Segmentation of heterogeneous document images : an approach based on machine learning, connected components analysis, and texture analysis

Omid Bonakdar Sakhi

UNIVERSITÉ PARIS-EST

Segmentation of heterogeneous document images : an approach based on machine learning, connected components, and texture analysis

Dissertation

submitted to the
Department of Computer Science
Universit Paris-Est
for the fulfillment of the requirements for the doctoral degree
Doctorat en Informatique

by

Omid Sakhi

Thesis advisor:
Prof. Dr. Laurent Najman

Thesis co-advisor:
Dr. Xavier Hilaire

July 8, 2013

Abstract

Document page segmentation is one of the most crucial steps in document image analysis. It ideally aims to explain the full structure of any document page, distinguishing text zones, graphics, photographs, halftones, figures, tables, etc.

Although to date, there have been made several attempts of achieving correct page segmentation results, there are still many difficulties. The leader of the project in the framework of which this PhD work has been funded[1] uses a complete processing chain in which page segmentation mistakes are manually corrected by human operators. Aside from the costs it represents, it demands tuning of a large number of parameters; moreover, sometimes segmentation mistakes escape the vigilance of the human operators.

Current automated page segmentation methods are well accepted for clean printed documents; but, they often fail to separate regions in handwritten documents when the document layout structure is loosely defined or when side notes are present inside the page. Moreover, tables and advertisements bring additional challenges for region segmentation algorithms. Our method addresses these problems. The method is divided into four parts:

1. Unlike most of popular page segmentation methods, we first separate text and graphical components of the page using a boosted decision tree classifier.

2. The separated text and graphical components are used among other features to separate columns of text in a two-dimensional conditional random fields framework.

3. A text line detection method, based on piecewise projection profiles is then applied to detect text lines with respect to text region boundaries.

4. Finally, a new paragraph detection method, which is trained on the common models of paragraphs, is applied on text lines to find paragraphs based on geometric appearance of text lines and their indentations.

Our contribution over existing work lies in essence in the use, or adaptation, of algorithms borrowed from machine learning literature, to solve difficult cases. Indeed, we demonstrate a number of improvements : on separating text columns when one is situated very close to the other; on preventing the contents of a cell in a table to be merged with the contents of other adjacent cells; on preventing regions inside a frame to be merged with other text regions around, especially side notes, even when the latter are written using a font similar to that the text body.

Quantitative assessment, and comparison of the performances of our method with competitive algorithms using widely acknowledged metrics and evaluation methodologies, is also provided to a large extend.

[1]This PhD thesis has been funded by Conseil Général de Seine-Saint-Denis, through the FU16 project Demat-Factory, lead by Safig SA

Resumé

La segmentation de page est l'une des étapes les plus importantes de l'analyse d'images de documents. Idéalement, une méthode de segmentation doit être capable de reconstituter la structure complète de toute page de document, en distinguant les zones de textes, les parties graphiques, les photographies, les croquis, les figures, les tables, etc.

En dépit de nombreuses méthodes proposées à ce jour pour produire une segmentation de page correcte, les difficultés sont toujours nombreuses. Le chef de file du projet qui a rendu possible le financement de ce travail de thése [2] utilise une chaîne de traitement complête dans laquelle les erreurs de segmentation sont corrigées manuellement. Hormis les coûts que cela représente, le résultat est subordonné au réglage de nombreux paramètres. En outre, certaines erreurs échappent parfois à la vigilance des opérateurs humains.

Les résultats des méthodes de segmentation de page sont généralement acceptables sur des documents propres et bien imprimés; mais l'échec est souvent à constater lorsqu'il s'agit de segmenter des documents manuscrits, lorsque la structure de ces derniers est vague, ou lorsqu'ils contiennent des notes de marge. En outre, les tables et les publicités présentent autant de défis supplémentaires à relever pour les algorithmes de segmentation. Notre méthode traite ces problèmes. La méthode est divisée en quatre parties :

1. A contrario de ce qui est fait dans la plupart des méthodes de segmentation de page classiques, nous commençons par séparer les parties textuelles et graphiques de la page en utilisant un arbre de décision boosté.

2. Les parties textuelles et graphiques sont utilisées, avec d'autres fonctions caractéristiques, par un champ conditionnel aléatoire bidimensionnel pour séparer les colonnes de texte.

3. Une méthode de détection de lignes, basée sur les profils partiels de projection, est alors lancée pour détecter les lignes de texte par rapport aux frontières des zones de texte.

4. Enfin, une nouvelle méthode de détection de paragraphes, entraînée sur les modèles de paragraphes les plus courants, est appliquée sur les lignes de texte pour extraire les paragraphes, en s'appuyant sur l'apparence géométrique des lignes de texte et leur indentation.

[2]Cette thèse a été financée par le Conseil Général de Seine-Saint-Denis, par l'intermédiaire du projet Demat-Factory, initié et conduit par SAFIG SA

Notre contribution sur l'existant réside essentiellement dans l'utilisation, ou l'adaptation, d'algorithmes empruntés aux méthodes d'apprentissage automatique de données, pour résoudre les cas les plus difficiles. Nous démontrons en effet un certain nombre d'améliorations : sur la séparation des colonnes de texte lorsqu'elles sont proches l'une de l'autre ; sur le risque de fusion d'au moins deux cellules adjacentes d'une même table ; sur le risque qu'une région encadrée fusionne avec d'autres régions textuelles, en particulier les notes de marge, même lorsque ces dernières sont écrites avec une fonte proche de celle du corps du texte.

L'évaluation quantitative, et la comparaison des performances de notre méthode avec des algorithmes concurrents par des métriques et des méthodologies d'évaluation reconnues, sont également fournies dans une large mesure.

Acknowledgements

This work would not have been possible without the support of many people. I wish to express my sincere gratitude to all those who gave me the possibility to complete this thesis.

First of all, I would like to thank my supervisor, Prof. Laurent Najman, for helping me realize this project and for his guidance during the course of this work.

I would like to appreciate Dr. Xavier Hilaire, who provided me with valuable technical feedback and suggestions, as well as pointing out mistakes and carefully reviewing many versions of this thesis. I am deeply indebted to him for more than three years of scientific help.

I would also like to thank Dr. Apostolos Antonacopoulos for granting access to datasets for ICDAR2009 and ICDAR2011 competitions and I also thank Christian Clausner for his answers to many of my emails regarding Prima Layout Evaluation and Altheia and his help for debugging some bugs that I encountered during the course of performance evaluation of our method.

No words can express my gratitude toward my parents who helped me night and day through difficult times, and for all the support and care they provided.

Finally I would like to thank members of the jury including Prof. Dr. PASSAT Nicolas, Prof. Dr. GERAUD Thierry and the president of the jury, Prof. Dr. WENDLING Laurent, for their time and consideration to review this manuscript.

This PhD thesis has been funded by Conseil Général de Seine-Saint-Denis, through the FU16 project Demat-Factory, lead by Safig SA.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Introduction

For centuries, paper documents such as handwritten manuscripts and books have been used as the main source for preserving knowledge. Then, with the advent of printing, printed books, magazines and newspapers have replaced the traditional way of preserving information. Nowadays, the most common way to store documents is by storing them in a digital format. It is fast to search, cheap and portable. While many of the ancient documents have been lost over the years, a substantial number has been preserved. Not only these documents are fragile, they are also inaccessible for the public. As a consequence, many libraries and organizations around the world have decided to convert their collections to a digital format.

Two most common ways for digitizing documents are to use image scanners or digital cameras. However, both methods produce images that demand a large storage and they are unsearchable. Here, document image analysis comes into play. Document image analysis is the subfield of digital image processing with the goal of converting document images to searchable text form. The whole process starts with segmenting a document image into different parts such as text, graphics and drawings and forming a layout structure of the document. Finally, having a layout structure, methods can determine the reading order of the document or send text regions to an optical character recognition (OCR) module which converts text regions into searchable format.

In this thesis, we develop a method for segmenting a document image into its different parts. Currently, Safig, the leader of Demat-Factory, use a software to do the same task, but with the supervision of human. The undisclosed software that they use has many tunable parameters that need to be monitored carefully for each document in order to produce correct segmentation result. Our goal is to automate this process and generalize the method in a way that it can be applied to a broad range of documents. There are many open-source and commercial applications such as Tesseract-OCR, OCRopus, ABBYY FineReader that perform the same task. However, as we will examine later, they are more fitted for well formatted printed documents that are free of noise. Our goal is to

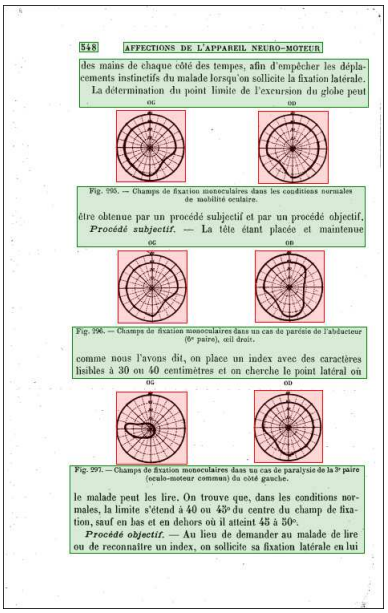


Figure 1.1: A sample of a correctly segmented document image by ABBYY FineReader 2011.

improve the segmentation quality for the corpus provided to us, which mostly consists of handwritten documents with degraded quality and side notes, forms and books. We start with an introduction to document page segmentation.

1.2 Document page segmentation

Document page segmentation is the main component of geometric layout analysis. Given an image of a document, the goal of page segmentation is to decompose the image into smaller homogeneous regions (zones or segments) of handwritten and printed text. The difference between page segmentation and layout analysis is that layout analysis algorithms use these segments to assign contextual labels (title, author, footnote,...) to them and to also find the reading order of each segment. Figure 1.1 shows a correctly segmented document image. In multi-columns documents, a page segmentation algorithm is also responsible for segmenting text columns separately, so that text lines from different columns are not merged.

The reason for segmenting documents into smaller regions in the first place is that text regions are sent to an OCR (Optical character recognition) or reading order detection modules for further processing and to convert them into ASCII format. Hence, obtained regions should also be classified as containing text or non-text elements, because character recognition modules assume that the incoming data contains text, so their outputs are unpredictable for regions containing graphics or other non-textual components.

Although it is not mandatory to segment text regions into paragraphs before passing text lines to optical character recognition modules, it is necessary to have correct paragraphs for reading order detection. Thus, in this work we also group text lines into paragraphs after identifying each text region. It is worth noting that there are more than one possible way to segment a document image into text regions correctly that depends on the reading order in the ground truth, but there is only one solution for segmenting text regions into paragraphs.

Because of the important role page segmentation plays in document layout analysis, and of its direct effect on the optical character recognition step, it has been explored deeply for the last four decades by document imaging community and many algorithms have been proposed in the literature. A comprehensive overview of these algorithms is provided by Nagy in [65] and Cattoni et al. in [22]. In this work we review many of these algorithms in the domain of text/graphics separation, text line and region detection. Then a new system for page segmentation is proposed that improves the results of segmentation in areas where problems still exist by current algorithms.

1.3 Overview of the approach

We view a document as a scene of connected components (CCs). Thus, the first step in our method is an image binarization and extraction of all connected components. The goal of the system is to find locations of all paragraphs inside the document image. In order to form paragraphs, we need to detect text lines correctly. And to do so, we have to detect text regions considering the geometric alignment of CCs. Text lines in multi-column documents should be separate from one another and in the case where side notes exist, we have to use the alignment of the CCs to separate side notes from the main text. All these operations should be carried out without reading the actual text or understanding the context of the text.

After binarization, we have a set of connected components that either belong to text or non-text regions. The next step is to correctly classify each connected component. Each CC has a set of intrinsic features (height, width, eccentricity, ...) and a set of extrinsic features (features from surrounding area). Using these features, we train a set of weak classifiers by the help of boosting which allows us to classify components as text or non-text. In chapter 3 we go deeply into the details of this method.

At this point, we are interested to segment a document image into regions of text that have some form of alignment. We consider both sets of textual and non textual CCs for this task. Note that some non-text CCs such as tables and rule lines are highly effective in separating regions, so we do not discard non textual components. In chapter 4 we show how the method effectively makes the best out of these sets to separate text inside tables.

Having all text elements and their regions, detecting text lines is the next step. Chapter 2 thoroughly reviews major methods in the literature for text line

detection and we conclude that the text line detection algorithm by Vassilio Papavassilious [75] is the most effective among the others. The original method overlooks line detection in side notes, but being the best among the many and the fact that we have already separated problematic areas such as side notes, we adopt this method for the benefit of our own. Detailed explanation of this method is chapter 5.

The final stage of the system is to group text lines into paragraphs. In chapter 6 we propose a method based on a trainable binary tree model that maximize the probability of preserving groups of lines using a goodness criterion for paragraphs.

