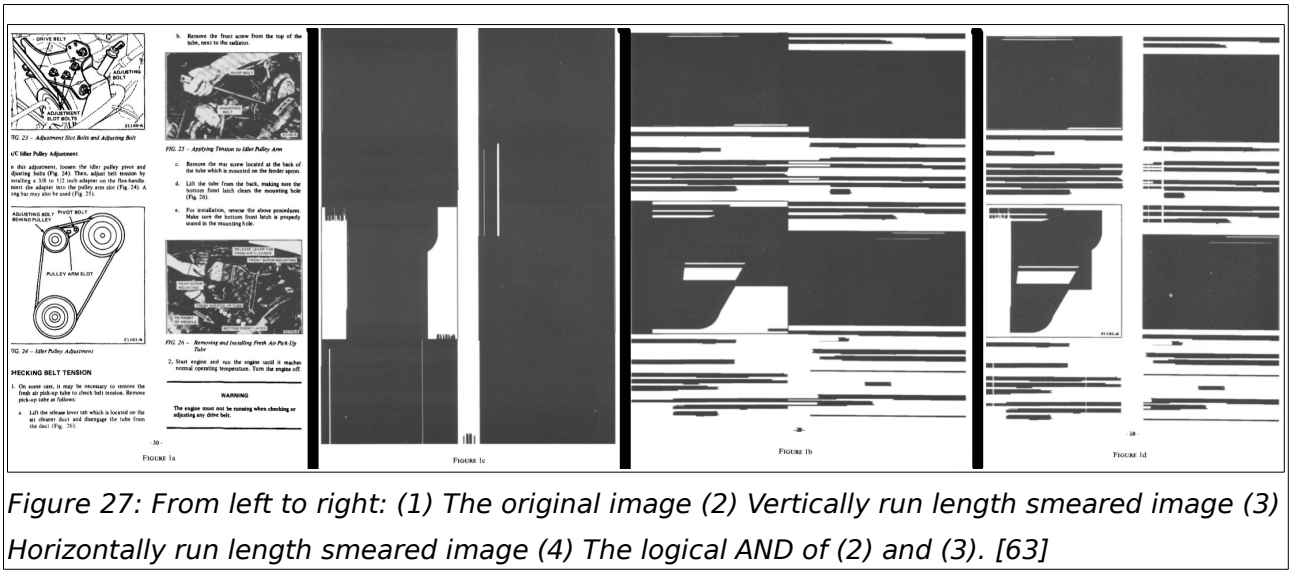


Figure 27: From left to right: (1) The original image (2) Vertically run length smeared image (3) Horizontally run length smeared image (4) The logical AND of (2) and (3). [63]

Multiresolution Morophology. Bloomberg [75] discusses an approach to document image analysis which uses morphological operations at multiple resolutions. For an in-depth overview of morphological image processing (including definitions for terms such as structuring element, dilation, erosion, opening, and closing) the reader is referred to [76]. While connected component based techniques are effective on pages with only characters, they can exhibit practically unbound time and memory requirements when presented with pages consisting of halftones, graphics, and/or handwritten notes. While, in connected component analysis, a region's shape is primarily dictated by the configuration of its “ON” pixels, Bloomberg considers a region's shape based upon relationships between adjacent “ON” and “OFF” pixels and then considers texture to be the statistical distribution of such shapes in an image. Bloomberg describes a morphological image processing operation called the “generalized opening,” based upon the Hit-or-miss transformation [77], which is useful for localizing shapes and textures of interest within an image. The rationale behind carrying out operations at multiple resolutions is that a single document image typically will contain shapes and textures of various sizes. While it may be more advantageous to process smaller regions at a higher resolution, larger regions

generally require only a course view (lower resolution). Multiresolution image processing exploits such size differences such that regions can be processed in their most appropriate resolutions. Bloomberg describes a solution to the problem of half-tone segmentation which closes the image (performs a dilation followed by an erosion) with a large structural element, followed by an opening (erosion followed by dilation) in order to only keep the half-tone regions while removing the text ones. Since using large structural elements on a high resolution image ends up being very costly, Bloomberg instead carries out the same functionality by carrying out a cascade of openings and closings using a 2x2 structural element while subsampling the image in between these operations.

Bloomberg further goes on to illustrate how multiresolution morphology can be used for the detection of italicized words (and also a separate technique for detecting bold words). To detect italics, Bloomberg looks for edges inclined at about 12° from the vertical. A 6x13 structural element is used which consists of 4 “OFF” pixels on top of one another aligned at an approximately 12° angle followed by 4 “ON” pixels in the same configuration. The aforementioned pixel sequences are separated by “don’t care” pixels in between and to the sides as shown in Figure 28.

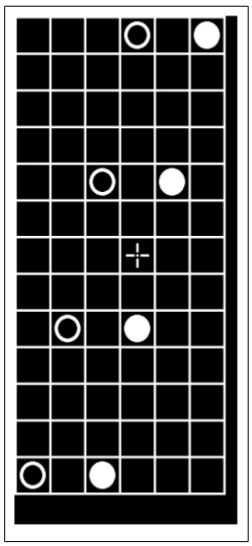


Figure 28: 6x13 structural element used by Bloomberg [75] in detecting italics. The open circles on the left represent “OFF” pixels, the closed circles represent “ON” pixels, and the empty squares are “don’t care” pixels.

As illustrated by Figure 29 (a), the “generalized opening” operation (hit or miss transform followed by dilation) is carried out on the image using the structure element shown in Figure 28 in order to find which regions of the text contain italics. The resulting “ON” pixels of the generalized opening are considered the “seed” pixels for italicized words. A closing operation followed by an opening as carried out on the seed pixels in order to, respectively, merge the correct seed pixels, and then get rid of noise. After performing a small vertical dilation on the result, the final seed image, Figure 29 (b), is ready. Next, a word mask, Figure 29 (c), is created by sub-sampling the image by a factor of 4 and using a cascade of openings and closings as was done for the half-tone segmentation problem, followed by a small horizontal dilation. The final italics selection mask, Figure 29 (d), is then created by keeping only those words in the word mask which overlap the seed pixels in the final seed image, thus resulting in a mask which only keeps the italicized words in the image.

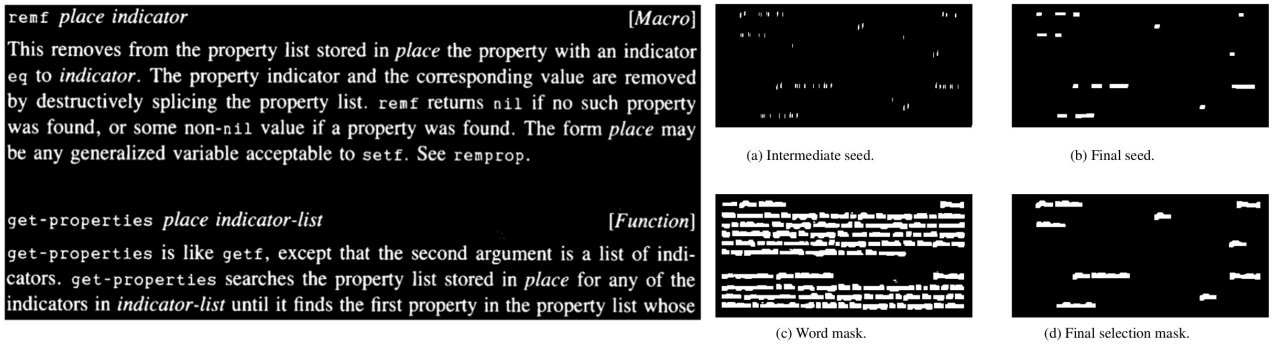


Figure 29: Bloomberg’s italics detection [75]. Original text is on the left and on the right is (a) the intermediate seed image, (b) the final seed image, (c) the word mask, and (d) the final italic word selection mask for the image.

Hybrid Physical Structure Analysis

Hybrid structure analysis can simply be regarded as any mixture of the top-down and bottom-up approaches previously described. Several of the hybrid segmentation algorithms found in the literature utilize a combination of “splitting and merging” strategies [60] [78] [79]. Such algorithms will begin by carrying out a top-down methodology in order to first split the page into regions which appear homogeneous, usually based upon horizontal or vertical projection profile measurements. Liu et al. [78], for instance, utilize an algorithm which operates as

follows: If a region is found to be in-homogeneous based upon certain critieria then it is split into 4 rectangular sub-regions using an adaptive thresholding technique to choose the line positions. If two regions within the previously split region are then found to be homogeneous based upon the same criteria then they are merged. This is continued recursively until there are no more splits and merges to be made. There are also various hybrid techniques which operate based upon measuring the space between text areas, referred to as "maximal white space calculation." Such techniques such as those of Okamoto et al. [80] and Bruel [81] are derived from Baird's "Shape-directed Cover Algorithm" [82]. This subsection will first discuss Baird's shape directed cover algorithm as well as a newer algorithm which is in the same spirit. Two notable open source systems which perform document layout analysis are Tesseract [57] and OCR-Opus [83]. Both systems utilize a combination of bottom-up and top-down physical layout analysis techniques and will briefly be discussed as well.

Shape-directed Cover Algorithm. In an attempt to combine the strengths of top-down and bottom-up methods (i.e. faster run time for the more greedy bottom up methods but more global knowledge for top-down), Baird et al. [82] proposed a "global-to-local" strategy which first finds the rectangular coordinates of all foreground connected components and then finds all of the maximal white space rectangles surrounding them. A white space rectangle is considered maximal if it contains only white pixels and cannot be further expanded while staying entirely white. The white space rectangles are then sorted into a binary tree structure where the right-most white space rectangles are at the root, and the left-most are the leaves. Multi-way branches which occur when there is more than one maximal white space rectangle at a given X coordinate, are handled using singly linked lists as entries in the binary tree. Unlike most of the previous top-down physical layout analysis research, Baird focuses intently on algorithmic complexity. When he denotes the number of maximal white space rectangles as m and the number of foreground rectangles as n, he found his algorithmic complexity for sorting the white space rectangles to be $O(nlogn + m)$.

Once the rectangles are sorted, a subset of these rectangles denoted as the "cover set" is chosen. Any regions of the image not covered by the union of this cover set will define the segmented text blocks. In order to speed up processing time, the rectangles in the cover set are chosen in a greedy fashion using the high level information available in the binary tree. In terms of processing speed, this can prove advantageous over the X-Y Cut algorithm which uses extensive backtracking. The

cover space is chosen based upon domain specific information. For instance, in Manhattan layouts, the white space rectangles between columns will typically have a high (but not too high) aspect ratio. Baird et al. thus assigns shape scores to the rectangles in order to favor the most significant and choose the cover space based upon these scores. Experiments were run on over 100 Manhattan layouts which included typewritten and printed pages from letters, magazines, books, journals, and newspapers, which included complex layouts consisting of headers, footers, embedded mathematical equations, graphs, multiple columns, etc. The authors reported near perfect results for large column structures but would observe errors for smaller blocks of text especially in the presence of noise.