A brief outline of these processes is given here.

- **Image binarization** is the process that converts a given input gray-scale image into a bi-level representation. In our case, pixels that belong to text characters are assigned a value of 0 and pixels of non textual components and background have a value of 1.
- **Connected components analysis** is the process of extracting and labeling connected components from an image. In our case, all pixels of text and non-text elements that are connected and have the same value are extracted and assigned to a separate component.
- **Noise removal** tries to detect and remove noise pixels from the document image. At this stage we only remove components that contain less than a predefined number of pixels. The exact number of pixels can only be determined through trial and error. A large number may remove points, diacritics and punctuation marks. On the other hand, a small number may not remove some dust and speckles from the scene.
- **Text/Graphics separation** is the process that classifies each component into being part of text or graphics.
- **Text region detection** is the process that separates homogeneous regions of text that belong to separate columns. It is also responsible for separation of side notes from the main text region.
- **Text line detection** is the process that finds text lines inside every text region. It is also responsible for breaking characters that are touched from two adjacent lines and have formed a single component incorrectly.
- **Paragraph detection** is the process that groups text lines into paragraphs based on their indentations and geometry.

1.4 Contribution of this dissertation

The main contributions that are presented in this dissertation are:

1. A new hybrid method for text/graphics separation. Figure 1.2 shows the ability of our text/graphics separation method in segmenting a document

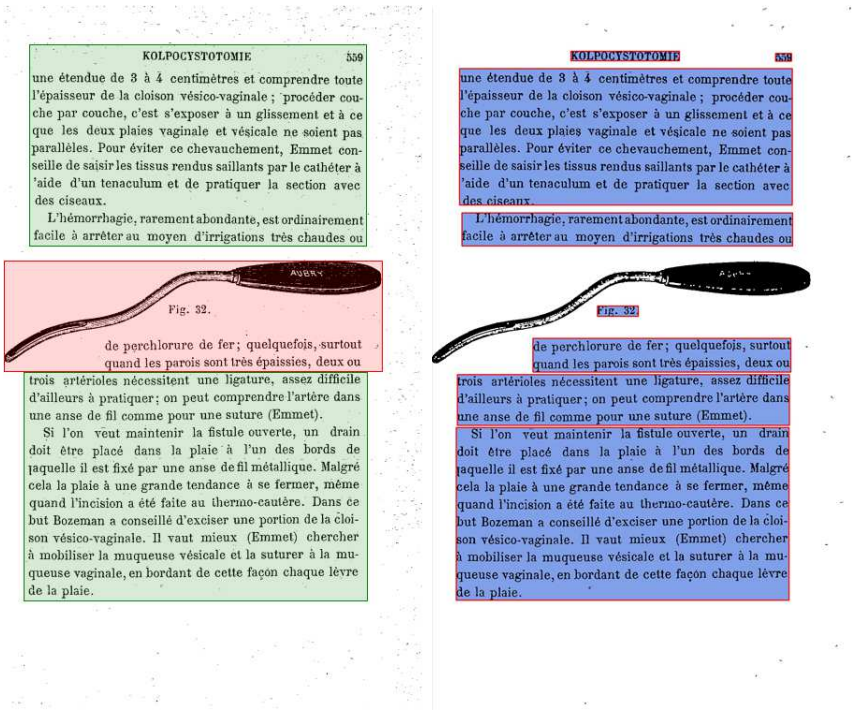


Figure 1.2: **Left**; A document page that is segmented incorrectly by ABBYY FineReader 2011 and **Right**; The same document segmented by our method.

page. The proposed method is based on both intrinsic and extrinsic features that have the advantages of both connected component based and block-based methods in detecting graphical elements inside a document image.

2. Identification of side notes in historical document images as a new step in document image analysis.

   Although due to the advances in page segmentation algorithms during the last decade, page segmentation methods can segment multi-column documents correctly, but problems still exist in some area such as detecting side notes when they are situated close to the main text. Figure 1.3 shows the result of page segmentation with ABBYY FineReader 2011 that has gone wrong. Note that when the segmentation is wrong, the result of OCR is also not impressive. Figure 1.4 demonstrates the result of page segmentation, obtained from our method in the form of paragraphs for the same image.

3. A powerful framework for text region detection and column separation that provides the ability of inducing prior knowledge about the document into the detector.

4. A new trainable method for grouping text lines into paragraphs based on a binary tree model that maximize the probability of preserving groups

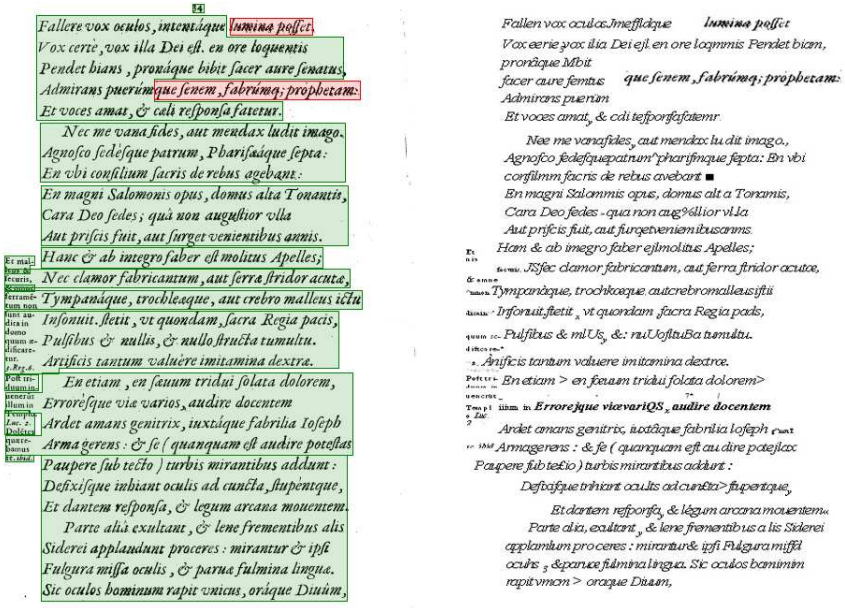


Figure 1.3: Left; A document page that is segmented incorrectly by ABBYY Fine Reader 2011 and **Right**; Its corresponding result for character recognition.

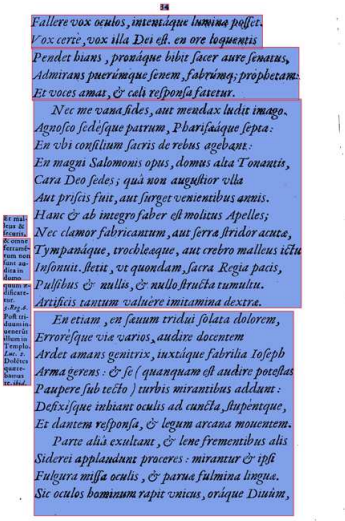


Figure 1.4: The result of paragraph detection of the image in figure 1.3 by our method.

of text lines using a criterion that depends on the geometric appearances and indentations of text lines in a paragraph.

1.5 Our datasets

One of the datasets used for testing and training in this work is a selection of document images from a huge corpus, that were provided for the purpose of this project. The original corpus contains historical handwritten manuscripts and old printed forms, documents and advertisements, mostly written in French language. We have chosen a subset of 100 documents from this corpus that represents the original set. Figure 1.5 shows some samples of our own dataset.

In addition, we gathered 61 documents used as part of the dataset in ICDAR2009 [6] page segmentation competition and 100 documents from the dataset used in ICDAR2011 [4] historical document layout analysis competition, to be able to compare our results with the state-of-the-art methods. Some samples from these datasets are shown in figure 1.6.

For each document a ground truth file is available. The ground truth file contains the true structure of text regions and text lines for each document. By far ground truth data are the most important part of the work for the purpose of both training and testing various parts of the system. They are usually in text format and contain coordinates of objects in a hierarchical structure. For text and graphics separation we only need the original document image and the location of true text regions. Every component that does not belong to a text region is considered a non-text element. A collection of textual and non-textual components are used for the purpose of training. In text region detection, we divide every document into sites with predefined heights and widths. By knowing whether a site is located on the text or non-text region of the ground truth, we can easily generate our true labels for the purpose of training our region detector. For text line and paragraph detection we consider the geometry appearance and relative position of text lines defined in the ground truth structure.

In the first stages of the project, we had no ground truth date for any of the datasets. So we developed a software to generate our own true data. Figure 1.7 shows a screen shot of our developed software. After opening an image, the software renders a scene of all connected components extracted from the image. The user can group several connected components as a text line. With the same strategy the user can group several text lines as a paragraph. In case a large connected component (a table or a frame of advertisement) occults other smaller components, the user has the option to deactivate that particular component. Finally, the application generates an XML file for each document with the correct structure to be used as a ground truth. Figure 1.8 shows part of an XML file created by our software.

Later, we got access to ground truth data for ICDAR2009 dataset. The new ground truth data are also provided in XML format but the nodes and elements of the new XML documents are different from ours. Because the dataset and ground truth data belong to Prima group at University of Salford,

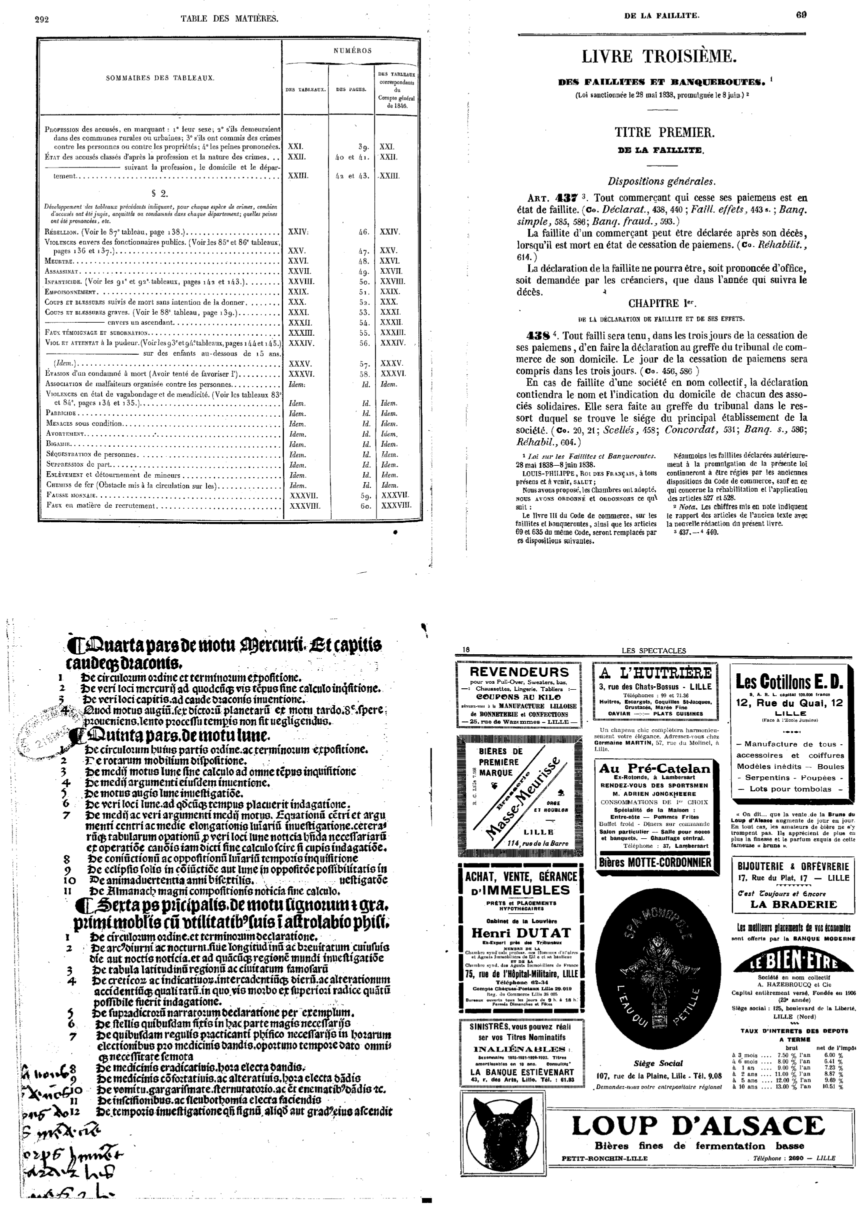


Figure 1.5: Some sample documents from our own corpus.

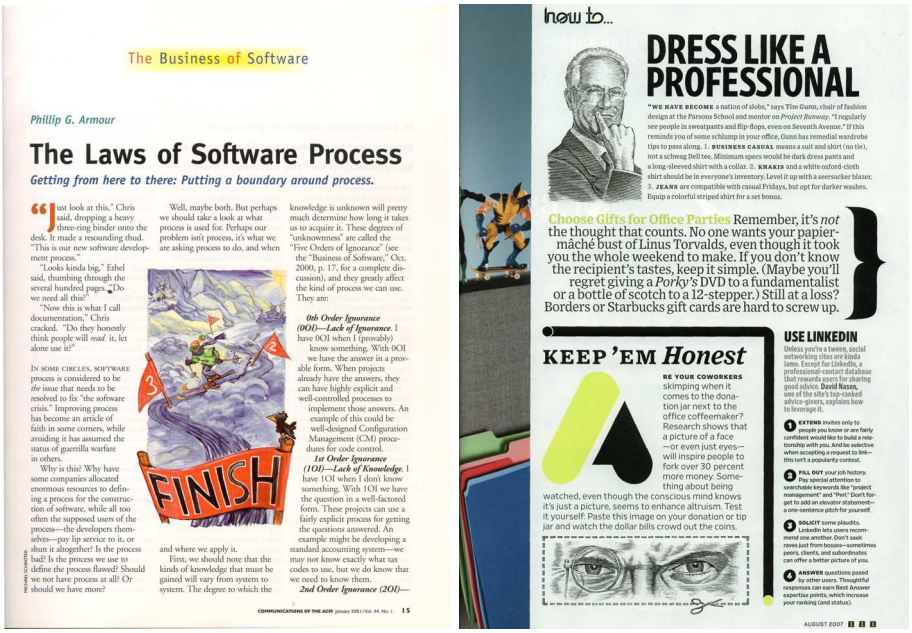


Figure 1.6: Two sample documents from the dataset for ICDAR2009 competition [6].

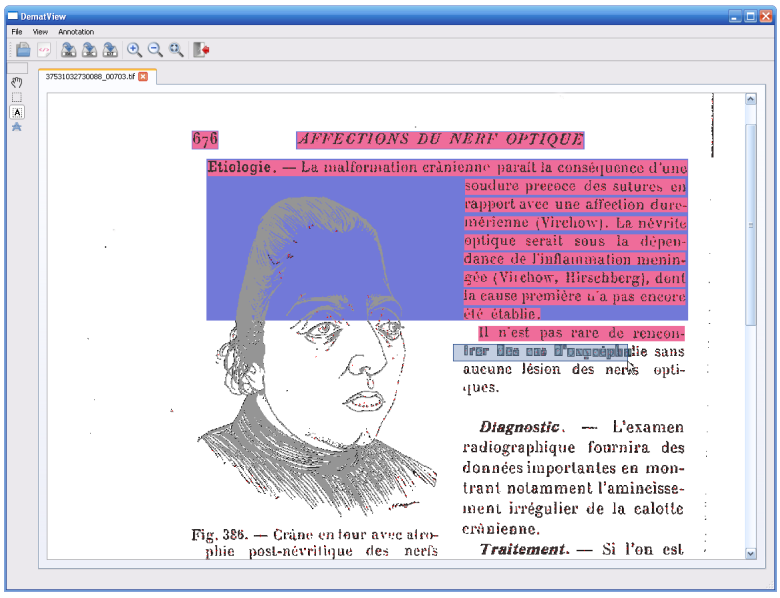


Figure 1.7: A screen shot of our developed software for generating XML ground truth documents for each document image

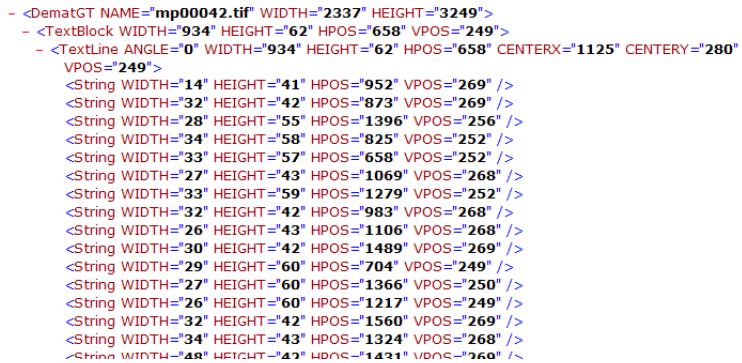


Figure 1.8: A screen shot that shows part of the contents of an XML file created by our software

they can only be read and write using Aletheia[25] from Prima tools. The other tool from Prima group is Prima Layout Evaluation[25] that compares the XML output from a segmentation process with the ground truth data and provides detailed evaluations on how they differ from each other. Due to this new tool, we annotated every document with Aletheia and produced XML ground truth documents for the purpose of evaluation.

1.6 Organization of this dissertation

Figure 1.9 gives an overview of the organization of this thesis, highlighting the connection between different chapters of this thesis. This dissertation is organized as follows:

Chapter 2 outlines the major areas of work in document page segmentation and addresses key parts of text segmentation in document literature.

We describe our system with all the details in the span of four chapters. Chapter 3 deals with various aspects of text/graphics separation of out method. Text region detection is explained in chapter 4. Text lines and paragraphs detection are accounted in chapters 5 and 6, respectively. We note the experiments and results for each part of the method at the end of its own chapter.

Finally, in chapter 7 we give recommendation for future work.

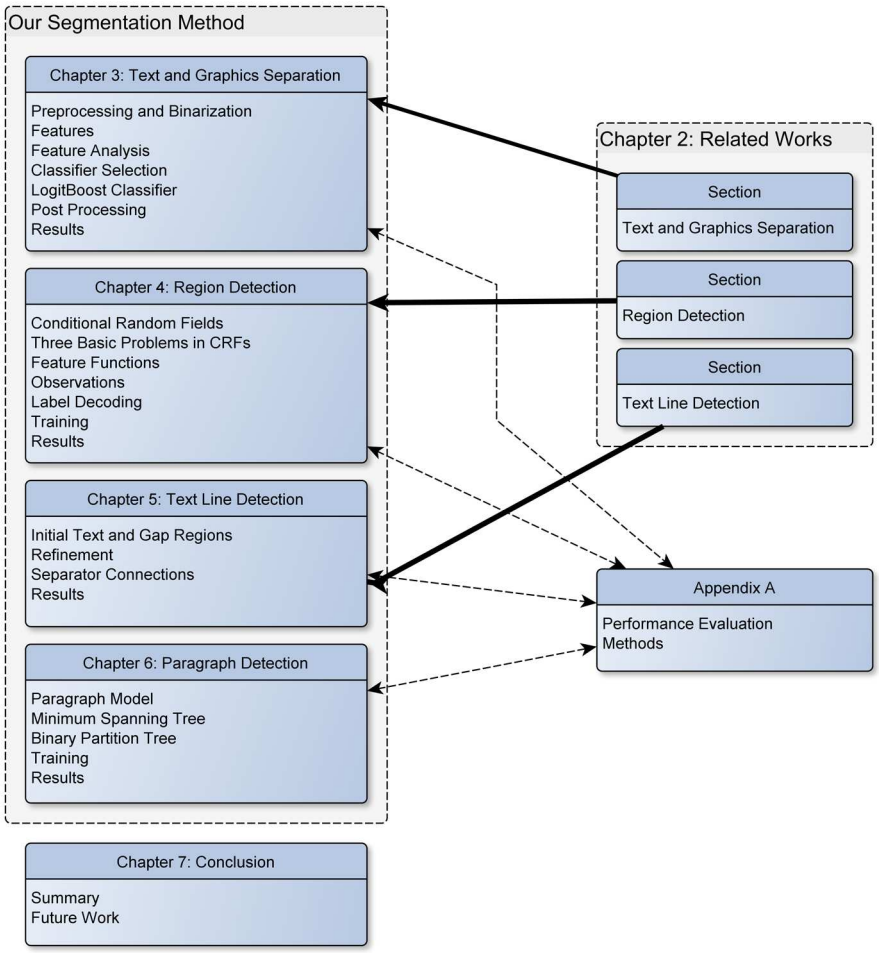


Figure 1.9: A visualization of the structure of this thesis displaying the connection between different chapters and their contribution to different areas of page segmentation.

Chapter 2

Related work

2.1 Text/graphics separation

Separating text and non-text elements inside a document image is the foremost part of any page segmentation algorithm. In other words, the output of a page segmentation algorithm should not contain graphics, rule-lines or any other element that is not in coherence with OCR/HCR[1] modules.

Due to the importance of this step, attempts have been made to develop effective methods and algorithms since the early days of document image analysis. However, the choice of an optimal method depends heavily on the type of the document image that has to be processed. Documents in general can be divided into two major categories [65]. Maps, music scores, engineering drawing and organization charts are samples of the first category which contain mostly graphics. Scanned pages, camera-captured forms, text books, historical manuscripts and advertisements belong to the second category which contains mostly text.

In general, obtaining text characters by analysis of connected components is much easier in mostly text documents compared to mostly graphics due to the simplicity of its background. Textured backgrounds are particularly difficult to handle. Examples of such textured backgrounds can be found frequently in camera-captured scenes. One might reason that camera captured scenes are out of the scope of document image analysis, but sometimes methods for locating text in these scenes can be adapted for the field of document analysis as it did in ICDAR2005 text locating competition [59]. For this reason, two groups of algorithms have emerged. The first group of methods work at the connected component level. These methods assign a label text, non-text to each connected component of the image. The methods in the second group analyze the texture of a region containing pixels within an image which can be of convex or free shape and assign one of the mentioned labels to that region. In this section, we briefly note some of these methods and outline their strengths and weaknesses.

[1] Handprint character recognition

2.1.1 Connected component based methods

As the name suggests, connected component based methods work with connected components to discriminate text from graphical elements within the document image. Maybe one of the earliest methods and still popular for its robustness and usability with increasingly complex documents is that of Fletcher and Kasturi [33]. The method is based on Hough transform, working on the center of the bounding boxes and works by grouping aligned components into strings of characters. Then it classifies all isolated components as graphics.

There are major drawbacks to this approach:

- Tables and borders around advertisements have a center that is usually located inside the text area. So it is easy to group them incorrectly as part of a character chain, unless there are some constraints that govern the size of component.
- The classifications of short strings of characters are not reliable due to lack of votes in the Hough space to efficiently discriminate them.
- The method may find diagonal alignments when text lines are packed closely and there are not enough gaps between them.
- Punctuation marks, diacritics and broken characters, are not aligned with other components in a string of text, and they may become a seed for misclassification.

Despite all these limitations, authors of [96] have recently published a paper and the results are improved. The difficulty of the problem lies not only in the classification of these components but also in the separation of interacting components. When textual and non textual elements interact locally, finding a solution becomes more difficult. Figure 2.1 shows two cases of such a problem. In [28] Doermann tries to address this issue with a method based on stroke level properties to separate components. As an illustration, of the potential discrimination power of the stroke level properties, it is noted that in hand-completed forms and pre-printed boxes, lines are produced by a machine and have more regularity than the associated handwritten text. Considering only the widths of the strokes and examining the population of widths at the cross section level, strong separability can be achieved between the two populations.

Another type of problem that frequently arise in a connected component based method is that large graphical components are often broken into pieces, and they are composed of many small isolated components that behave like text components. Many methods try to address this problem by isolating a graphical component and its sub elements as a whole rather than classifying each one separately. Figure 2.2 illustrates graphical elements from two documents in our corpus that exhibit this issue.

One possible solution is to apply a method like the one proposed by B. Waked in his thesis [99]. The idea is that text regions can be regarded as a set of small bounding boxes that are regular in height and are usually aligned horizontally or vertically, whereas a non-text image or half-tone graphics is irregular. The

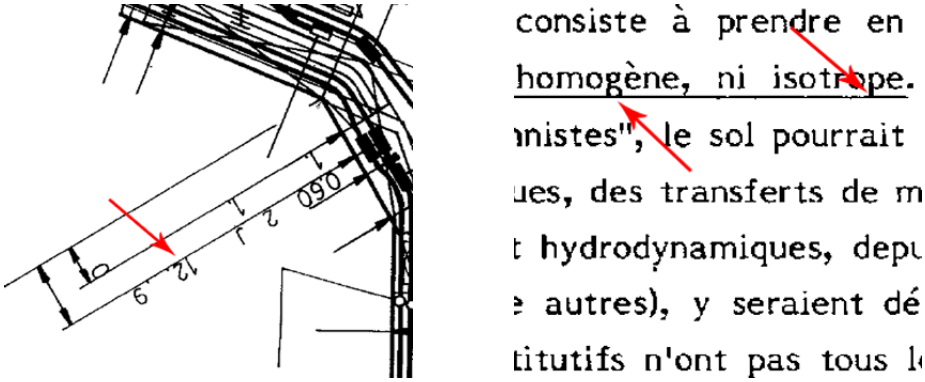


Figure 2.1: The image on the **Left** shows part of a document page from [96] when text and numerical components touch graphical elements and the image on the **Right** is part of a document from our corpus showing a difficult situation when underlines touch characters of a text line.