White space cover algorithm by Breuel. Breuel presents a variation of Baird's white space analysis algorithm which is simpler to implement (requires less than 100 lines of java code) [81]. The algorithm starts by picking one of the black rectangles, called the “pivot”, toward the center of the image. Since the maximal white rectangle cannot contain the pivot, there are now four distinct possibilities for the maximal rectangle's location: above, below, to the right, and to the left of the pivot. Each sub-rectangle is then evaluated using a quality measure to determine which is most likely to contain the maximal rectangle. After the sub-rectangles and their respective quality measures are inserted into a priority queue, the above steps are repeated. This process continues until a fully white-space region is detected. The rectangle corresponding to this region is the optimal solution. The results of this algorithm were described as favorable when run on the same dataset as Baird (the UW3 Database [84]), with no errors observed on 223 pages. An in-depth evaluation, however, was not provided. Figure 30 illustrates a more recent technique called “the White Space Cuts Algorithm,” [85] which combines Baird and Bruel’s approaches.

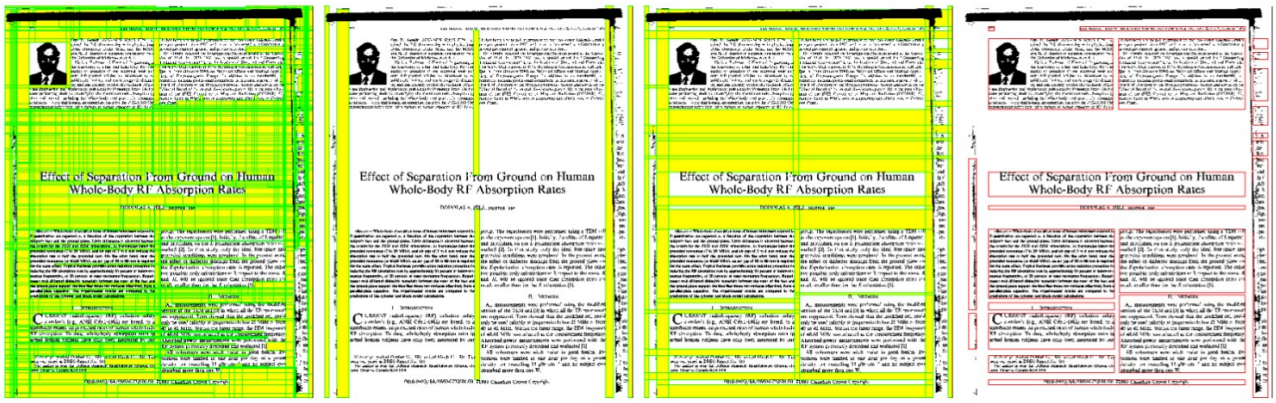


Figure: 30. An example image illustrating different steps of the whitespace-cuts algorithm [85]. Left to right: whitespace cover of the page background, extracted vertical separators and borders, extracted horizontal separators, extracted page segments.

OCRopus Open Source OCR System. Bruel, who was discussed previously for his novel variation of Baird's white space cover technique, is the project lead for OCRopus, an open source Google-sponsored project which addresses various problems in Document Analysis through the use of large scale machine learning. All of the project's modules are written in Python and the project emphasizes modularity, easy extensibility, and reuse. OCRopus is aimed at both the research community as well as large scale commercial document conversions [83]. The system includes overridable modules including, but not limited to, noise removal, skew detection, text/image segmentation, layout analysis, textline recognition, optical character recognition, and statistical language modeling. For more details on the architecture as well as algorithms implemented by this system, the reader is referred to [86].

Tesseract Layout Analysis Module. Tesseract, another Google-sponsored project as described earlier, utilizes a hybrid page layout analysis algorithm [57] which starts by utilizing bottom-up techniques in order to locate "tab-stops" on each text line. These "tab-stops" can represent the left and right edges of columns at that particular vertical location of the page. Each left and right tab stop is connected to form a "column partition," a vertical slice of a column at the given text line. A "column partition set" is the group of all "column partitions" at a vertical position of the page (i.e. stretching from the left of the page to the right). The column partition sets are then iterated in order to derive the column structure that makes the most sense for the entire page. Once the column structure of the page has been derived, this structure is applied in a top-down fashion in order to derive the page's reading order.

Tesseract's physical layout analysis module is used in this work to gain an initial estimate of the document's physical structure and will therefore be discussed in further detail in a later section.

Document Logical Structure Analysis

While the physical layout analysis step of a document analysis system generally divides an image into areas of text and non-text while determining an initial estimate of the page's basic columnar structure, the logical layout analysis step will further investigate the resulting structure in order to determine where splits or merges may need to be made based upon the perceived syntactic meaning of the document's components. The document analysis system may then iteratively transition back and forth from physical to logical analysis based upon further document understanding until some criteria is met. Once all of the document's components have been fully classified and segmented, the result of the logical analysis will be an increased understanding of the page's components. The page may, for instance, be composed of the chapter name at the very top, a page number to the top right, and several columns of text. The columns may consist of imagery, half tones, mathematical equations, block quotations, as well as various other components.

In the literature, document logical layout analysis research can be most broadly split into three main categories: (1) type-specific detection, (2) zone classification, and (3) page classification [87]. Type-specific logical layout analysis techniques, which are the primary focus of this thesis, emphasize the use of separate algorithms to detect possible components (i.e. text, image, math, half-tone, chemical equation, etc.) and make no assumptions about whether or not these components have already been correctly segmented. The input to a type-specific logical analysis algorithm may, for instance, consist of a single block which should really be logically separated into multiple separate blocks, and/or various blocks which actually need to be merged. Zone classification techniques, on the other hand, assume that the document has already been properly segmented into type specific zones and all that needs to be determined is what these types are. Such techniques will extract features from the zones and then use these in order to directly classify the zone type as one of a finite set of types (i.e. normal text, mathematics, imagery, etc). Page classification layout analysis research is geared toward classifying an entire page based upon the type of its content. A page may, for instance, be categorized as a title page, table of contents, appendix, glossary, regular page, etc.

2 Literature Review

Although page classification is an important component of document understanding, such a region-wide generalization should only be made after an in-depth analysis of the page is carried out to gain an understanding of all of its zones. It would not make sense to segment the page into logical zones based upon an ill-fitted estimate of what contents the page is expected to have. Likewise, zone classification may be an important step in logical layout analysis but it misses the important point that no physical layout analysis technique done prior to logical analysis is perfect. A full understanding of the under and over segmentations made by the initial physical layout analysis algorithm may rely upon the type of content in question. If, for instance, a table element is detected by the logical analysis module, it may make the most sense to then check for oversegmentations that could have been made by the physical analysis module. All of the elements that are clearly part of a larger table should be merged such that the table is then segmented appropriately. Likewise, there may be a block of text which has actually been undersegmented by the physical analysis module. An example of an undersegmentation is seen when there are inline mathematical expressions within a paragraph of text. While the entire paragraph may have been correctly segmented by the physical layout module, it is very important for the mathematical expressions within it to be segmented from the normal text in order to avoid subsequent recognition errors. In such an instance, it is then the logical layout analysis module’s job to detect candidate mathematical expression regions and then utilize the appropriate physical layout techniques necessary to segment them from the normal text.

Since proper physical segmentation and page classification are dependent upon the type of content in question, type-specific detection is the primary focus of this thesis. After a brief overview of some of the page and zone classification techniques found in the literature, type-specific detection techniques, the focal point of this work, will be discussed in greater detail. The focus will be primarily on type-specific detection of mathematical expressions. Detection of other zones such as tables, logos, and music scores will also be briefly discussed.

Page Classification

The motivation for page classification research is two-fold. Firstly, it is important in facilitating faster document processing. If a page's specific type can be known then the corresponding type-specific layout analysis techniques may be employed to reduce processing time. Secondly it is important to facilitate faster indexing of page

types. For instance if the title page is known, it will be very fast to do a document wide search to find the author of the work. When a user is searching for a specific document, knowing the class under which the document resides may allow for a quicker and more fruitful query experience. Page classification techniques found in the literature utilize a wide range of feature and classifier types in order to categorize an entire page. Although some methods utilize the output of a commercial OCR engine [88], the majority of techniques only use features taken directly from the document image [89] [90]. Page classification can be carried out at various stages in the document layout analysis process. The stage at which a final decision regarding the page type is made may vary based upon the type of document fed into the system, it's physical and logical layout, etc. For an in-depth overview of how the problem of page classification has been previously approached, the reader is referred to [91].

Zone Classification

Zone classification techniques are employed in order to logically label regions independently of physical segmentation. Such techniques operate under the assumption that whatever physical layout technique was carried out prior to logical analysis has already properly segmented the page into logically independent zones (i.e. normal text, equation, table, image, etc). In the literature these techniques vary based upon feature extraction methods, classification techniques, and the number of zone types to choose from. While earlier works [92] [93] may only distinguish between 2-3 zone types, more recent work [94] [95] is observed to distinguish a wider variety of zone types (i.e. 9-10).

The work done in [93] employs features based on the zone's spatial distribution of pixels to train a binary decision tree classifier that distinguishes text from non-text. Fan et al. [92] use pixel density features to first segment text from non-text and then use a “pixel connectivity histogram” in order to classify nontext as either photographic imagery or vectorized graphics. The pixel connectivity histogram takes into account every foreground pixel of the given zone, measures the number of other foreground pixels connected to it (the connectivity measurement), and gives the number of such foreground pixels that fall under each connectivity measurement found. The classification algorithm utilized was not specified by this work. Wang et al. [94] use a 25 dimensional vector composed of run-length, spatial, and background features in order to train an optimized decision tree to classify a zone as one of nine zone types. Abd-Almageed et al. [95] extract features of the zones based on the run-length and

spatial distribution of foreground pixels. The partial least squares algorithm is then carried out on these features in order to reduce their dimensionality. A novel hybrid classification approach which combines the benefits of a one-against-all classification scheme with those of a one-against-one scheme is used to determine the zone type from the reduced feature space. An SVM is used as the underlying binary classifier. Zones are classified into one of 10 logical types (chemical drawing, small text and symbols, drawing, halftone, logo or seal, map, math, ruling, table and large text). Both [94] and [95] are evaluated on the University of Washington III (UWIII) data-set [84].