Figure 2.2: Two graphical figures from our corpus that exhibit broken components with text like features.

size of its components varies from small to large, and they are rarely aligned in either direction. Therefore, the sum projection of pixels in non-text regions on x or y axis does not present a regularity between the peaks and the gaps. The method uses the standard deviation of runs in projection profiles to decide whether the region contains text or graphics. At this point, the reader might wonder "where do these regions come from?" As it turned out the method, first performs region detection by analyzing diagonal run-lengths around each connected component, so as to first detect regions before taking out graphical elements. Because of this, if the region detection merges a text region with a closely located non-textual element, the result from the method is unpredictable.

Another approach proposed by Bloomberg [10] is solely based on image processing techniques and has been recently revisited by [15]. The main technique in Bloomberg's text/graphics separation method is based on multi-resolution morphology. In this method, an image is closed with a rather large structure element to consolidate half-tone or broken components. Then the image is opened with an even larger structure element to remove text blobs (strings of characters) and to preserve some portions of graphical components. The remaining portions, called seeds, can be grown to cover graphics. In [15] Bukhari explains that if a non-text component is only composed of thin lines and small drawings, then it vanishes from the seed image after the second opening with the large structure element. Therefore, Bloomberg's method often fails to detect non-textual components when they do not contain any solid bunch of pixels. Then, Bukhari proposes several modifications to the original algorithm that improves the classification accuracy by a 3% and 6% increase on UW-III and ICDAR2000 datasets, respectively. One of the modifications is to add a hole-filling morphological operation after the first closing. While the hole-filling step improves the detection of non-text elements, it cannot be profitable for the documents in our corpus because of the many document images that contain advertisements with large frames around a bunch of text lines. The hole-filling step, fills inside these frames and the page becomes empty of text. The same fate happens to table structures with closed boundaries.

A rather different approach is utilized in [16] and [42]. Both methods extract features from connected components and use some form of machine learning to train a classifier. The first method extracts the shape of a component and its surrounding area. Then a multi-layer perceptron (MLP) neural network is trained to learn these shapes and classify the component as either text or non-text. The second method extracts a set of 17 geometrical features (e.g. aspect ratio, area ratio, compactness, number of holes) that are computed from the component, and a support vector machine (SVM) is trained to handle classification.

2.1.2 Region-based methods

Not all methods are applicable to documents with complex background structure. In fact, in some cases it is not even feasible to extract text components because different shades of colors pass over and through the text region and generate many components in such a way that text components are lost among

them.

In such circumstances, it would be logical to take advantage of methods that work on a block or a region of document instead of the connected components. The idea is that regions of text have a particular texture that can be revealed with methods such as wavelet analysis, co-occurrence matrices, Radon transform, etc.

Shape of the regions can be rectangular or free form. They can be strips of overlapping blocks or coming from a multi-resolution structure like quad-tree or image pyramids. To name a few methods, Journet [44] uses a method based on orientation of auto-correlation and frequency features to segment regions of text in historical document images. In [46], Kim et al propose an approach based on support vector machine (SVM) and CAMSHIFT algorithm.

In CC-based methods, the geometry of the components is clearly defined. However, no matter how well the methods work, there is always an issue of confidence at the boundaries of regions. This issue becomes larger as the gap between text and non-textual components gets smaller. Using large block implies high confidence on the computed features, but poor confidence on regions' boundaries. Using small blocks implies just the opposite. One solution is to use overlapping blocks, but then the problem becomes to assign a label to pixels when two or more overlapping blocks have a disagreement about the label of the pixels. Confidence based methods for combination of evidence (e.g. Dempster-Shafer) also may fail. Consider the case where two blocks have overlapping pixels. One block is located on the mostly text area of the document, and the other block is located on the mostly graphical one. The pixels in between get evidences from two sources that have a high degree of conflict. The Dempster-Shafer theory is criticized for being incompetence in the mentioned situation [104].

The other problem with region-based methods is their ineffectiveness to detect frames and lines that belong to a table. Thus, the result for parts of a table that contain text, comes back as text for every pixel of that region, whereas it includes separating rule lines.

2.1.3 Conclusion

Whenever it is feasible to compute the connected components of a document, it is preferable to assign the label to the connected component itself and avoid using a region-based method. However, to overcome some of the drawbacks of the component based methods, a hybrid approach should be adapted that also takes advantage of texture features around the component. Our method is based on this strategy and is described in chapter 3.

2.2 Text region detection

Text region detection is the actual process of segmenting a page into separate homogeneous text regions. In our view, text region detection is only one part of the page segmentation process along side node separation, graphics separation and other pre-processing and post-processing steps. However, in the literature authors often use page segmentation, page decomposition or geometrical layout analysis on behalf of region detection. Maybe one reason is that in some cases, methods do not explicitly separate text and graphics before segmenting page into text regions. In one case that we reviewed before [99], separation of text and graphics comes even after detecting regions. So the point is that region detection, page segmentation, page decomposition may all refer to the same process with the same goal.

Many methods have been proposed for this goal. Back to the days when Nagy wrote his famous review on document image analysis [65], he divided page segmentation methods into two major top-down and bottom-up approaches categories. Top-down analysis attempts to find larger components, like columns and block of text before proceeding to find text lines, words and characters. Bottom-up analysis forms words into text lines, lines into text blocks and so on. Then within the last decade and due to advances in computation and optimization techniques, more and more authors began to report a new emerged category. Khurshid wrote [45]: "Many other methods that do not fit into either of these categories are therefore called hybrid methods.".

It is part of our belief that this type of categorization does not represent the majority of the available methods in the literature any more. It is more convenient to divide current methods into three categories; distance-based, texture-based and white space analysis. Distance-based methods try to analyze the distance between connected components and to separate regions that fall apart. Examples of these methods use Voronoi diagram [2, 47, 48], Delauney tessellation [102] or run-length features [93, 32], map of foreground and background runs .

Texture-based methods use wavelets analysis [49, 1], spline wavelets [27], wavelet packets [30], Markov random fields (MRF) with pixel density features [69], auto correlation [44] in a single or multi-resolution framework to segment pages into text regions. Finally, methods that focus in the white background of the document and try to find the maximal empty rectangles that can separate columns of text [13, 85, 14].

In [84], Shafait et al, evaluate the performance of six famous algorithms, including X-Y cut [66], run-length smearing [101], whitespace analysis [8], Docstrum [71], and Voronoi-based [48] page segmentation. Also several methods for historical document layout analysis are reported in [4], but unfortunately except for their concise description in the paper, none of them is available publicly for study and experimentation.

In the rest of this section, we briefly highlight some advantages and weaknesses of each category, and then we provide an overview of our approach.

2.2.1 Distance-based methods

Run-length smearing (RLSA) [101] is perhaps the oldest known technique to segment a page into homogeneous regions. It smears components using the perceived text direction to form a distinct block of text. The Docstrum algorithm [71] starts by finding the K-nearest neighbors of each connected component and connects them by edges. Then the histogram of distances and angles are computed for all edges. Text lines are found by grouping pairs of closest neighbors, and the method proceeds to form text blocks by grouping text lines. Ferilli et al., [32] indicate that white run lengths embody a distance-like feature similar to the one explored in the Docstrum method, and borrow the idea to merge closest neighbors based upon the values of horizontal and vertical white runs between pairs of connected components. If adjacent regions from different columns have a similar homogeneity, often they are merged. Because of this, other variants of the run-length method have also been proposed to improve the results. To name a few, Constrained Run-Length Algorithm (CRLA) [98] and selective CRLA [93].

Clearly, methods based on Voronoi diagram like [47, 48] also follow the same strategy to filter the edges of a graph and separate regions of text. Voronoi++ [2] is the latest proposed method that utilizes dynamic distance thresholding instead of global thresholds that, to some extent, addresses the problem of over segmentation around large characters and grouping of dissimilar text sizes. Despite an increase of 33% in the detecting accuracy of regions compare to [48], the text precision and recall of the method on Washington III (UW-III) database are 68.78% and 78.30%, respectively.

The battle front in all these methods comes down to how well they can apply distance thresholds to remove appropriate edges and group the remaining connected components as a text region. In the mentioned works, different thresholds have been set, not all of which are always suitable for other datasets. In other words, these methods rely upon certain assumptions about document layouts, and they fail when the underlying assumptions are not met. The other issue with these methods is that because they rarely take advantage of the alignment of components, they often merge side notes with the main text body. In fact, we are not aware of a method that uses the alignment of components, shape of regions or distance between components, all at the same time.

2.2.2 Whitespace analysis

Methods in this category try to analyze the background structure (whitespace) of the document image to determine the physical document layout analysis. The rudimentary algorithms for whitespace analysis were limited to axis-aligned rectangles. As a consequence, they have to correct document rotation before the search operation every time. In 2003, [13] Breuel developed a method that can find maximal empty rectangles more efficiently, which would consequently be applied to considerably more complex type of documents. The new method

seeks to find a set of parameters (location, width, height and orientation) for each rectangle by using interval arithmetic and a branch-and-bound (B&B) reduction [51] over the four-dimensional parameter space to efficiently exclude large regions of parameter space that does not contain good solutions.

The development of this algorithm has led to a system for document layout analysis in [14]. In this system the algorithm finds N largest non-overlapping maximal white-space rectangles in a resolution independent framework. Details of the methods are noted in [12] as well as [13]. The algorithm is also used as part of the method in [86, 85] to model document layouts statistically.

As noted on several occasions in the papers, finding maximal rectangles using this method eliminates the need for page rotation correction prior to background analysis and is efficient for complex documents. However over-segmentation errors may occur where word gaps have exactly the same position and of the same width as the column separator. In one reported case [85] noise prevented the method to find the white space needed to correctly match the model. The authors have not evaluated their algorithm on handwritten documents with side notes and under-segmentation may appear in this situation. Furthermore, the effects of having tables and advertisements' frames are not considered in the discussions and because of rule lines, the algorithm may not be able to separate columns of text inside a table structure.

2.2.3 Texture-based methods

Page segmentation using texture-based methods utilizes filtering methods to recognize the underlying texture of a region and to segment an image into homogeneous regions. In most instances, the main method aims at segmenting an image into coherent and homogeneous regions containing text, graphics and background all at the same time. Thus, these methods do not attempt to recognize graphical elements a priori.

The method in [27] is one that uses polynomial spline wavelets aiming to segment documents and detecting graphical components at the same time. It uses cubic B-spline wavelets for this purpose. It is suggested that B-spline and D-spline wavelets can be a good candidate for document segmentation. Then the method use k-means classification with k being the pre-defined number of different textures in the processed image to classify wavelet features. The final classification stage isolates each text line separately and authors use post-processing, including morphological opening to merge text lines and correct unsatisfactory results.

Kumar et al [49] also use wavelet filters for segmenting complex images, including document images, written text with a pen and camera-captured image from soda cans, into text, background and picture components, but the difference is that instead of using a pre-designed wavelet like D-spline or B-spline, they train wavelet filters to match their needs using a collection of ground-truth images. As a post-processing step they use pixel-based Markov random fields to refine the segmentation results.

[21] is another texture-based method that uses a neuro-fuzzy approach. The accuracy of the main algorithm is about 59%. However, after morphological operations, the authors report 96% as the accuracy of the method.

Finally, we mention the method in [69] based on Markov random fields (MRF). The authors consider that any clique is constructed from 4-connected neighbors, and the interaction terms only consider horizontal and vertical directions. For each node, nine features are gathered from each scale and just two scales are considered. These features are density values of black pixels (text) associated to the current clique in both scales. This method has not been evaluated on popular datasets; however, we mention it because MRFs are one of the predecessors to what we use in our method involving conditional random fields.

Texture-based methods on their own can be used to classify and isolate text regions, but they are not specifically designed to separate regions of text. As a consequence, refinement and merging operations are usually needed to correct segmentation results. Because of these merging operations, under-segmentation often occurs. The second issue is that in many circumstances, parts of graphical elements are situated close to text regions, so they are recognized as text and are merged with the text region. The third and fundamental problem with texture-based methods is that there is no theoretical proof about which filtering method is better than the other except to empirically report the results. The main three parts of a texture-based method is the filtering strategy, the classifier and the training method. So many authors publish methods by changing one or all parts and report new results.

2.2.4 Conclusion

In conclusion, texture-based methods that are solely based on image filtering, fit best for finding text in complex scenes other than document images. Example of such scenes might be text detection from images of vehicle's plates registration or from camera-captured scenes. Methods based on whitespace analysis and distance-based methods have their own drawbacks. Clearly, a method is desirable that takes all consideration into account.