Type-specific Classification

Type specific classification techniques make no assumption about the accuracy of any physical segmentation carried out prior to logical analysis. Thus a type specific technique will, not only detect the type of a given region, but also choose what further physical layout analysis may be required in order to ensure that the given region is properly segmented (i.e. table regions, for instance, may require a different segmentation technique than what would be required to segment normal text regions). The primary focus for this thesis is in type-specific classification: specifically, the proper detection and segmentation of mathematical equations. After an in depth overview of mathematical equation detection and segmentation techniques found in the literature, type-specific classification of other types will also be very briefly discussed.

Mathematical Expression Detection

Only a dozen independent studies which included the type-specific segmentation of mathematical expressions from document images were observed in the literature [96][97][98][99][100][101][102][103][104][105][106][107]. Although the research of math detection in document images may be largely uncharted, it is no mystery that there are a wide variety of mathematical expressions prevalent in text books, journals, and technical papers. Such documents are often desired to be viewed through portable devices, desktop computers and/or screen reading software for general convenience as well as assistive technology purposes. Even for “digitally-born” PDF documents, it is rare that mathematical regions of text end up being properly viewable by most software. For scanned documents this problem is even worse. Most commercial OCR modules employed typically have no understanding of mathematical expressions and become confused by their presence. A common result of the presence of mathematical expressions during OCR is garbled output, not only of

the mathematical expression regions, but even nearby normal text regions which would otherwise be recognized correctly. One of the early motivations for mathematical expression detection was, not necessarily to properly recognize these regions, but rather to ensure that their presence does not reduce the accuracy of normal commercial OCR software [102].

While there has been a relatively small amount of work found in the literature geared toward the logical and physical segmentation of mathematical regions, the OCR of these regions is a relatively mature field of study [108][109]. In mathematical OCR, however, it is typically assumed that all regions are perfectly segmented prior to recognition, either manually or automatically. In the literature, these regions are most often manually segmented prior to evaluation such that the recognition problem is evaluated independently of the segmentation problem. The only commercial mathematical OCR software found in the literature to date, “Infty” [110], implements an expression detection module, however provides no detailed evaluation of it [98]. Their most thorough evaluations are thus carried out on regions which were manually segmented beforehand. Since the type-specific logical and physical layout analysis of mathematical expressions is a largely uncharted area of study while mathematical OCR has been studied extensively, the primary focus of this thesis is on layout analysis rather than recognition. While mathematical OCR is certainly a very important processing step, it is outside of the scope of this work. For a recent in-depth literature survey on mathematical OCR the reader is referred to [109].

Before discussing the existing literature on expression detection, it is important to first specify some common notation and practices. Mathematical expressions found in printed text can either be located on a separate line from normal text or be mixed in with the text. Expressions falling under the first category are commonly termed as either displayed/isolated by the literature while those which fall under the second are termed as embbeded/inline. Expression detection techniques most often operate in several steps referred to as passes. The first pass often consists of locating initial expression candidates referred to as seed regions. These regions are then either removed or grown in further passes based upon various heuristics. The term, “digital-born document” refers to a document which was created directly from a computer rather than being scanned in. The dozen independent studies found in the literature each consist of either one or more conference or journal papers and will be briefly reviewed in the chronological order of their first publication.

Lee et al. [97] (1995). While this study is primarily geared towards expression recognition, the system developed also included an expression detection module. Bayes decision rules are employed in order to locate displayed expression regions. If a displayed region is detected then the entire line on which it resides is labeled as an expression region. No detailed analysis of the detection accuracy is provided. Similar work was also later carried out in [111].

Inoue et al. [98] (1998). An early study carried out by the authors of the "Infty" commercial OCR software, this work explains the software's underlying expression segmentation module. The module recognizes normal text and segments it from mathematical text in the same step by using information obtained from a commercial OCR engine along with a dynamic programming algorithm. Experiments are run on 50 pages of Japanese text, of which, detection errors are reported to have occurred on every page. No thorough evaluation is provided.

Fateman [99] (1999). The technique described operates in three separate passes and includes an interactive system which allows the user to manually correct any segmentation errors which may have been made. During the first pass, each connected component in the image is separated into one of two "bags": one for normal text, the other for mathematical text. The math bag initially contains all italicized letters, roman digits, punctuation, special symbols, and horizontal lines. These are considered as the seed regions. The second pass will then group the math-bag components into zones and, according to horizontal and vertical proximity, grow the seed regions by relabeling nearby text components as belonging to the math bag. On the third pass, remaining punctuation connected components in the math bag that are still isolated are moved to the text bag. Remaining isolated greek letters, roman numbers, etc are kept in the math bag. Text components that are close in proximity to the math bag, and could be considered as math such as "sin", "cos", etc. are moved to the math bag. Finally, the results are then shown to a human for interactive editing and correction.

Toumit et al. [100] (1999). A specialized top-down physical segmentation technique operating on the entire image with image reductions to segment math regions is briefly described but no specific details are provided. Displayed expressions are located under the assumption that they are always centered and on their own line. No further specific details are given. Embedded expressions are located by first finding special characters (i.e. "+", "=", ">", etc) and propagating around these using rules

specific to the given symbol. Various concepts and heuristics are defined and utilized for mathematical expression detection, primarily: atomic structures are single mathematical symbols, composite structures are logical groupings of atomic structures; implicit structures have no graphical representation (i.e. the multiplication operator when representing the multiplication of a with b as simply ab). To represent mathematics, a tree structure is used which allows each node to have more than two children. It is argued that mathematics is not inherently a binary recursive data structure, but simply a recursive one. While a+b+c can be represented by a binary tree, matrices, integrals, and vectors cannot necessarily be represented in this way. No detailed evaluation was carried out in this work.

Garain et al. [101] (2000). Four relavent works from Garain and his advisor, Chaudhuri, will be herein briefly reviewed. Garain's earliest technique first segments all text lines (this includes those of displayed expressions) by measuring horizontal projection profiles and denoting the boundary between two lines as local minima of these profiles. Next, each text line is separated into its constituent connected components. The mean and standard deviation is calculated for the distance of the bottom of the text line to the bottom of each connect component. Since math expressions may contain elements whose distance from the baseline varies more than normal text, this metric can, in some cases, be very helpful. If the standard deviation is above a predefined threshold then the line is expected to contain an expression. If such a line is also observed to be vertically separated from normal text lines significantly then it is labeled as a displayed expression region.

Embedded regions are then found by first looking at remaining normal text lines to find mathematical characters (i.e. “+”, “=”, etc.). When such characters are found, they are considered as seeds for the expression region and are recursively merged with their neighbors based upon the following criteria: (1) if the seed region is just a binary operator then the immediate left and right “words” are merged with the seed operator, (2) “words” adjacent to the seed region on the immediate left and right are merged if they contain one or more mathematical symbols, superscrips/subscripts, single dots/ellipsis, or numbers. Their detection algorithm is tested on 120 pages containing a total of 140 mathematical expression zones. Of these 120 pages, 20 are taken from the UWIII dataset. 132/140 expression zones were properly detected, with eight of them being entirely missed and three entirely false detections. No partially

correct detections are presented in the results (i.e. either the detection is completely correct or completely wrong based upon their evaluation technique).

In 2003 a morphological technique was proposed by Chowdhury [112] in order to segment displayed expression regions from all other regions in the text. A morphological approach is first carried out to segment table, text, and graphic zones. Further segmentation is then carried out on the result in order to find the displayed expression regions. Such regions are categorized into the three basic types: those which contain large horizontal lines, those which contain a long vertical separator (as is often seen in matrices or determinants, and all others. In order to segment the regions which fall under the “all others” category, the page is first closed with a horizontal structuring element in order to locate the text lines. Subscripts and superscripts (localized based on proximity and size relationship) are associated with their text lines. Features such as number of subscripts and superscripts, vertical overlaps, presence of tall symbols, and horizontal positioning of connected components are used by a decision tree classifier to locate the displayed regions. Similar techniques are used to localize regions containing horizontal or vertical lines. The techniques are evaluated on a set of 197 images. While roughly 97% of the displayed regions were reported to have been correctly segmented, no measurement was given of false positives. The technique was tested on embedded expression regions to have a 68% true positive rate, again no indication was given of false positive rate.

In 2004, Garain and Chaudhuri propose a technique for segmenting embedded expressions from document images [113]. N-grams are utilized in order to spot sentences output from a commercial OCR that are likely to contain embedded expressions. Sentences containing phrases like “such that”, “note that”, “denote”, etc. were shown to have a higher probability of containing embedded expressions than those which did not. A dataset containing 400 scanned pages of scientific documents including various science books, journals, conference proceedings, etc. is utilized by this study for evaluation [114]. Words recognized by the commercial OCR are evaluated as possible embedded expression candidates based upon the probability of their sentence to contain embedded expressions based on N-grams, the commercial OCR's confidence rating for constituent letters within the word in question, italic/bold/normal type style detection, inter-character spacing within the word in question, and variance of the bottom y coordinates for the constituent symbols within the word in question.

2 Literature Review

Experiments were carried out on the 400 scanned pages from the dataset which contained over 3000 embedded expressions. Evaluation includes a count of the true positives, false negatives, and false positives. Also included in the evaluation are partially recognized regions. The evaluation scheme strives to combine all of these measurements into one single metric/score. Partial recognitions are weighted based upon how many components were supposed to be identified as math in the region vs how many were actually identified[3]. The false negative count is weighted at zero for some reason, and thus does not even factor into the score. Based upon their scoring technique, ranging from 0 to 1, an average score of .963 was obtained for all 400 pages.

In 2009, Gerain experiments with methods for detecting both displayed and embedded expressions in document images [115]. Gerain approaches the problem by first extracting features for displayed and then embedded expressions from the document image and then experimenting with various averaging techniques on the respective features in order to see which has the best discrimination power for the given problem. Features are first extracted in order to classify an entire line of text as either being a displayed expression or not. In this study, the tendency of displayed expressions to also contain normal text seperators which don't belong to the expression (i.e. commas, periods, phrases like “and”, “therefore”, etc.) is not accounted for. The features used to detect displayed expressions include a measurement of the vertical space above and below the text line in question, the vertical scatter of the bottom y coordinates of the connected components on the line in question, the pixel height of the text line in question relative to the average pixel height of all the lines on the page, and the number of mathematical symbols on the text line. Each of these features are normalized to a value between 0 and 1 by using the (((need to write the math expression 1-e^(-featureval). The exponential allows for slight changes in the quantities being measured to have a large impact on the feature values.