2.3 Text line detection

Detecting text lines in document images is a critical step towards optical character recognition (OCR) or handwritten character recognition (HCR). This process refers to the segmentation of each text region into distinct entities, namely text lines. The overall performance of an OCR/HCR system strongly relies upon the results from text line detection process. Different methods have been proposed for detecting printed and handwritten text lines. In both cases there exist several challenges; however, in printed documents, detecting text lines is a rather straightforward process and is not a fit for detecting handwritten text lines. In contrast, most of the proposed methods for detecting handwritten text

Figure 2.3: Two documents in our corpus that exhibit variability of font sizes in newspaper style documents.

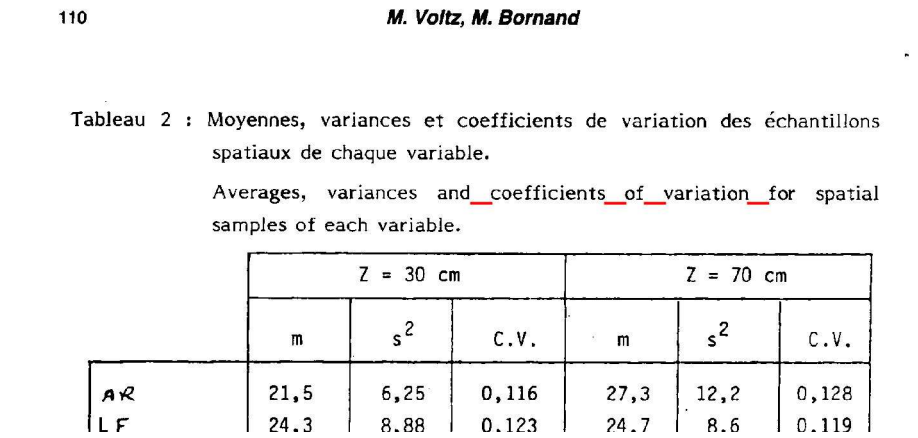
lines are adaptable for detecting printed text lines.

Here are some challenges that arise in text line detection:

- **Variability of font size** is a challenge in some documents (e.g. newspaper style). Most methods assume that the distribution of font sizes in one page is Gaussian. As a result, they tune the parameters for the average font size. This often leads to overlook large characters in the title or errors when there are sudden changes in font sizes. Figure 2.3 illustrates this problem clearly.

- **Slanted text lines** are straight text lines that are not aligned with the x-axis. Most algorithms prefer to start with a skew correction for the whole page and then proceed to detect text lines. However, this is still a challenge when text lines with different slant angles are present on the same page.

- **Touching text lines** frequently occur in text regions, especially in handwritten manuscripts where text lines are located close to one another. This situation is a challenge for most methods. Connected component-based methods that work their way by aligning and grouping connected components together, fail because in the case of touching text lines, two characters from two lines that have touched are registered as one large connected components that is not in alignment with either of the text lines. If not dealt with properly, this will lead to either under-segmentation of two lines into one line, or over-segmentation of two different text lines into



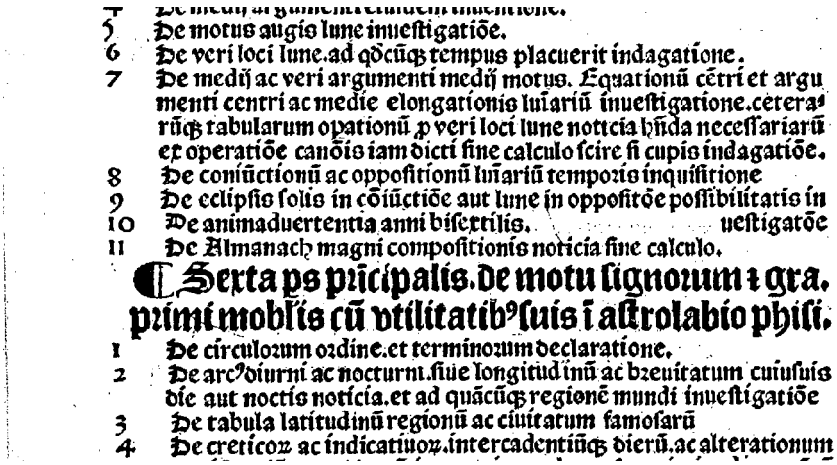
110 M. Voltz, M. Bornand

Tableau 2 : Moyennes, variances et coefficients de variation des échantillons spatiaux de chaque variable.

Averages, variances and coefficients of variation for spatial samples of each variable.

	Z = 30 cm			Z = 70 cm		
	m	s^2	C.V.	m	s^2	C.V.
AR	21,5	6,25	0,116	27,3	12,2	0,128
LF	24.3	8.88	0.123	24.7	8.6	0.119

Figure 2.4: Part of a document in our corpus that contain large gaps between words.



5 De motus augis lune inueſtigatiõe.
6 De veri loci lune.ad qõcũq; tempus placuerit indagatione.
7 De medĩ ac veri argumenti medĩ motus. Equationũ cẽtri et argumenti centri ac medie elongationis luĩariũ inueſtigatione.ceteraꝝ rũq; tabularum opationũ p veri loci lune notticia hũda neceſſariarũ et operatiõe canõis iam dicti ſine calculo ſcire ſi cupis indagatiõe.
8 De coniũctionũ ac oppoſitionũ luĩariũ temporis inquiſitione
9 De eclipſi ſolis in cõiũctiõe aut lune in oppoſitõe poſſibilitatis in
10 De animaduertentia anni biſextilis. ueſtigatõe
11 De Almanach magni compoſitionis noticia ſine calculo.

Sexta ps pĩcipalis.de motu ſignoꝝ t gra. primi mobĩs cũ vtilitatibꝰ ſuis ĩ aſtrolabio phiſi.

1 De circuloꝝ ordine.et terminoꝝ declaratione.
2 De arcꝰ diurni ac nocturni.ſiue longitudinũ ac breuitatum cuiuſuis die aut noctis notícia.et ad quãcũq; regionẽ mundi inueſtigatiõe
3 De tabula latitudinũ regionũ ac ciuitatum famoſarũ
4 De cretícoꝝ ac indicatiuoꝝ.intercadentiũq; dierũ.ac alterationum

Figure 2.5: Part of a document in our corpus that shows closely situated text lines.

5 or more text lines.

- **Large gaps** between words may lead to over-segmentation of text lines and often cause fragments in text lines. Figure 2.4 shows an example of large gaps between words.

- **Closely situated text lines** can become a challenge when the distance between two components from a single text line is larger than the distance between two components from different text lines. Text lines in Figure 2.5 are so close that distance-based methods such as minimum spanning tree, fail to produce correct results.

- **Highly curved text lines** and calligraphy can be found in some freestyle

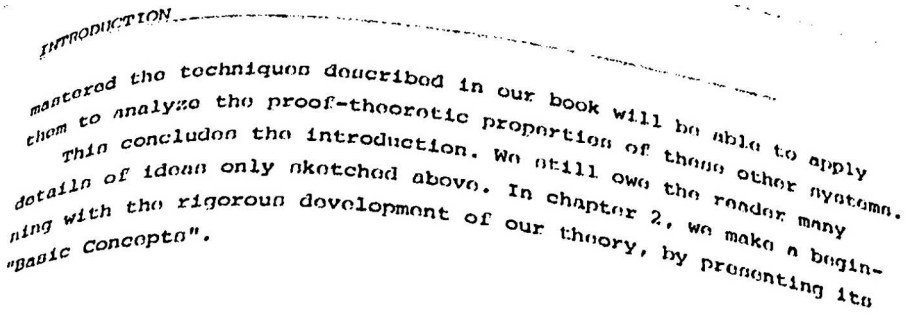


Figure 2.6: Part of a camera-captured document from [17] that shows skewed text lines.

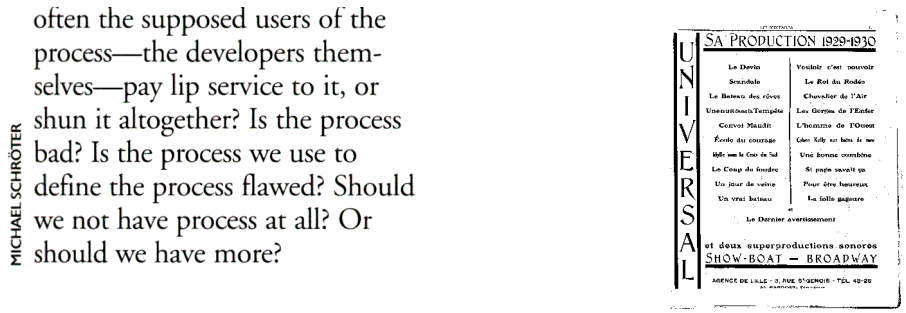


Figure 2.7: Left image is a cropped part of a document from ICDAR2009 database and **Right** image is an advertisement page from our own corpus. Both show vertical text lines that are hard to detect.

historical document; however, they are seldom seen in more recent documents.

- **Skewed text lines** appear on camera-captured books. Thick books are often hard and time-consuming to be scanned manually using flat scanners. One solution is to use an automated system to turn pages and take photos of each page. The downside is that text lines appear to be skewed near the borders. Figure 2.6 shows part of a camera-captured document from [17] containing the mentioned skewed text lines.

- **Vertical text lines** often exist in magazine-style pages or advertisements. They come in two flavours. Either characters are in correct direction but appear on top of each other, or the whole line with all characters is rotated by $\pm 90°$. Most methods are designed to detect slanted text lines with a limited degree of freedom; however, over-segmentation errors occur when the algorithm is not designed to deal with vertical text lines. Figure 2.7 shows an example of these vertical lines that can be found occasionally in documents.

- **Degraded quality** due to ageing process, is a challenge for ancient documents. Usually degraded areas affect the result of binarization step which as a consequence introduces false connected components and noise throughout the page that affect the true alignment of the characters in a text line.

- **Transparency effect** is a challenge in copied documents. In some copied documents, sheets are so thin that the contents on the back of the page appear on the front side of the document and depending on the binarization strategy, they may appear as connected components in the main page.

- **Scripting language** is not a problem by itself; however, some analysis techniques take advantages of the specific characteristic of the script to detect text lines. For example, some methods look to finding one base line or two base lines in order to detect text lines in English language. These methods may easily produce unpredictable results when applied to documents in other script languages such as Arabic or Chinese.

Many methods have been proposed to cope with these challenges, but because of the diversity of problems, each method has been designed specifically for a particular application. Thus, it presumes some assumptions of the class of the document and the scripting language to fit well for the majority of documents in that class. Related reviews are published in [54] that presents a ten-year survey of methods for text line segmentation from historical documents and [79] reviews handwritten text line detection. In this section, we review some of these methods and then we decide which approach to take for detecting text lines in text regions in our approach. The detailed description of our method is written in chapter 5.

We divide methods into the ones that are designed for detecting printed text lines and those for detecting handwritten text, but with more emphasis on methods for detecting handwritten text lines, since they are easily applicable to printed text regions.

2.3.1 Printed text line detection

Nagy wrote in his review [65]: "Excellent methods are available for locating all the text on high-contrast printed pages down to the word level with an accuracy that exceeds that of subsequent steps..." It is a fact that text line detection from clean and well-formatted printed documents is now a solved problem. Perhaps projection based methods are one of the most successful top-down algorithms for machine printed documents [52] because the gap between two adjacent text lines is typically big, and the text line is easily separable by analyzing peaks and valleys of the projection profiles. Even so, when one or more aforementioned challenges exist, the success rate of the methods plummets substantially.

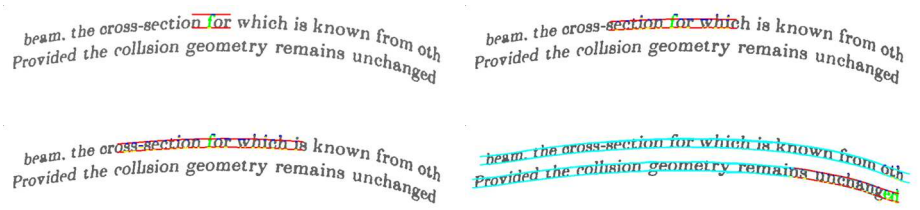


Figure 2.8: These images from [18] illustrate how the snakes grow to engulf a text line.

In the review section for text region detection, we already reviewed distance-based methods that try to segment document image into text regions. Some methods use the same strategy to detect text lines. In one method [102] Delaunay triangulation is used to find a set of edges between connected components. Then the algorithm looks for the principal direction among the shortest edges to estimate the direction of the text lines. This method can be applied to pages containing slanted text lines successfully. Nevertheless, because the strategy for filtering edges based on their lengths is somehow primitive and the method uses a global threshold to remove edges, it becomes ineffective for detecting skewed text lines or separating side notes.