After text lines are labeled as either displayed or normal, embedded expressions are then sought out for the remaining normal text lines. Individual words within sentences are classified as either displayed or normal based upon the following features. As in the previous study, linguistics is incorporated in order to detect

[3] This may prove problematic since components with a large number of pixels will be weighted just the same as components that are very small. For this reason, pixel-accurate methods of evaluation are utilized in this work, as will be explained in a later section.

sentences which are likely to contain embedded expressions. Other features incorporated for embedded expression detection include the commercial OCR confidence rating of each word of the sentence, the typestyle of the given word (i.e. italic, bold, etc), the scatterdness of the connected components within the word about the textline, and the average horizontal gap between characters within the the word in question. After normalizing these features, they are used to classify every word of a sentence as either normal text or embedded expression text[4]. The features for displayed text lines are each combined into a single scalar value through one of the following averaging techniques: arithmetic mean, geometric mean, harmonic mean, and weighted mean. Lines are chosen as displayed text or not by comparing the resulting feature value to a scalar feature value found empirically which varies depending upon the averaging technique used. A very similar approach is used to detect words segmented by the commercial OCR engine that are embedded expressions.

Experiments are carried out on 200 scanned pages. 150 of the pages were taken from Garrain's corpus [114] and the other 50 were taken from the INFTY database [116] which only contains manually segmented displayed expression images. Training to determine thresholds is carried out on 50 of the images, and evaluation carried out on the other 150. Tests using the weighted average method wherein the weights are determined through a gradient descent algorithm showed the best results. Using their specialized efficiency metric which takes into account false negative, false positives, true positives, and partial recognitions, [113] a score of 87% is achieved for embedded expression extraction while a score of 88% is achieved for displayed expression extraction.

Kacem et al. [102] (2001). As illustrated by Figure 31, the primary motivation of this work is, not to properly segment all mathematical regions for mathematical recognition purposes, but rather, to only segment those regions which may interfere with a normal commercial OCR engine (i.e. that could result in errors). The proposed technique identifies various mathematical symbols in the document without the use of any commercial OCR engine. These symbols include product, summation, integrals,

[4] Garrain relies on commercial OCR engines to segment his words within sentences, prior to classification. He makes no corrections to improper segmentations made by the commercial OCR. Thus in the situation where a word is improperly segmented by the commercial OCR as containing part of an expression and part of a normal text all in one block, Garrain's algorithm will always either result in false positive or false negatives.

roots, fraction bars, large brackets (ie that surround horizontally overlapping expressions on different lines), small delimiters (i.e. normal paranthesis/brackets), and binary operators such as plus, subtraction and equals. These symbols are identified based on a measurement of their connected component's aspect ratio, area, and pixel density. Using a training set which contains various appearances of these symbols in printed text, a histogram is created for all measurements in order to know their distribution. Based upon these histograms upper and lower bounds are set on these measurements for each symbol. When identifying a new connected component, a label is assigned to it based upon the intersection of all three measurements. Rather than doing an immediate binary label based upon this information, a “membership degree” is assigned to the connected component for each of the possible symbol types. If any of the membership degrees are within the upper and lower bounds, the symbol type with the highest degree is assigned to the connected component. This method was evaluated on a test database of 460 mathematical symbols and 95.3% of connected components were found to be well-labeled. It was not indicated whether or not there were false positives.

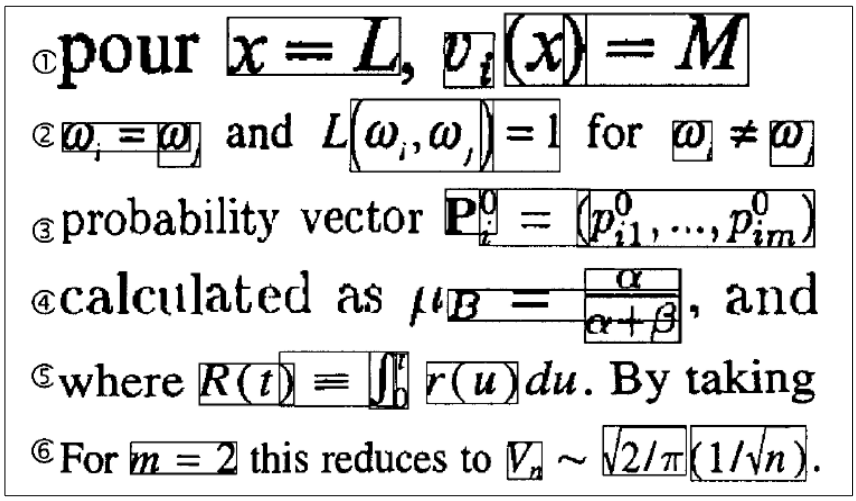


Figure 31: Regions detected as belonging to expressions are shown above bounded by rectangles. Note that, although most of the regions are over-segmented with various symbols being missed altogether, subtraction of the above labeled regions will result in improved accuracy for most commercial OCR engines which would otherwise be confused by the presence of the various mathematical expressions [102].

Once the connected components are labeled as indicated above, the text lines are determined by grouping all the connected components based upon proximity. The

specific algorithm used is not specified. Math symbols found within text lines are often used as heuristics to dictate whether or not lines should be merged. For instance, in the case of a large fraction bar, it is clear that there should be both a numerator and denominator. This may require vertically merging part of the two lines together. Once the lines are extracted, their aspect ratios and position in relation to other lines is measured in order to determine whether or not they are likely to be a displayed equation. Further measurements are then made on each connected component's vertical position within its corresponding line and its height in relation to the average connected component height for the line. For every line, all of the connected components are labeled as one of the following topography features: overflowing, ascending, descending, centered, high, or deep as illustrated by Figure 32.

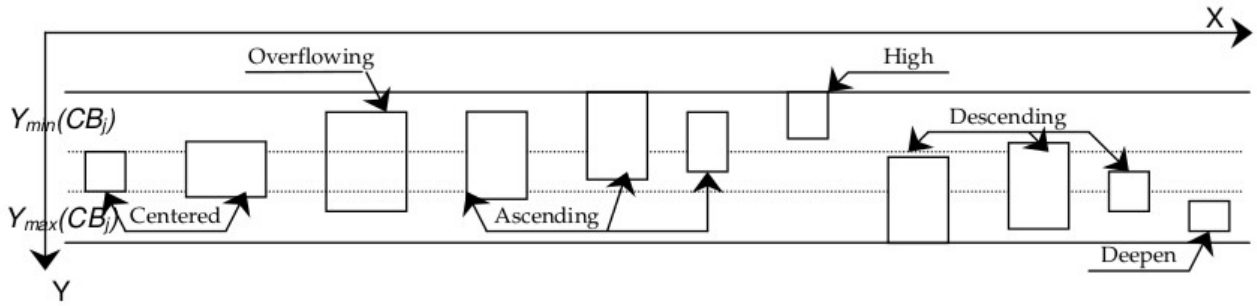
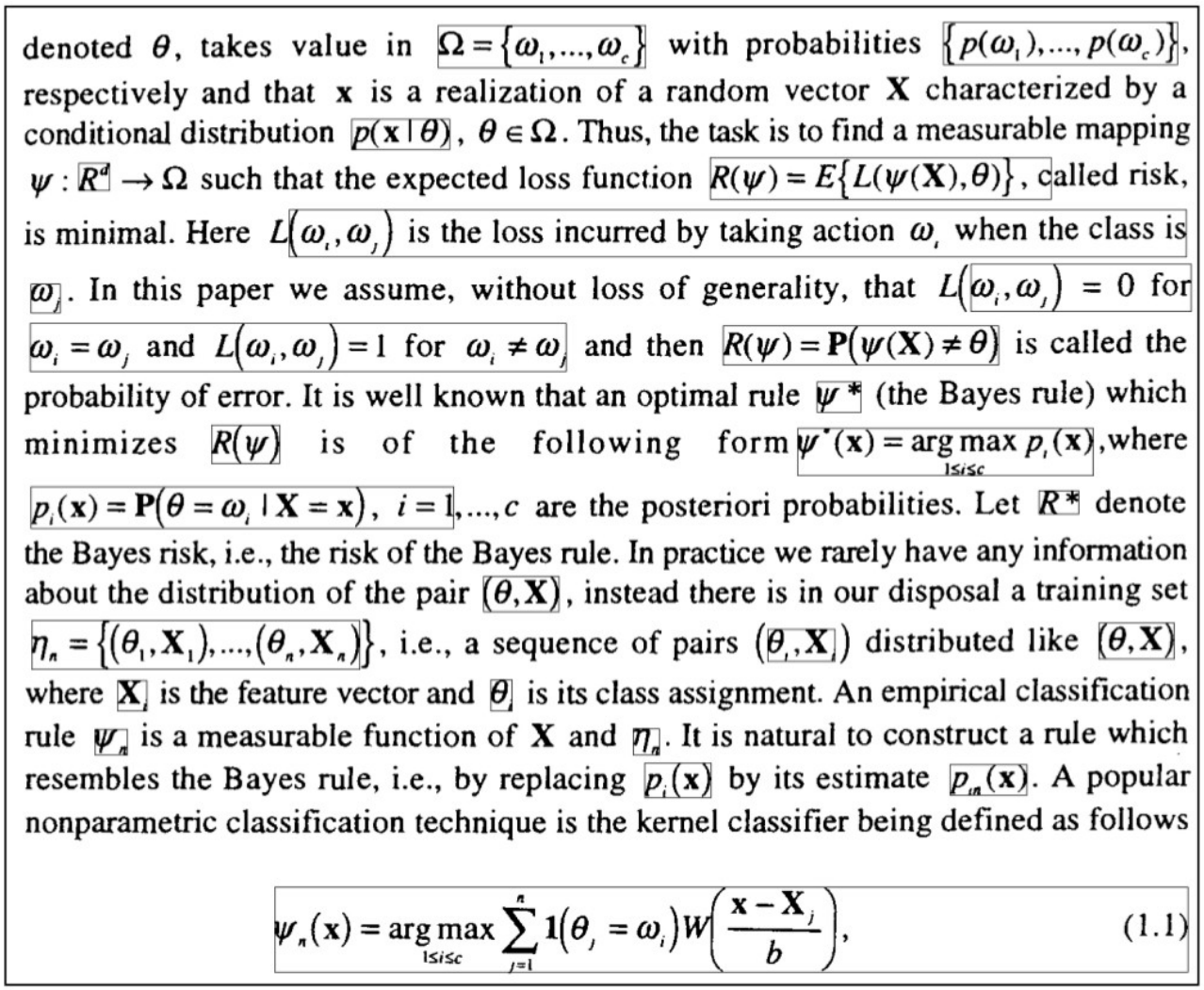


Figure 32: A connected component's possible topography features on text line j based upon vertical location in reference to line j's upper and lower central bands (Cbj) [102].