To detect skewed text lines that exist on camera-captured documents, Bukhari et al. [17, 18] propose a method based on active contour models. Active contour model is a framework for finding the outline of an object from a possibly noisy image [24]. The method starts with multiple initial closed contours (snakes) that engulf text blobs. Each snake is associated with an energy that is usually the sum of the internal and external forces, computed from the gradient vector flow (GVF) at the boundaries of the contours. Then in each iteration, snakes grow by a fixed number of pixels to minimize the associated energies. All overlapping pairs of snakes are merged together to form a single snake that covers a curled text line. Results indicate that this method has the potential to detect highly curled text lines with different spacing and font sizes. The analysis of the errors shows that most of them are due to over-segmentation of a single line in the ground-truth into multiple detected text lines. Figure 2.8 illustrates how snakes grow on characters to engulf the skewed text line.

Another method [19] is also proposed for detecting text lines in camera-captured documents. One advantage of this new method is its ability to locate curled text lines. The approach is using an oriented anisotropic Gaussian smoothing function to generate a filter bank. For each pixel, the maximum value among all scales and orientations is selected for the final smoothed image. Then Horn-Riley based ridge detection algorithm [41, 80] is used to enhance text line structures from the smoothed image. The result from this work is mentioned in CBDAR2007 document image de-warping contest [83]. Although the method performs well on documents with printed text, it would be hard to be adapted for handwritten documents where alignment of the strings and non-uniform gaps between words are an issue and the results many contain many text line fragments.

2.3.2 Handwritten text line detection

Detecting text lines from handwritten documents has been one of the growing trends in the past decade because of the high demand for digitization and preservation of historical documents by most libraries around the world. In general, detection of handwritten text lines has more challenges compare to printed text lines, and it demands robust methods to deal with. We divide proposed methods into three different categories namely; projection-based, texture-based and distance based methods.

Projection based methods

Projection based methods divide into two categories; global and piecewise projections. Global projection based methods like X-Y cut have been one of the earliest methods that were applied to printed documents. However, they were not effective for detecting high skewed text lines. Moreover, the presence of dark borders in some scanned documents or moderate noise could also affect the detection rate of the method. To address this problem for handwritten historical documents, piecewise projection based methods are proposed in many works including [75, 90, 95]. Using piecewise projection, a document is divided into either blocks or non-overlapping vertical zones with equal width. Then the vertical projection is computed in each zone separately.

An example of using global projection profiles can be found in [62]. Manmatha and Rothfeder use projection profiles to detect handwritten text lines, specifically for the George Washington manuscripts. The assumption is that these particular manuscripts contain lines that are approximately straight and roughly parallel to the horizontal axis. Therefore, global projection profiles are used. Global projections may fit this dataset, but in order to detect skewed text lines, most authors use piecewise projections. Figure 2.9 displays application of this method on one document for detecting text lines.

[87] presents an algorithm using adaptive local connectivity map for retrieving text lines from historical handwritten documents. The algorithm is designed for solving particularly complex problems, including fluctuating text lines, touching or crossing text lines, and low quality image that do not lend themselves easily to binarization. First, it defines a local projection profile at each pixel of the image and transforms the original image into an *Adaptive local connectivity map* (ALCM), an acronym coined by the authors, that is a gray-level image in which each pixel's value represents a summation of foreground pixel intensities. By thresholding the ALCM map, text lines reveal themselves in the form of a binary image followed by a connected component analysis. Figure 2.10 shows the steps of this method to extract text lines.

In [105] Zahour et al. divide an image into non-overlapping vertical strips and compute a projection profile for each strip to segment it into blocks based on the peaks and valleys of the profiles. Just like the other method in [87], this

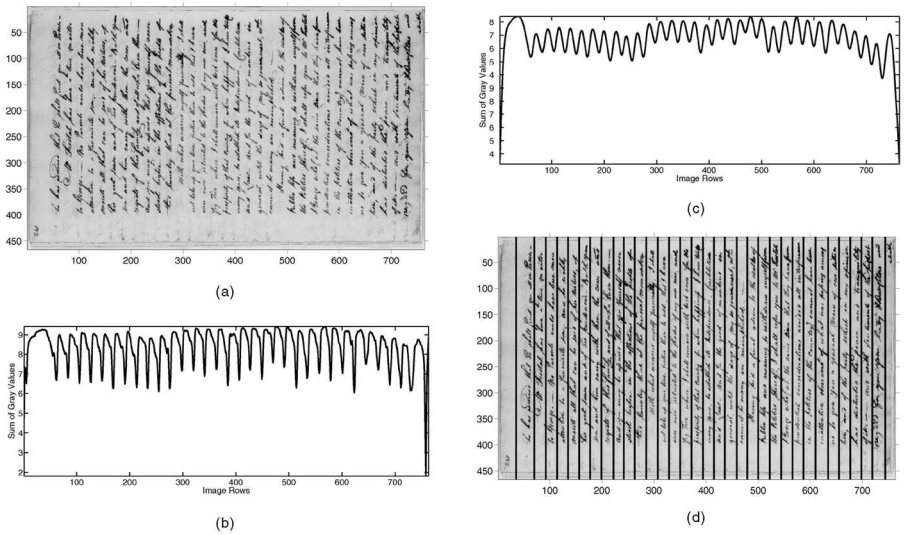
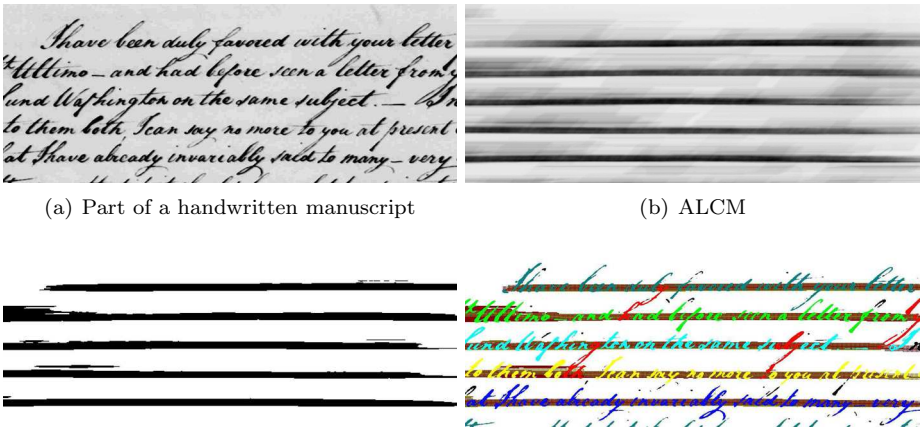


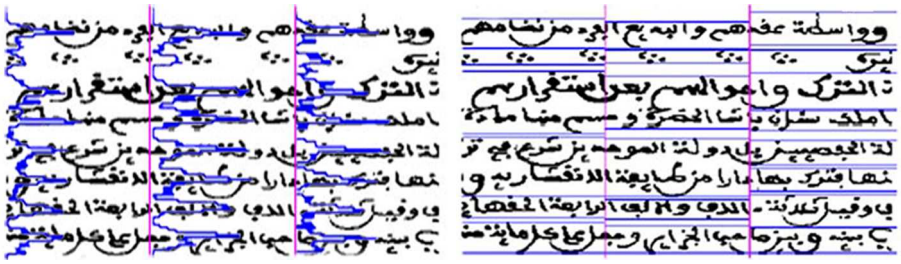
Figure 2.9: Images [62] show line segmentation steps. **a** A rotated image. **b** Vertical projection profile of the rotated image. **c** Smoothed version of the vertical projection profile. **d** Detected text line separators by detecting peaks in the smoothed projection profile.



(a) Part of a handwritten manuscript (b) ALCM

(c) Binarization of ALCM (d) Detected text lines

Figure 2.10: Text line detection method in [87] using adaptive local connectivity map.



(a) Divided strips and their projection profiles (b) Extracted blocks

Figure 2.11: Block extraction steps in [105]

method works on blocks. But the difference is that the authors assume two different types of documents, namely tightly spaced documents (TSD) and widely spaced documents (WSD) to better cope with overlapping and multi-touching components. Figure 2.11 shows the projection profiles for several strips and the result of their block extraction. For each block, the method computes two features based on fractal dimensions resulting from the classical box-counting algorithm. Then an unsupervised fuzzy C-means 2-class classifier is used to separate blocks into tightly packed or widely spaced. Each block type is approached differently for detecting text lines.

Another method is proposed by Arivazhagan et al. in [7] that starts by obtaining candidate lines using piecewise vertical projection profiles similar to the one used in [105]. Then the method draws a decision for any obstructing element based on the bivariate Gaussian probability density of a distance metric to find out whether the element belongs to the line above or the line below. After applying the piecewise projection profile to the document image, the complete set of text line separators are drawn by connecting a valley of a profile associated with a block on the right, to a valley from the block on its left and continuing the line straightly in a situation where a valley is still unconnected. For each connected component in which a line passes through, it may belong totally to either the line above or the live below, or it may need to be broken into two components. The method uses a distance metric decision to determine which approach should be taken for each obstructing component.

The last reviewed projection based method is proposed by Papavassiliou et al. [75]. This method is very similar to the one in [7] with one difference that it adds another stage based on *Hidden Markov Models* (HMM) to correct some misleading peaks and valleys of projection profiles and therefore, it segments vertical strips into better situated blocks compared to other methods. Initially, like other methods, text and gap areas are extracted by detecting peaks and valleys in a smoothed projection profile computed for each vertical strip. Then a Viterbi algorithm locates the optimal succession of text and gap areas based upon the statistics drawn from the initial set of blocks. Finally, a text line separating technique is applied to assign connected components into appropriate text lines. Results for preliminary version of this method are published in [90] and submitted to the ICDAR2007 [38] and ICDAR2009 [40] handwritten

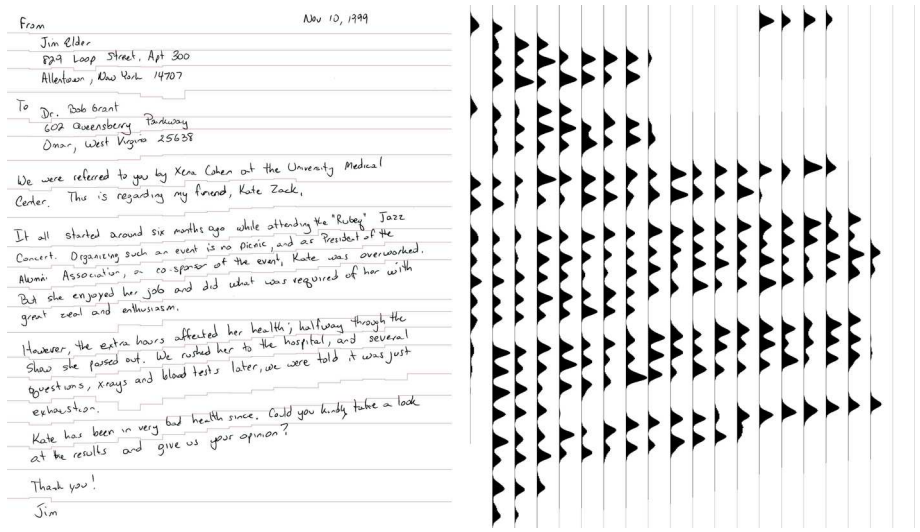
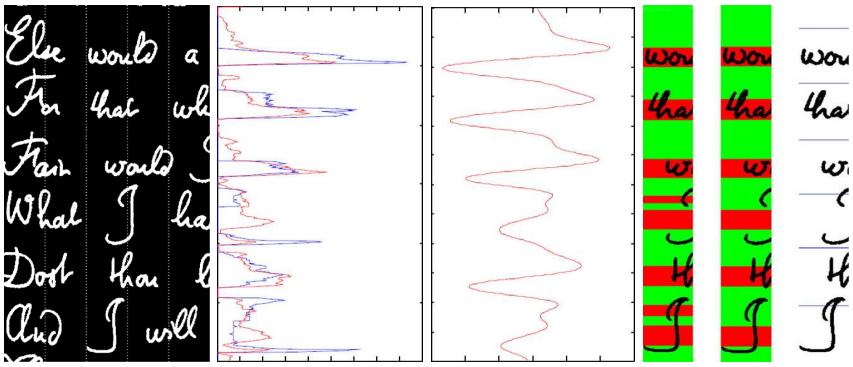


Figure 2.12: **Left** image shows segmented lines from a document in [7] and **Right** image displays projection profiles of vertical strips.

segmentation contests.

Hough-based methods can also be considered as projection based methods. They are based on Hough transform but instead of projecting pixel values into x or y axis, they project intensities onto parameter's space. In general, the purpose of Hough transform is to find imperfect instances of objects within a certain class of shapes by a voting procedure. The classical Hough transform is concerned with the identification of lines in an image, and text lines in the case of document image. However, it is not limited to lines, and any kind of shape can be found inside an image. In document image analysis, Hough transform is used in a variety of situations. Some methods use it to detect text lines or the skew angle of the text lines, or to drive different characteristics such as the direction of a connected handwritten word, or even to detect table lines. Here we note two methods that use Hough transform specifically for detecting text lines.