The topography features are used to help determine the type of a symbol. For instance subscripts would be descending or deep, while superscripts would be ascending or high. Radicals would be overflowing, and fraction bars would be centered. Subscripts and superscripts are found by comparing the relative size and position of two adjacent connected components. Training is carried out on these measurements using a histogram approach similar to the one described for math symbol identification. Next, rule-based context is propagated based upon the specific math symbols in question. For instance, each connected component inside a radical symbol. Since summations, products, or integrals are often accompanied by limits, these are sought out above and below such symbols.

The technique was evaluated on 100 pages with roughly 93% of the equations reported as being properly segmented. While the technique was reported to be fairly reliable for extracting displayed expression regions, it faced problems with embedded

expressions. Greek or italic symbols which should have been labeled as expressions were often ignored as illustrated by Figure 33.



denoted θ, takes value in $\Omega = \{\omega_1, ..., \omega_c\}$ with probabilities $\{p(\omega_1), ..., p(\omega_c)\}$, respectively and that $\mathbf{x}$ is a realization of a random vector $\mathbf{X}$ characterized by a conditional distribution $p(\mathbf{x} \mid \theta)$, $\theta \in \Omega$. Thus, the task is to find a measurable mapping $\psi : R^d \to \Omega$ such that the expected loss function $R(\psi) = E\{L(\psi(\mathbf{X}), \theta)\}$, called risk, is minimal. Here $L(\omega_i, \omega_j)$ is the loss incurred by taking action ω_i when the class is ω_j. In this paper we assume, without loss of generality, that $L(\omega_i, \omega_j) = 0$ for $\omega_i = \omega_j$ and $L(\omega_i, \omega_j) = 1$ for $\omega_i \neq \omega_j$ and then $R(\psi) = \mathbf{P}(\psi(\mathbf{X}) \neq \theta)$ is called the probability of error. It is well known that an optimal rule ψ^* (the Bayes rule) which minimizes $R(\psi)$ is of the following form $\psi^*(\mathbf{x}) = \arg\max_{1 \le i \le c} p_i(\mathbf{x})$, where $p_i(\mathbf{x}) = \mathbf{P}(\theta = \omega_i \mid \mathbf{X} = \mathbf{x})$, $i = 1, ..., c$ are the posteriori probabilities. Let R^* denote the Bayes risk, i.e., the risk of the Bayes rule. In practice we rarely have any information about the distribution of the pair $(\theta, \mathbf{X})$, instead there is in our disposal a training set $\eta_n = \{(\theta_1, \mathbf{X}_1), ..., (\theta_n, \mathbf{X}_n)\}$, i.e., a sequence of pairs $(\theta_i, \mathbf{X}_i)$ distributed like $(\theta, \mathbf{X})$, where $\mathbf{X}_i$ is the feature vector and θ_i is its class assignment. An empirical classification rule ψ_n is a measurable function of $\mathbf{X}$ and η_n. It is natural to construct a rule which resembles the Bayes rule, i.e., by replacing $p_i(\mathbf{x})$ by its estimate $p_{in}(\mathbf{x})$. A popular nonparametric classification technique is the kernel classifier being defined as follows

$$\psi_n(\mathbf{x}) = \arg\max_{1 \le i \le c} \sum_{j=1}^{n} \mathbf{1}(\theta_j = \omega_i) W\left(\frac{\mathbf{x} - \mathbf{X}_j}{b}\right), \qquad (1.1)$$

Figure 33: A result of Kacem's expression segmentation technique [102]. Note that the theta symbol is only segmented for the cases when context propagation dictates that it should be. When it is by itself it is missed entirely while when it is wrapped in a parenthesis or has a subscript it is segmented.

Jin et al. [103] (2003). Isolated expressions are extracted based on a Parzen window classifier and embedded expressions are extracted based on 2-D structure analysis and various heuristics. The technique is evaluated on a dataset consisting of 93 pages from technical journals. 10% of the pages in this set are used for training and the other 90% for evaluation. The results are reported as favorable, however no thorough evaluation or specific results are provided.

Drake and Baird [104] (2005). Drake and Baird utilize a technique based upon Kise's bottom-up Area Veronoi Diagram-based physical segmentation method [73]. For each line of text, the Area Veronoi Diagram is calculated and then each vertex and edge is classified as either normal text or mathematical. All of the results for the line are then combined in order to classify the line as normal text or as a displayed expression. A strength of this technique is that the veronoi diagram is invariant to skew of the page. Thus an expression could be detected by this technique regardless of its angle in reference to the rest of the page. The input to both training and evaluation are images of isolated text lines which were cut out of page images manually or synthesized in isolation using Latex. The dataset contains roughly 4,400 connected components labeled as math, with about half used for training and the other half used for testing. The Veronoi Diagram's edges were also labeled in the dataset, with roughly 4,000 used for both training and testing purposes respectively. From reviewing the confusion matrices provided it was found that the true positive rate for math connected component detection was 88% and false positive rate was ~7%. The algorithm was not tested on any lines that contained a mixture of math and normal text.

Tian et al. [105] (2005). Tian et al. propose a technique aimed at segmented both displayed and embedded expressions. Displayed expressions are found by calculating the average y distance of the center of all connected components on a line from the line's center. If this measure is above an empirically determined threshold, then the line is declared as a displayed expression candidate. To confirm whether or not the candidate is truly a displayed expression region, the line's connected components are run through a recognizer specifically designed from mathematical symbols. If any mathematical symbols are found then the text line is confirmed to be a displayed expression region. Embedded formulas are found by analyzing the spatial orientations of connected components on the text line, recognizing mathematical symbols, and employing propagation rules based upon these symbols. The technique is evaluated on more than 100 pages of technical documents to achieve a true positive rate of 95.19% for displayed expressions and 90.12% for embedded expressions. False positive rates are not reported.

Yamazaki et al. [106] (2011). Yamazaki et al. describe a technique which they have integrated into OCRopus [83]. The technique only detects displayed expressions and uses features very similar to those proposed by Garain. Also included are the

following features: standard deviation of symbol aspect ratio within a text line and left indentation measurement. Rather than using the averaging techniques employed by Garain, a SVM is used. The system is tested on an unspecified number of pages containing 542 displayed expressions, of which, 531 are identified correctly. No further evaluation is provided.

Lin and Baker et al. [107] (2012). X. Lin from Peking University and J. Baker from University of Birmingham collaborated at this year's 12th International Conference on Document Analysis and Recognition, in developing novel techniques for expression segmentation in digitally-born PDF documents [117][118]. In 2012, Lin proposed a technique for segmenting embedded expressions in digitally-born PDF documents (the displayed expressions segmentation is not evaluated in this work). Since the documents used in this study are digitally-born it is assumed that all typesetting information and text is available within the PDF's on which the experiments are run. In order to detect embedded expressions, text lines are evaluated each in turn (no OCR is required). First the words on the text line are segmented by using an adaptive thresholding technique on the PDF's image. A histogram is created which gives the frequency of horizontal gap lengths throughout the line. The second most frequent gap length is used for determining the word gap threshold (the first most frequently occurring gap is typically the distance between individual characters within words). Characters such as parenthesis, equals signs, sums, etc. are segmented as words regardless of their left and right horizontal gaps, assuming that their unicode is available within in the PDF.

Once the words are segmented, 12 features are calculated for each individual word. These include 7 geometric layout features, 3 character features, and 2 context features. The geometric layout features include the variance of font size of the symbols within a word based on the PDF's typesetting information, variance of the y-coordinates of the symbols, variance of inter-character gap, variance of the bounding box width and height, a measure of the degree to which all the symbols in the word correspond to the same language (i.e. English or Non-English), and percentage of English characters found within the word. The character features include the amount of mathematical characters in the word, type of the leftmost character, and type of the rightmost character. The type of the left most and right most characters give an indication as to whether or not the words in between or to the left or right are mathematical. For instance, if the right-most character of the word is an “=” then it can be inferred that whatever is directly to the left must also be mathematical.

Context features include type of the right most symbol of the previous word (word to the left), and type of the left-most symbol of the next word (word to the right). Continuing with the "=" example, if the right-most symbol of the word to the left is an "=" sign, then it can be inferred that the current word is in some form mathematical in that it is part of the equation. All of the aforementioned features are normalized to some value between [-1,1]. The features are then fed into a binary SVM classifier which has been trained on labeled datasets.

Experiments are carried out on 50 journal papers and 5 mathematical text books. 2 pages from each paper and 20 pages from each textbook are randomly selected, thus experiments are carried out on 200 pages in total. The 200 pages are divided into 5 equal subsets and 5-fold cross-validation is carried out. In each round a single subset is used for evaluation and the 4 others are used for training. This is repeated 5 times such that all 5 subsets are evaluated in this manner. The precision (positive predictive value) and recall (true positive rate) measurements are made for each of the 5 evaluations and then averaged to get the final result: 86.94% precision and 84.29% recall[5].

Also in 2012, Lin et al. proposed a new technique for the evaluation of expression segmentation methods [119]. A new evaluation metric is proposed which takes into account oversegmentations, false positives, merges, etc. Weights of various error types can be set based on specific application scenarios by changing parameters of the evaluation tool implemented. For instance, in document information retrieval of a math equation, a false negative should typically be weighted much higher than a false positive. Either area-based evaluation or symbol-based evaluation is offered but no pixel-level accuracy is achieved.

The dataset and groundtruth are claimed to be publicly available but truly are not since the documents used are not in the public domain. The dataset has 194 digitally generated PDF pages. In total, 400 document pages were carefully selected with an aim to be statistically representative of a wide variety of documents. Sources for these documents range from conference proceedings, journals, books, and reports. For each source document at least 1 and at most 8 pages are selected and added to the dataset. Documents are selected with publication years ranging from 1977 to 2010. Domain topics include mathematics, computer science, biology, and physics. 65% of the document pages are single column and the remainder are multicolumn.

[5] It should be noted that this segmentation technique suffers similar problems to Garain's 2009 approach, in that the initial word segmentation is never fixed, regardless of how incorrect the adaptive thresholding technique may be.

PDFs are also included that are generated by different PDF-writers (i.e. AFPL Ghostscript, Acrobat Distiller, Acrobat PDFWriter, ESP Ghostscript, GNU Ghostscript, Miktex PdfTex, etc). The number of displayed and embedded formulas in each page is counted and selected so that there is a wide variety of both counts.

2013 ICDAR Proceedings: Lin et al. During the 2013 ICDAR proceedings Lin et al. collaborated with Baker et al. with the goal of improving mathematical expression segmentation accuracy for digitally-born PDF documents [117]. In this work, it is argued that improper initial physical segmentation of text lines that contain math causes significant problems in formula identification. The authors describe various cases of commonly mis-segmented mathematical expressions separated into three basic categories. The first category describes a text line containing expressions that are oversegmented into two vertically overlapping lines (occurs with fractions, sums/integrals when upper/lower bounds are present, etc). The second category describes matrices and other grid-like expressions which results in similar difficulties. The third category describes a single expression which is covered by multiple lines. This occurs when an expression on the left side of an equation is set equal to multiple expressions where each subsequent expression after the first is covered by a new line. It is argued that, for identification and recognition purposes, it is best that each new line of such an expression is merged during physical segmentation as opposed to keeping each expression on the right of the equals as a separate segment.

To address these problems, a learning-based text-line merging technique is utilized. The technique utilizes one classifier to find the first two categories of mis-segmented expressions and a second classifier for the third category. First the text body is segmented from the header/footer regions. The text lines and columns are then found using projection profile cuts. Next, for each line, the decision is made as to whether or not the current line should be merged with the next line, based on the first two improperly segmented categories. For this purpose, several features are utilized. Features include vertical space between the textlines, the relative horizontal width of the textlines, the difference of indentations between the two text lines, ratio of average textline character widths and heights, ratio of main font sizes used in the lines (for digitally-born PDF's the font sizes are typically available), 2 features describing existence of fraction signs, the existence of a large operator in either line, features describing whether or not the text line ends with a binary operator, and if the lower line ends with a formula index. Then some of the aforementioned features are

employed just to describe the individual textlines themselves rather than the relation between two consecutive ones.

The classification task is separated into two stages: the first aims at properly segmenting all individual expressions and the second aims at merging single expressions that span multiple lines. When training for both of these stages, performance is compared on 7 machine learning algorithms: SVM, MLP, Decision Tree, Random Forest, Bayesian Network, Bootstrap Aggregatting (Bagging), and Adaboost. Bagging and Adaboost were reported to obtain the best performance during training for the first and second stage of segmentation respectively and were thus adopted for evaluation. The technique is evaluated on 600 document pages. 100 of these pages are used for training while the other 500 are used for evaluation. Precision and recall is reported on both stages for 100 of the the images tested, and then for the remaining 400 accuracy is only reported. The precision, recall, and accuracy were, for the most part, all reported to be above 90%. False positive rates are not reported.

2013 ICDAR Proceedings: Baker et al. Also at the 2013 ICDAR proceedings was a similar work lead by Baker et al. from University of Birmingham [118]. A tool, Maxtract [120] is introduced. This tool uses projection profile cutting to segment the mathematical expressions, and this is not very accurate at the task of segmentation. A better segmentation technique is described and its effects on the accuracy of Maxtract are reported. A histogram-based approach is described for line segmentation. The approach is to first extract all of the connected components on the page, and then determine initial lines based on grouping the connected components based on vertical proximity. A histogram is then constructed for the entire page which captures the horizontal distance between each adjacent component on each text line. Two local minimums are commonly observed in a similar location on the histograms of their pages. These local minimums are used to represent the minimum and maximum distance expected of a “principle” text line. Principle text lines are those that would correspond to either normal lines or the main lines in big math expressions (for instance, ones that contain summations, integrals, etc). The “non-principle” lines are the ones which may correspond to limits or upper and lower bounds, and thus may need to be merged with their nearest principle line. Such lines tend to be more sparesely distributed horizontally than the principle lines. Any characters having a distance observed outside of the aforementioned range for “principle” lines is considered a candidate for being part of a “non-principle” line.

A second pass is then used to correct any lines which may have been mistakenly labeled as non-principle, the heights of the non-principle lines are compared with the height of their next line. If the maximum connected component height of the non-principle line is greater than that of the principle line divided by some threshold then the nonprinciple line is re-labeled as principle. The threshold value is determined empirically on a small sample set. A third pass then checks that the lines that were considered as principle truly are principle. This is done by making sure that all of the principle lines have a height greater than the maximum height of the non-principle lines. The resulting non-principle lines are then merged with their adjacent horizontally overlapping principle lines. If a non-principle line has no adjacent horizontally overlapping principle line then it will be converted to a principle line.

The technique is evaluated on 200 pages comprising a mixture of technical journals and text books. 96.9% accuracy is reported. The technique was then further manually evaluated on a larger dataset of 1000 pages from more than 60 mathematical papers and an accuracy of 98.6% was reported. The most common error was reported as that of incorrectly classifying a non-principle line as principle. This occurs when the horizontal distance between characters on a non-principle line is similar to that of the principle line. The authors then carry out a further experiment which integrates the aforementioned segmentation technique into their Maxtract software. Note that the aforementioned technique does not necessarily go so far as to logically segment the entire mathematical expression regions, it only serves to ease the physical segmentation task prior to logical labeling. Unfortunately this also results in some errors which wouldn't occur with normal segmentation techniques. For instance, footer regions were sometimes mistaken for non-principle lines and incorrectly merged with their preceding line. A technique similar to that reported in the earlier literature [107] is then employed after the aforementioned segmentation step to identify mathematical zones.

The modified Maxtract software is evaluated on two datasets. The first dataset has 184 document pages and the second has only 10 pages. On the first dataset, the math expression identification technique was reported to have, for displayed expressions, a true positive rate of 73.18%, 7.85% false positive rate, and 1.26% false negative rate. 6.56% of the regions were reported as being oversegmented while 12.41% of the region were reported as undersegmented. No results were reported for embedded expressions in the first dataset however. In the second smaller dataset which contains only ten images, results were reported for both displayed and

embedded expressions. While the isolated expressions have a true positive rate of 78.85% and false positive rate of 1.92%, the embedded expression only have a true positive rate of 35.6% and false positive rate of 26.08%, and thus appear to require significant improvement.

Detection of Other Zone Types

While type-specific detection of mathematical expressions is the primary focus in this work, there are many other aspects to type-specific layout analysis that also need to be worked on. These may include, for instance, the segmentation of tables [121], musical scores [122], chemical equations [123], circuit diagrams [124], etc. Although these will not be studied for this work, they remain as important challenges in the field.

Check this out:::

a supervised over-

sampling technique, SMOTE [14], is adopted to generate more

positive instances, balancing the training data

SMOTE: N. V. Chawla, K. W. Bowyer, L. O. Hall, and W. P. Kegelmeyer, “Smote:

Synthetic minority over-sampling technique,” Journal of Artificial In-

telligence Research, vol. 16, pp. 321–357, 2002.

Lin, X., Gao, L., Tang, Z., Lin, X. and Hu, X., "Performance evaluation of mathematical formula identification.", in 10th International Workshop on Document Analysis Systems, 287-291 (2012).

-----^ Same authors as SVM article!!!! Describes the problems of evaluation very well. Each researcher uses their own dataset and has their own specific method for evaluation. It is, therefore, very difficult to objectively compare existing techniques. Makes a good argument about how, in much of the existing literature, describing identified regions as “partially correct” can give the developers very few clues as to how to find solutions to the problems at hand. For instance, a partially correct region could have been improperly merged with another region or split incorrectly. The metric which describes “partially correct” regions gives no indication of which error has occurred.

Bibliography

[1] K. Wilcox, and A. Stephen. "Are Close Friends the Enemy? Online Social Networks, Self-Esteem, and Self-Control," *Forthcoming Columbia Business School Research Paper No. 12-57*, Date posted: October 3, 2012.

[2] D. A. Vise, and M. Malsee, *The Google Story*. New York City: Dell Publishing, 2005.

[3] H. F. Shantz, *The History of OCR, Optical Character Recognition*. Manchester Center: Recognition Technologies Users Association, 1982.

[4] S. V. Rice, F. R. Jenkins, and T. A. Nartker, "The Fourth Annual Test of OCR Accuracy," Technical Report 95-03, Information Science Research Institute, University of Nevada, Las Vegas, July 1995.

[5] L. Vincent. "Google Book Search: Document Understanding on a Massive Scale," *International Conference on Document Analysis (ICDAR)*, 2007, pp. 819 - 823.

[6] R. Unnikrishnan, and R. Smith. "Combined Script and Page Orientation Estimation Using the Tesseract OCR Engine," *Submitted to International Workshop of Multilingual OCR*, 25th July 2009, Barcelona, Spain.

[7] Z. Huang, M. Cmejrek, and B. Zhou. "Soft Syntactic Constraints for Hierarchical Phrase-Based Translation Using Latent Syntactic Distributions," *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*, 2010, p. 138–147.

[8] P. W. Handel, "Statistical Machine," United States Patent Office. 1,915,993, Jun. 27, 1933.

[9] A. Kleiner, and R. Kurzweil, "A Description of the Kurzweil Reading Machine and a Status Report on Its Testing and Dissemination," *Bulletin of Prosthetics Research*, vol. 27, no. 10, pp. 72-81, Spring. 1977.

[10] M. Bokser, "Omnidocument Technologies," *Proceedings of the IEEE*, vol. 80, no. 7, pp. 1066-1078, July. 1992.

[11] ABBY FineReader, "ABBYY FineReader for Personal Use," Internet: http://www.nuance.com/for-business/by-product/omnipage/professional/index.htm, Date Accessed: April 3 2013.

[12] Nuance Inc., "OmniPage Professional," Internet: http://www.nuance.com/for-business/by-product/omnipage/professional/index.htm, Date Accessed: April 3 2013.

[13] Iris Products and Technologies, "Introducing the New Readiris 14," Internet: http://www.irislink.com/c2-2115-189/Readiris-14--OCR-Software--Scan--Convert---Manage-your-Documents-.aspx, Date Accessed: April 3, 2013.

[14] Contributor: Bob Stein (uploader to http://archive.org), "New York Times August September 1901 Collection," Internet: http://archive.org/download/NewYorkTimesAugSept1901Collection/New_York_Times_August_September_1901_Part_7_text.pdf, Date Accessed: March 14, 2013..

[15] T. M. Breuel, and U. Kaiserslautern. "The hOCR Microformat for OCR Workflow and Results," *Ninth International Conference on Document Analysis and Recognition (ICDAR)*, 2007, pp. 1063 - 1067.

[16] R. Griffin, *Statistics*. London: Macmillon and Co., 1913, pp. 121-122.

[17] F. d'Albe. "On a Type-Reading Optophone," *Proc. Roy. Soc., Lond.*, 1914, pp. 373-375.

[18] G. Tauschek, "Reading Machine," United States Patent Office. 2,663,758, Dec. 22, 1953.

[19] M. Martin, "Reading Machine Speaks Out Loud," *Popular Science*, vol. 154, no. 2, pp. 125-127, Feb. 1949.

[20] D. Shepard, "Apparatus for Reading," United States Patent Office. 2,663,758, Dec. 22, 1953.

[21] J. Leimer. "Design Factors in the Development of an Optical Character Recognition Machine," *IRE Transaction on Information Theory*, 1962, pp. 167-171.