In [58], G. Louloudis et al. have proposed a Hough transform based handwritten segmentation method. Their earliest results have been published in [56, 57]. The method is adapted to deal with challenges of handwritten documents, including arbitrary slanted and skewed text lines, accent marks above or below the text lines and touching lines. After dividing the document image into equally sized blocks, the aim of Hough transform becomes to find dominant direction of connected components in each block. The algorithm also keeps track of the dominant direction of all components inside the document. To form text lines, for each connected component a decision is held based on rules that compare the direction of each component in a block with its adjacent blocks. An additional constraint is applied upon which, a text line is valid only if the corresponding skew angle of the line deviates from the dominant direction by



(a) Five zones 1-5 (b) Projection profile of zone 3 (c) first derivative (d) initial, refined and final regions

Figure 2.13: Steps for locating text line separators in part of document image. [75]

less than 2°. This method is successful provided that the free parameters are set correctly. Furthermore, because the algorithm keeps track of the dominant direction, the whole document must have text lines with roughly the same direction.

The method published in [61] is one more method based on Hough transform. In the first step, authors apply a Hough transform to each connected component namely the handwritten words to find the direction of a component. Then the algorithm searches for the nearest neighbors of each component in four principal directions. Once the neighbors are found, a weighted directed graph is built by connecting each component to its neighbors with a weighted edge proportional to the geometric distance between components. Finally, to form text lines, the algorithm removes top to bottom edges based on thresholding the length of edges.

Texture based methods

Any method that is based on some kind of filtering, shall it be Gabor, Wavelet, Gaussian or just the averaging operator can fit into this category.

The first method that we review is for text line segmentation from freestyle script-independent handwritten or printed documents. Y. Li et al. first have published their preliminary results for this method in [52] and later in [53]. For this method it is assumed that text lines have a horizontally elongated shape, but still a variation of $\pm 10°$ is allowed. The method estimates a probability density function based by convolving the image with a non-parametric anisotropic Gaussian kernel. The initial estimates of the text line boundaries are computed by thresholding this density function map and then a level set method evolves from the initial estimations to obtain the final text line boundaries.

Another method is proposed in [29]. Du et al. propose a script-independent method for segmentation of handwritten text lines based on a piecewise ap-

proximation of a Mumford-Shah functional. They indicate that boundary based level-set methods such as [52, 53] depend on the number of boundary evolution steps, and they are also sensitive to touching text lines. To handle these difficulties, their method seeks to minimize a Mumford-Shah functional using a piecewise constant approximation [92]. The initial estimates of the text lines are the same as [85], and then they are refined by visiting each pixel of the image in a given order. For each initial text line, a segmentation curve is set to segment the text line into two regions; inner and outer. In each iteration, these curves evolve by calculating their parameters based on the intensity of the region. The final results may contain line fragments due to large gaps between words; so morphological operators are used as part of the post-processing step to merge some of these fragments.

The last reviewed method is published in paper [88] for detecting handwritten Arabic text lines. Instead of summing values of adjacent pixels as in projection profiles in [87], Shi et al. apply steerable directional filters, each with a shape of an ellipse with a large focal distance. The height of the filter is chosen to be the same as the height of an average text and the width to be five times its height. Using a filter with a direction similar to the direction of text lines, the pixel value for that location has a greater response than when using another filter in any other direction. The result of filtering generates a map that is later thresholded adaptively to enhance the location of text lines.

Distance based methods

We only review one method in this category. This method has been recently proposed in [103] for segmentation of handwritten Chinese text regions into text lines. The heart of this method is a minimum spanning tree algorithm. In the first stage, the method extracts all the connected components of the document. The reasonable assumption is that components which belong to a single text line are close to one another compared to the components that belong to different text lines. Therefore, a minimum spanning tree is applied to connect neighboring components of the same line, and each line corresponds to a sub-tree. Then because of variability of layout of text lines and occasionally large gaps between words, the results are not prefect. Hence, the method use a second-stage clustering procedure to dynamically cut the edges of the tree into groups corresponding to correct text lines. A detection accuracy rate of 98.02% is reported for 803 unconstrained documents.

2.3.3 Conclusion

We have noted many methods in this section for text line detection, and eventually we have to decide which method is exploitable for detecting text lines from our corpus. An evaluation and comparison between methods is valid only if the results are available for the same dataset and with the same evaluation metric. We identified three different performance evaluation metrics; *Pixel Correspondence*, *Match Counting* and *Overall Pixel-Level HitRate*. We have already described Match Counting in the first chapter. Unfortunately, in our case, it

would be difficult to draw a conclusion solely based on the reported accuracy rate of each method. Also it is impossible to implement or find every method that is listed here. As a consequence we have to rely as much as possible on the reported accuracies when comparison is possible and then a theoretical and fundamental reasoning to decide which method is fit for our work. Table 2.1 shows some of the characteristics of our dataset.

Table 2.1: CHARACTERISTICS OF THE DOCUMENTS IN OUR CORPUS

Frequently occurs	Seldom occurs	Never occurs
Printed text lines	Slanted text lines	Calligraphy
Handwritten text lines	Vertical text lines	Highly skewed text lines
Large gaps between words	Degraded quality	
Rule lines	Transparency effect	
Close text lines	Multi-script documents	
Touching text lines	Black borders	
Multi-column documents		
Salt & Paper noise		
Side notes		
Variety of font sizes		

In order to choose the best line detection method as the base for our work, first we have to compare methods in a situation where both datasets and the evaluation metrics are the same, then after pruning weak methods, we have to go into details of remaining methods to reason whether they might fail to achieve good results when applied to our documents or not.

We have named fourteen methods for detection of text lines so far. Table 2.2 presents some of these methods, and the accuracy rate that they have achieved by applying to ICDAR2007 [39] dataset.

Table 2.2: ACCURACY RATES FOR METHODS APPLIED TO ICDAR2007 DATASET [39]

Method	Reported In	dataset	Evaluation Method	Accuracy
Papavassiliou 2010 [75]	[75]	ICDAR07	Match Counting	98.3%
Louloudis 2009 [58]	[58]	ICDAR07	Match Counting	97.4%
Stafylakis 2008[90]	[90]	ICDAR07	Match Counting	97.1%
Louloudis 2006 [56]	[39]	ICDAR07	Match Counting	95.4%
Bukhari 2009a [18]	[18]	ICDAR07	Match Counting	90.7%

Another table 2.3 presents the detection accuracy rate for CBDAR2007 dataset. Note that for the two methods reported there, the comparison is a challenge because one method gains the upper hand using one evaluation metric and using another evaluation metric the situation changes. To solve this dilemma, we decide to choose the evaluation metric that is more appropriate and well designed. In this case, it is Match Counting, which is the same evaluation that we have described in Appendix A. Based on this evaluation metric the result of both methods can be considered the same.

Table 2.3: ACCURACY RATES FOR METHODS APPLIED TO CBDAR2007 DATASET

Method	Reported In	dataset	Evaluation Method	Accuracy
Bukhari 2009a [18]	[18]	CBDAR07	Match Counting	90.76%
Bukhari 2009a [18]	[18]	CBDAR07	Pixel Correspondence	97.96%
Bukhari 2009 [19]	[19]	CBDAR07	Match Counting	91.05%
Bukhari 2009 [19]	[19]	CBDAR07	Pixel Correspondence	90.50%

From this set, the proposed method by Papavassiliou [75] has gained the upper hand. On the other hand, Shi has proposed two methods in [87] and [88]. Both methods are applied to DARPA/MADCAT dataset and they have achieved 95.00% and 99.5%, respectively. The evaluation metric is unclear, but since both methods are proposed by the same author, we assume that they are using the same evaluation metric. Therefore, the method in [88] is superior to the one in [87].

One more comparison between the methods proposed by Du [29] and Li [52, 53]. Both methods are applied to multi-lingual scripts from University of Maryland handwritten dataset and pixel hit rate is used as the evaluation metric. Table 2.4 summarizes the results. The results indicate that both methods are roughly equal in their power to detect text lines and indeed both are similar in the method that they use.

Table 2.4: ACCURACY RATES FOR METHODS APPLIED TO UNIVERSITY OF MARYLAND'S DATASET

Method	dataset	Evaluation Method	Accuracy
Li 2008 [53, 52]	Hindi Scripts	Pixel HitRate	97.00%
Du 2009 [29]	Hindi Scripts	Pixel HitRate	98.00%
Li 2008 [53, 52]	Korean Scripts	Pixel HitRate	98.00%
Du 2009 [29]	Korean Scripts	Pixel HitRate	96.00%
Li 2008 [53, 52]	Chinese Scripts	Pixel HitRate	98.00%
Du 2009 [29]	Chinese Scripts	Pixel HitRate	98.00%

To sum up, all other methods as well as the top methods from previous groups are gathered in one table. Unfortunately, from this point forward these methods cannot be compared quantitatively. Table 2.5 presents the results.

Table 2.5: ACCURACY RATES FOR ALL REMAINING METHODS

Method	dataset	Evaluation Method	Average Accuracy
Xiao 2003 [102]	N/A	N/A	N/A
Malleron 2009 [61]	1 Image	N/A	84.71%
Yin 2009 [103]	HIT-MW[92]	Pixel HitRate	98.02%
Manmatha 2005 [62]	George Washington Scripts	N/A	83.00%
Li 2008 ,Du 2009 [53, 52, 29]	Various Scripts	Pixel HitRate	97.5.00%
Shi 2009 [88]	DARPA/MADCAT	N/A	99.5%
Zahour 2008 [105]	N/A	N/A	N/A
Arivazhagan 2007 [7]	CEDAR-FOX [23]	N/A	97.00%
Papavassiliou 2010 [75]	ICDAR07	Match Counting	98.3%

The proposed methods by Xiao [102], Malleron [61] and Yin [103] depend on the distance between connected components. Xiao's method use a global threshold to separate edges between lines from edges between words and characters. However, in some of our documents various font sizes exist on one page. Larger characters demand more gaps between them and thus a suitable threshold that works well for part of the document may fail for the other part. Moreover, when text lines are packed close to one another, Yin's method and Malleron's method fail to find the correct text lines or text line fragments.

Manmatha's method is using global projection profiles to detect text lines. While this may work for documents that have aligned text lines, it is unsuitable for freestyle handwritten documents.

Methods proposed by Du and Li [53, 52, 29] are very sensitive to touching text lines and large gaps between words. As a consequence many text line fragments appear in segmentation results using these algorithms. Both authors have tried to utilize post-processing steps such as morphology operators to merge these line fragments. Such post-processing steps may improve the results by merging some line fragments, but on the other hand, they may also merge multi-column text lines and side notes.

The method proposed by Shi [88] works well as long as there are enough gaps between lines. When text lines are packed, it is not guaranteed that the maximum output value of several direction filters is pointing in the direction of the text line. The reason is that characters from different lines may accidentally align and form a false line.

The three remaining methods by Zahour [105], Arivazhagan [7] and Papavassiliou [75] are in the same category. All of them are using piecewise projection profiles. There is not enough evaluation data from Zahour's method, but Arivazhagan and Papavassiliou methods are quite the same. The only difference is that Papavassiliou is using a Hidden Markov Model to correct some of the detected peaks and valleys of the projection profiles before drawing line separators. As a result, his results are slightly better, and it works well for our corpus.

In our approach, we use a variant of Papavassiliou's method [75] to detect text lines. Details of our method is written in chapter 5.

Chapter 3

Text/graphics separation

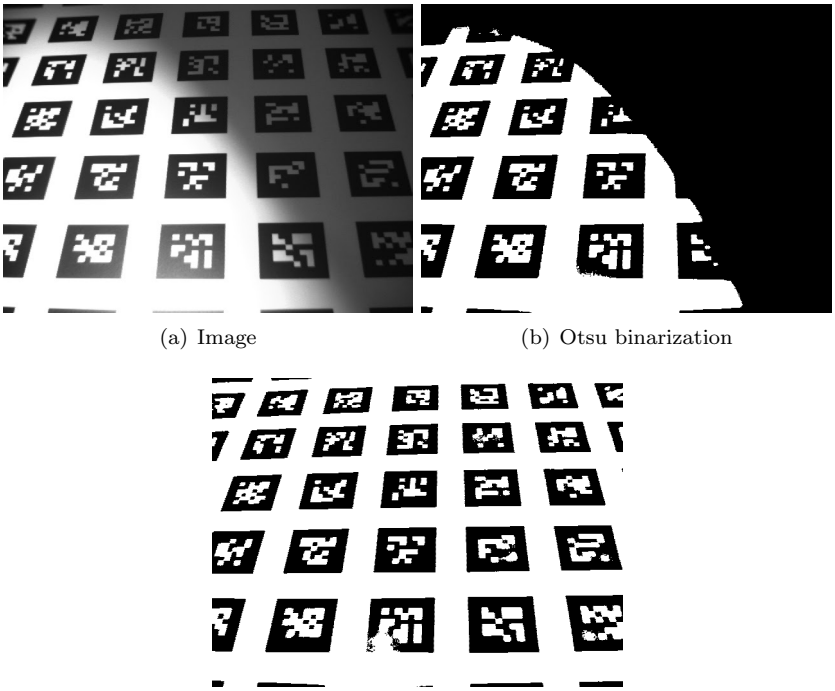
Separation of text and graphics in document images is the first step in our methodology. This is an important stage that not only improves the results of text region and text line detection by not allowing graphical drawings to be merged with text regions but also by providing assistance for separating text regions as we will see in the next chapter.