[22] M. H. Weik, "A Fourth Survey of Domestic Electronic Digital Computing Systems," Internet: http://ed-thelen.org/comp-hist/BRL64-i.html#IBM-1401, Date Accessed: 2013..

[23] IBM, "IBM Systems Reference Library," 1964.

[24] L. O. Eikvil, "Optical Character Recognition," Norwegian Computing Center, 1993.

[25] J. J. Hull, and S. L. Taylor. "Document Image Skew Detection: Survey and Annotated Bibliography," *World Scientific*, 1998, pp. 40-64.

[26] R. Smith, "Apparatus and Method for Use in Image Processing," United States Patent Office. 5,583,949, Dec 10, 1996.

[27] S. Li, Q. Shen, and J. Sun, "Skew Detection Using Wavelet Decomposition and Projection Profile Analysis," *Pattern Recognition Letters*, vol. 28, no. 5, pp. 555-562, Jan. 2007.

[28] W. Postl. "Detection of Linear Oblique Structures and Skew Scan in Digitized Documents," *Proc. 6th Int. Conf. Pattern Recognition*, 1986, pp. 687-689.

[29] S. N. Srihari, and V. Govindaraju, "Analysis of Textual Images Using the Hough Transform," *Machine Vision and Applications*, vol. 2, no. 1, pp. 141-153, Jan. 1989.

[30] R. Smith, "A Simple and Efficient Skew Detection Algorithm via Text Row Accumulation," *Proc. of the 3rd Int. Conf. on Document Analysis and Recognition*, vol. 2, no. 1, pp. 1145-1148, Jan. 1995.

[31] R. Smith. "An Overview of the Tesseract OCR Engine," *Proc. Int. Conf. Document Anal. Recognit.*, 2007, pp. 629-633.

[32] S. S. Bukhari, F. Shafait, and T. M. Breuel. "Coupled Snakelet Model for Curled Textline Segmentation of Camera-Captured Document Images," *Proc. 10th Int. Conf. on Document Analysis and Recognition*, 2009, pp. 33-53.

[33] L. Likforman-Sulem, A. Zahour, and B. Taconet, "Text Line Segmentation of Historical Documents: a Survey," *Internation Journal of Document Analysis and Recognition*, vol. 9, no. 2, pp. 123-138, Jan. 2007.

[34] R. Smith. "Tesseract OCR Engine: What It Is, Where It Came From, Where It Is Going," *OSCON*, 2007.

[35] S. Mori, C. Y. Suen, and K. Yamamoto, "Historical Review of OCR Research and Development," *Proceedings of the IEEE*, vol. 80, no. 7, pp. 1029-1058, Jul. 1992.

[36] S. Mori, N. Hirobumi, and Y. Hiromitsu, *Optical Character Recognition*. New York: Wiley & Sons, Inc., 1999, pp. 193-367.

[37] R. Smith, "History of the Tesseract OCR Engine: What Worked and What Didn't. How to Build a World-Class OCR Engine in Less Than 20 Years," *SPIE-IS&T*, vol. 8658, no. 2, p. 12, Feb. 2013.

[38] F. L. Alt, "Digital Pattern Recognition by Moments," *Journal of Associated Computed Machinery*, vol. 9, no. 1, pp. 240-258, Jan. 1962.

[39] R. Casey. "Moment Normalization of Hand Printed Characters," *IBM Journal of Research and Development*, 1970, pp. 548-557.

[40] AForge.NET, "Blobs Processing," Internet: http://www.aforgenet.com/framework/features/blobs_processing.html, Date Accessed: 2013.

[41] M. D. Mcllroy, "Development of a Spelling List," *IEEE Trans on Communications*, vol. 30, no. 1, pp. 91-99, Jan. 1982.

[42] G. Nagy, "At the Frontiers of OCR," *Proc. IEEE*, vol. 80, no. 2, pp. 1093-1100, Jul. 1992.

[43] Y. Y. Tang, C. D. Yan, and C. Y. Suen, "Document Processing for Automatic Knowledge Acquisition," *IEEE Transactions on Knowledge and Data Engineering*, vol. 6, no. 1, pp. 3-21, Feb. 1994.

[44] A. M. Namboodiri, and A. Jain. "Document Structure and Layout Analysis," *Advances in Pattern Recognition*, Springer-Verlag, London. 2007.

[45] M. Nadler, "A Survey of Document Segmentation and Coding Techniques," *Computer Vision Image Graphics Process*, vol. 28, no. 2, pp. 240-262, Nov. 1984.

[46] R. Heralick. "Document Image Understanding: Geometric and Logical Layout," *Proc. IEEE Conf. Computer Vision and Pattern Recognition. Seattle*, 1994, pp. 385-390.

[47] Y. Y. Tang, S. W. Lee, and C. Y. Suen, "Automatic Document Processing: a Survey," *Pattern Recognition*, vol. 29, no. 12, pp. 1931-1952, Dec. 1996.

[48] S. Mao, A. Rosenfeld, and T. Kanungo, "Document Structure Analysis Algorithms: a Literature Survey," *Document Recognition and Retrieval*, vol. 5010, no. 10, pp. 197-207, Jan. 2003.

[49] N. Chen, and D. Blostein, "A Survey of Document Image Classification: Problem Statement, Classifier Architecture and Performance Evaluation," *Inter-*

national Journal on Document Analysis and Recognition, vol. 10, no. 1, pp. 1-16, May. 2007.

[50] S. Marinai. "Introduction to Document Analysis and Recognition," *Machine learning in document analysis and recognition, Berlin, Germany: Springer*, 2008, pp. 1-20.

[51] G. Nagy, S. Seth, and M. Viswanathan, "A Prototype Document Image Analysis System for Technical Journals," *Computer*, vol. 25, no. 1, pp. 10-22, Jan. 1992.

[52] Y. N. Elglaly, F. Quek, T. Smith-Jackson, and D. Dhillon. "Touch-Screens Are Not Tangible: Fusing Tangible Interaction With Touch Glass in Readers for the Blind," *ACM International Conference on Tangible, Embedded and Embodied Interaction (TEI), Barcelona, Spain*, 2013, pp. 245-252.

[53] F. Goudail, *Statistical Image Processing for Noisy Images*. New York: Klewer Academic Plenum Publishers, 2004.

[54] R. Miller, "Ink-Jet Basics," Internet: http://www.thetonesystem.com/inkjet_basics.html, Date Accessed: 2013.

[55] O. G. Guleryuz, "A Multiresolutional Algorithm for Halftone Detection," *Proc. SPIE Image and Video Communications and Processing*, vol. 5685, no. 1, pp. 1098-1105, Jan. 2005.

[56] N. Otsu, "A Threshold Selection Method From Gray-Level Histograms," *IEEE Trans. Sys. Man. Cyber*, vol. 9, no. 1, pp. 62-66, Jan. 1979.

[57] W. R. Smith. "Hybrid Page Layout Analysis via Tab-Stop Detection," *Proceedings of the 10th International Conference on Document Analysis and Recognition*, 2009, pp. 241-245.

[58] B. Xie, and G. Agam, "Boosting Based Text and Non-Text Region Classification," *Document Recognition and Retrieval Proc. SPIE*, vol. XVIII, no. 1, pp. 1-9, Jan. 2011.

[59] A. Gourdol, "CSS3 Regions: Rich Page Layout With HTML And CSS3," Internet: http://www.adobe.com/devnet/html5/articles/css3-regions.html, Date Accessed: 2013..

[60] T. Pavlidis, and J. Zhou, "Page Segmentation and Classification," *Graphical Models and Image Processing*, vol. 54, no. 1, pp. 484-496, Jan. 1992.

[61] H. S. Baird, H. Bunke, and P. S. Wang. "Background Structure in Document Images," *Document Image Analysis, World Scientific, Singapore*, 1994, pp. 17-34.

[62] G. Nagy, and S. Seth. "Hierarchical Representation of Optically Scanned Documents," *Proc. of the 17th Conf. on Pattern Recognition*, 1984, pp. 347-349.

[63] F. Wahl, K. Wong, and R. Casey, "Block Segmentation and Text Extraction in Mixed Text/Image Documents," *Graphical Models and Image Processing*, vol. 20, no. 1, pp. 375-390, Jan. 1982.

[64] J. Higashino, H. Fujisawa, Y. Nakano, and M. Ejiri. "A Knowledge-Based Segmentation Method for Document Understanding," *Proc. 8th Int. Conf. on Pattern Recognition*, 1986.

[65] A. Dengel, and F. Dubiel, "Computer Understanding of Document Structure," *International Journal of Imaging Systems and Technology*, vol. 7, no. 1, pp. 271-278, Jan. 1996.

[66] A.K. Jain, Y. Zhong, "Page segmentation using texture analysis," *Pattern Recognition*, vol. 29, no. 5, pp. 743–770, May. 1996.

[67] T. Tokuyasu and P. A. Chou, "Turbo recognition: a statistical approach to layout analysis," *In Proceedings of the SPIE*, vol. 4307, no. 1, pp. 123-129, Jan. San Jose, CA, 2001.

[68] J. P. Bixler. "Tracking Text in Mixed-Mode Document," *Proc. ACM Conference on Document Processing System*, 1998, pp. 177-185.

[69] T. Pavlidis, "Algorithms for Graphics and Image Processing," *ZAMM - Journal of Applied Mathematics and Mechanics / Zeitschrift für Angewandte Mathematik und Mechanik*, vol. 63, no. 8, p. 395, Jan. 1983.

[70] L. O'Gorman, "The Document Spectrum for Page Layout Analysis," *IEE Trans. Pattern Analysis and Machine Intelligence*, vol. 15, no. 11, pp. 162-173, Nov. 1993.

[71] S. Arya, T. Malamotos, and D. M. Mount. "Space-Efficient Approximate Voronoi Diagrams," *Proc. 34th ACM Sympos. Theory Comput.*, 2002, pp. 721-730.

[72] Wikipedia, "Voronoi Diagram," Internet: http://en.wikipedia.org/wiki/Voronoi_diagram, Date Accessed: 2013.

[73] K. Kise, A. Sata, and M. Iwata, "Segmentation of Page Images Using the Area Voronoi Diagram," *Computer Vision and Image Understanding*, vol. 70, no. 3, pp. 370-382, Jun. 1998.

[74] E. G. Johnston, "Printed Text Discrimination," *Computer Graphics and Image Processing*, vol. 3, no. 1, pp. 83-89, Mar. 1974.

[75] D. S. Bloomberg. "Multiresolution Morphological Approach to Document Image Analysis," *1st ICDAR*, 1991, pp. 963-971.

[76] R. C. Gonzalez and R. E. Woods, *Digital Image Processing*. Upper Saddle River NJ: Prentice Hall, 2008, pp. 627-688.

[77] J. Serra, *Image Analysis and Mathematical Morphology*. New-York: Academic Press, 1982.

[78] J. Liu, Y. Tang, Q. He, and C. Suen. "Adaptive Document Segmentation and Geometric Relation Labeling: Algorithms and Experimental Results," *Proc. 13th Int'l Conf. Pattern Recognition, Vienna*, 1996, pp. 763-767.

[79] Esposito, D. Malerba, and G. Semeraro. "A Knowledge-Based Approach to the Layout Analysis," *Proc. Third Int'l Conf. Document Analysis and Recognition, Montreal*, 1995, pp. 466-471.

[80] M. Okamoto and M. Takahashi. "A hybrid page segmentation method," *Proc. 2nd Int. Conf. on Document Analysis and Recognition, Tsukuba Science City, Japan* , 1993, pp. 743-748.

[81] T.M. Breuel. "Two Geometric Algorithms for Layout Analysis," *Proceedings of the 5th International Workshop on Document Analysis Systems V*, 2002, pp. 188–199.

[82] H.S Baird, S.E Jones, and S.J Fortune. "Image segmentation by shape-directed covers," *Proceedings 10th ICPR, Atlantic City, New Jersey*, June, 1990, pp. 820–825.

[83] T. M. Breuel. "The OCRopus Open Source OCR System," *Proc. IS&T/SPIE 20th Annu. Symp.*, pp. 1 -15, 2008.

[84] I. Phillips, B. Chanda, and R. Haralick, "UW-III English/Technical Document Image Database," Internet:

http://www.science.uva.nl/research/dlia/datasets/uwash3.html, Date Accessed: 2013.

[85] F. Shafait. "Geometric Layout Analysis of scanned documents," *PhD thesis, University of Kaiserslautern*, 2008.

[86] F. Shafait. "Document Image Analysis with OCRopus," *Proc. IEEEInt'l Multitopic Conf.*, 2009.

[87] W. Abd Almageed, M. Agrawal, W. Seo, and D. Doermann. "Document-zone classification using partial least squares and hybrid classifiers," *Proc. Int'l Conf. on Patt. Reco.*, 2008, pp. 1-4.

[88] F. Cesarini, M. Lastri, S. Marinai, and G. Soda. "Encoding of modified X-Y trees for document classification," *Int. Conf. on Document Analysis and Recognition (ICDAR)*, 2001, pp. 1131–1135.

[89] C. Shin and D. Doermann. "Classification of Document Page Images Based on Visual Similarity of Layout Structures," *Proc. SPIE Conf. Document Recognition and Retrieval VII*, 2000, pp. 182-190.

[90] F. Esposito, D. Malerba, and G. Semeraro, "Classification in noisy environments using a distance measure between structural symbolic descriptions," *IEEE Trans. on PAMI*, vol. 14, no. 3, pp. 390–402, March. 1992.

[91] N. Chen and D. Blostein, "A survey of document image classification: problem statement, classifier architecture and performance evaluation," *International Journal on Document Analysis and Recognition (IJDAR)*, vol. 10, no. 1, pp. 1-16, June. 2007.

[92] K. Fan and L. Wang, "Classification of Document Blocks Using Density Feature and Connectivity Histogram," *Pattern Recognition Letters*, vol. 16, no. 9, pp. 955-962, September. 1995.

[93] D. Chetverikov, J. Liang, J. Komuves, and R. Haralick. "ZoneClassification Using Texture Features," *Proc. 13th Int'l Conf. Pattern Recognition, Vienna*, 1996.

[94] Y. Wang , R. Haralick and I. Phillips, "Document Zone Content Classification and Its Performance Evaluation," *Pattern Recognition*, vol. 39, no. 1, pp. 57 -73, Jan. 2006.

[95] W. Abd Almageed, M. Agrawal, W. Seo, and D. Doermann. "Document-zone classification using partial least squares and hybrid classifiers," *Proc. Int'l Conf. on Patt. Reco.*, 2008, pp. 1-4.

[96] M. Okamoto and A. Miyazawa, *An experimental implementation of a document recognition system for papers containing mathematical expressions*. Berlin: Springer, 1992.

[97] Hsi-Jian Lee and Jiumn-Shine Wang. "Design of a mathematical expression recognition system," *Proceedings of the Third International Conference on Document Analysis and Recognition*, Aug 1995, p. 1084.

[98] K. Inoue, R. Miyazaki and M. Suzuki. "Optical Recognition of Printed Mathematical Documents," *Proc. Third Asian Technology Conf. Math.*, 1998, pp. 280-289.

[99] R.J. Fateman, "How to Find Mathematics on a Scanned Page," *Proc. SPIE*, vol. 3967, no. 1, pp. 98-109, December. 1999.

[100] Toumit JY, Garcia-Salicetti S, Emtoz H. "A hierarchical and recursive model of mathematical expressions for automatic reading of mathematical documents," *Proceedings of Fifth International Conference on Document Analysis and Recognition (ICDAR)*, 1999, pp. 119–122.

[101] B.B. Chaudhuri and U. Garain. "An Approach for Recognition and Interpretation of Mathematical Expressions in Printed Document," *Pattern Analysis and Applications vol. 3*, 2000, pp. 120-131.

[102] A Kacem, A Belaid, B M Ahmed, "Automatic extraction of printed mathematical formulasusing fuzzy logic and propagation of context," *International Journal of Document Analysis and Recognition*, vol. 4, no. 2, pp. 97-108, December. 2001.

[103] J. Jin, X. Han and Q. Wang. "Mathematical Formulas Extraction," *Proc. of the 7th Int'l Conf. Document Analysis and Recognition (ICDAR), Edinburgh, Scotland*, 2003, pp. 1138-1141.

[104] D.M. Drake, H.S. Baird. "Distinguishing mathematics notation from english text using computational geometry," *Proceedings of the International Conference on Document Analysis and Recognition, Seoul*, 2005, pp. 1270–1274.

[105] X. D. Tian, W. Z. Sun, M. H. Ha. "Research on optical formulas extraction," *Proceedings of the 4th International Conference On Machine Learning and Cybernetics, Guangzhou, China*, August 2005, pp. 4886-4890.

[106] S Yamazaki, F Furukori. "Embedding a mathematical ocr module into ocropus," *International Conference on Document Analysis and Recognition*, 2011, pp. 880 - 884.

[107] X. Lin, L. Gao, Z. Tang, X. Hu, and X. Lin. "Identification of embedded mathematical formulas in PDF documents using SVM," *Proc of DRR XIX*, 2012, pp. 8297.

[108] K. Chan and D. Yeung, "Mathematical Expression Recognition: A Survey," *Int',l J. Document Analysis and Recognition*, vol. 3, no. 1, pp. 3-15, August. 2000.

[109] R. Zanibbi and D. Blostein, "Recognition and retrieval of mathematical expressions," *IJDAR*, vol. 15, no. 4, pp. 331-357, December. 2012.

[110] CD Malon, S Uchida, M Suzuki, "Mathematical symbol recognition with support vector machines," *Pattern Recognition Letters*, vol. 29, no. 9, pp. 1326–1332, July. 2008.

[111] Hsi-Jian Lee and Jiumn-Shine Wang, "Design of a mathematical expression understanding system," *Pattern Recognition Letters*, vol. 18, no. 3, pp. 289-298, March. 1997.

[112] Chowdhury, S.P., Mandal,S., Das, A.K., Chanda, B.. "Automated segmentation of math-zones from document images," *7th International Conference on Document Analysis and Recognition vol. 2*, 2003, pp. 755–759.

[113] U Garain, BB Chaudhuri, and A. R. Chaudhuri. "Identification of embedded mathematical expressions in scanned documents," *Int. Conf. Pattern Recognition*, 2004, pp. 384-387.

[114] U. Garain and B.B. Chaudhuri, "A Corpus for OCR of Printed Mathematical Expressions," *Int'l. Journal of Document Analysis and Recognition (IJDAR)*, vol. 7, no. 4, pp. 241-259, September. 2005.

[115] U. Garain. "Identification of mathematical expressions in document images," *Proc. Int'l Conf. on Document Analysis and Recognition*, 2009, p. 1340.

[116] S. Uchida, A. Nomura, and M. Suzuki, "Quantitative analysis of mathematical documents," *IJDAR*, vol. 7, no. 4, pp. 211-218, June. 2005.

[117] X Lin, L Gao, Z Tang, J Baker, M Alkalai, V Sorge. "A Text Line Detection Method for Mathematical Formula Recognition," *12th International Conference on Document Analysis and Recognition*, 2013, pp. 339-343.

[118] M Alkalai, J B Baker, V Sorge, X Lin. "Improving Formula Analysis with Line and Mathematics Identificatio," *12th International Conference on Document Analysis and Recognition*, 2013, pp. 334-338.

[119] X Lin., L Gao, Z Tang, X Lin and X Hu. "Performance evaluation of mathematical formula identification," *10th International Workshop on Document Analysis Systems*, 2012, pp. 287-291 .

[120] J Baker, "Maxtract," Internet: http://www.cs.bham.ac.uk/research/groupings/reasoning/sdag/maxtract.php, Date Accessed: 2013.

[121] R Zanibbi , D Blostein , R Cordy, "A survey of table recognition: Models, observations, transformations, and inferences," *IJDAR*, vol. 7, no. 1, pp. 1-16, March. 2004.

[122] Marinai and Nesi. "Projection based segmentation of musical sheets," *ICDAR, Bangalore, India*, 1999.

[123] J Baker, AP Sexton, V Sorge. "Comparing Approaches to Mathematical Document Analysis from PDF," *ICDAR*, 2011.

[124] SS Bukhari, F Shafait, and TM Breuel. "Document image segmentation using discriminative learning over connected components," *9th IAPR International Workshop on Document Analysis Systems*, 2010, pp. 183-190.

Colophon

<A colophon (literally, end stroke) is an inscription at the end of a written work, containing facts about its production. It may name artists, printers etc. and discuss typographic and technical details such as typefaces and papers.>

Attached electronic data

Description of attached files and folders[6]

- <file name> <description>
- <folder name> <description>
  - <file name> <description>
  - <folder name> <description>
- <folder name> <description>
  - <file name> <description>
  - <folder name> <description>
    - <file name> <description>
    - <file name> <description>
  - <file name> <description>
- <folder name> <description>

[6] You might find additional auxiliary files for download at this thesis' web location.

Accessing the attached electronic data

The paper version of this thesis should contain an envelope with an optical data medium (CD or DVD) here.

The digital version of this thesis contains the content of this medium as attachments to the PDF file. If your PDF viewer cannot handle attachments you may access these files at this thesis' web location.