3.1 Preprocessing

Preprocessing is the first step that happens after reading a document image into memory. The aim is to prepare the document for text and graphics separation. Our method works with connected components. Each component can be a character, punctuation, noise, part of a handwritten word, rule lines or graphical elements. To extract these components, binarization first takes place. It is worth noting that different binarization methods exist. When a document image is in good condition, Otsu's binarization method performs better than other methods such as Sauvola [82, 11] or Niblack [68]. The reason is that Otsu's method often keeps all parts of graphical drawings in a single piece. On the other hand Sauvola's binarization method should be used when dealing with low quality historical documents or bad lighting conditions. Figure 3.1 shows a part of a gray-level document with bad lighting condition and compares the result of binarization by Otsu and Sauvola methods. Figure 3.2 illustrates the advantage of applying Otsu's method to documents that contain drawings and graphical components. After binarization, a connected component analysis can simply extract all connected components (CCs) of the image.

3.2 Features

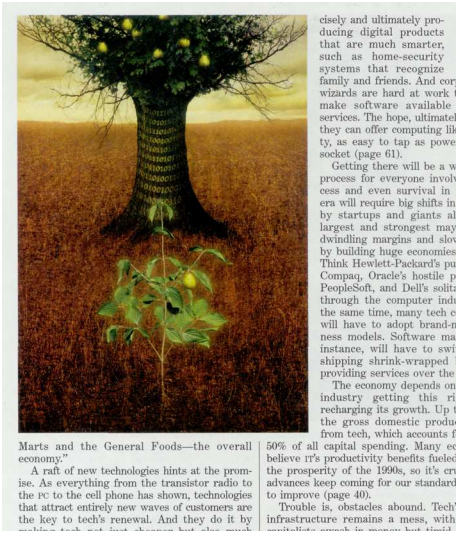
As already mentioned in the previous chapter, most methods use either block-based or component based approach for labeling components. To clarify this, if a method is assigning a label {text,graphics} to one component then it tends to extract component based features such as size, area, etc. And if the method



(a) Image (b) Otsu binarization

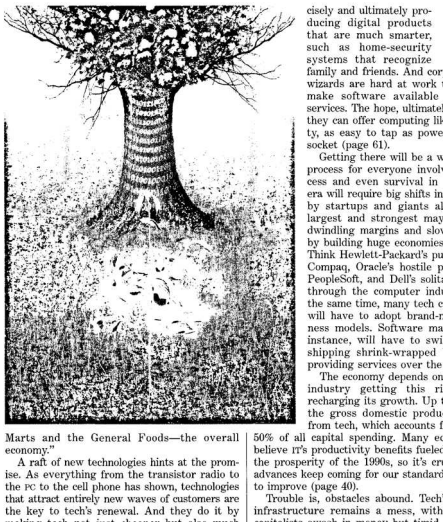
(c) Sauvola binarization

Figure 3.1: When dealing with low quality documents or documents that are captured in bad lighting condition, Sauvola's binarization method is preferable to Otsu's binarization. Image is from [11]



(a) Image

(b) Otsu binarization



(c) Sauvola binarization

Figure 3.2: Sauvola's binarization method breaks the integrity of graphical components and drawings in healthy documents and produces many small connected components that may resemble text characters. This image is part of a document from ICDAR2009 dataset.

is assigning a label to a region of the image, it tends to extract features based on texture analysis or projection profiles.

The method we describe here is a hybrid approach for separating text from graphics. It is designed to assign a label to one connected component but features are gathered based on both the characteristics of the components and the components in its neighborhood.

We consider 16 features as potential features for the purpose of separating text and graphics components. Each feature may help to discriminate components in a particular situation. The first three features are computed relative to other components of a page. Using these three features, the aim is to estimate how likely the current component is different from other components on the same page in regard to its elongation, height or solidity. The rest of the features are global features, extracted from all components of the pages from our training dataset. Height and width of a component are equal to height and width of the bounding box of the component. Elongation and solidity are defined as follows:

$$\text{elo} = \frac{min(\text{height}, \text{width})}{max(\text{height}, \text{width})}$$

and

$$\text{solidity} = \frac{\text{Number of pixels of the component}}{\text{height} \times \text{width}}$$

Considered features are:

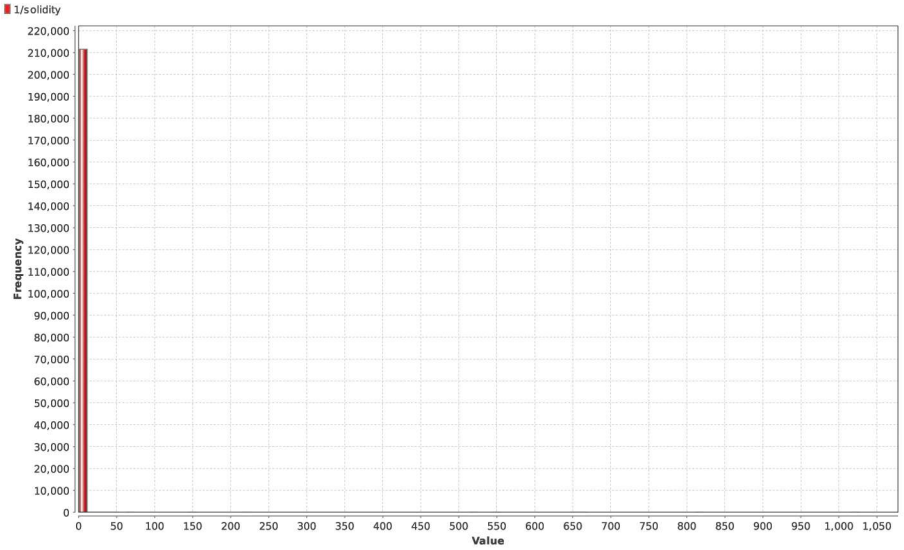
- **Log-normal distribution of 1/solidity** [1]. For most of characters, the number of black pixels divided by the area of their bounding box, namely solidity, is located in a fixed range. This property does not hold on tables, borders and many graphical drawings. Looking for outlier components of the page regarding this criterion may help in classifying them as non-textual components. Figure 3.3 shows histograms of 1/solidity and log(1/Solidity) for text components on the whole corpus which visually justify the use of Log-normal distribution.

  $$\frac{solidity}{\sigma\sqrt{2\pi}} \exp^{-\frac{(\ln x - \mu)^2}{2\sigma^2}}$$

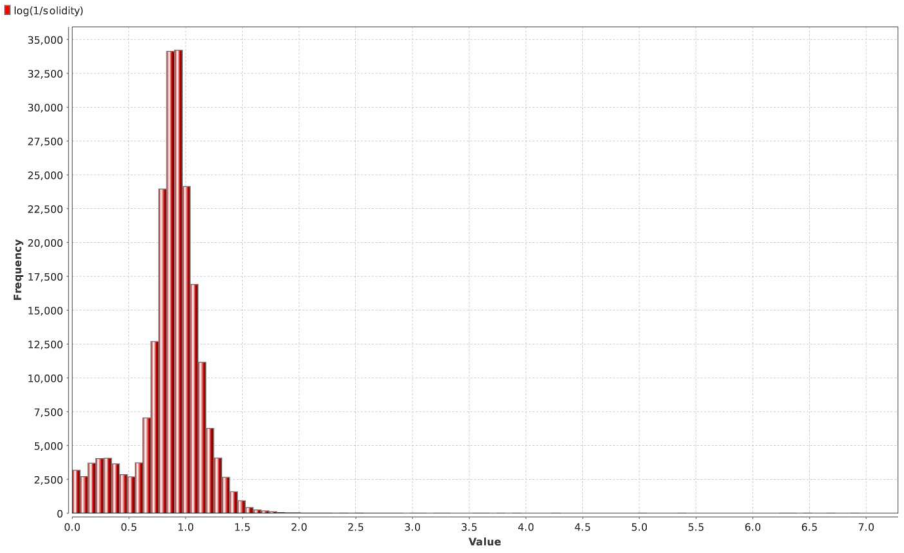
  where x is 1/solidity and μ and σ are the mean and standard deviation of 1/solidity's natural logarithm for all connected components on the page.

- **Log-normal distribution of 1/elongation** [2]. Except for some characters such as 1,l and I that resemble a very small rule line, the height and width ratio of a character, namely elongation, is located roughly in a

[1] LogNormalDist(1/solidity)
[2] LogNormalDist(1/elo)



(a) Histogram of 1/solidity of text components on the whole corpus.



(b) Histogram of log(1/solidity) of text components on the whole corpus.

Figure 3.3: Histograms of 1/solidity and log(1/solidity) of text components on the whole corpus.

fixed range. When the elongation approaches one, it represents a square. Most of characters have an elongation of about 0.57. However when the value approaches to zero, it usually represents a rule line. Figure 3.4 shows histograms of 1/elongation and log(1/elongation) for text components on the whole corpus.

$$\frac{elongation}{\sigma\sqrt{2\pi}} \exp^{-\frac{(\ln x-\mu)^2}{2\sigma^2}}$$

where x is 1/elongation and μ and σ are the mean and standard deviation of 1/elongation's natural logarithm for all connected components on the page.

- **Log-normal distribution of height** [3]. For pages that are written with a single font size, heights are roughly equal. Even for handwriting with connected scripts the width of CCs may be different but height values are still the same. This feature is looking for outliers such as large graphical elements and tables that have a considerably larger height compare to other CCs on the page. Figure 3.5 displays histograms of height of text components and its logarithm for components on the whole corpus.

$$\frac{1}{(\text{height})\sigma\sqrt{2\pi}} \exp^{-\frac{(\ln x-\mu)^2}{2\sigma^2}}$$

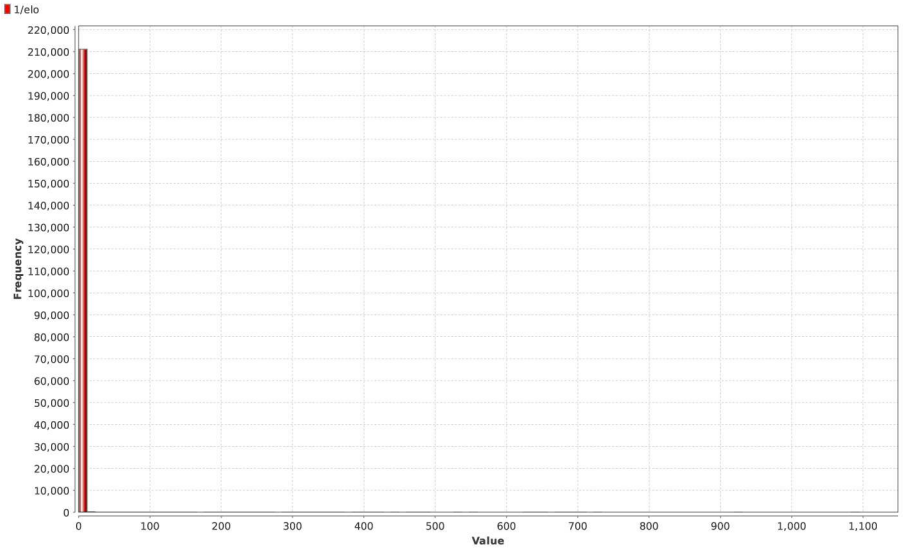
where x is the height of a component and μ and σ are the mean and standard deviation of height's natural logarithm for all components of the page.

- **Normalized X** [4] **and Y** [5] **coordinates of the component's center**. These two features have a range between 0 and 1. They are simply the X and Y parts of a component's center, divided by the width and height of the document image respectively. The reason behind using this feature is that most of the time noisy components, borders and frames are situated near the boundaries of a document. By themselves they do not provide a direct solution for locating non-textual elements but a classifier can utilize this information along side of other features to form boundaries in a feature space that can better discriminate text and non-textual components.

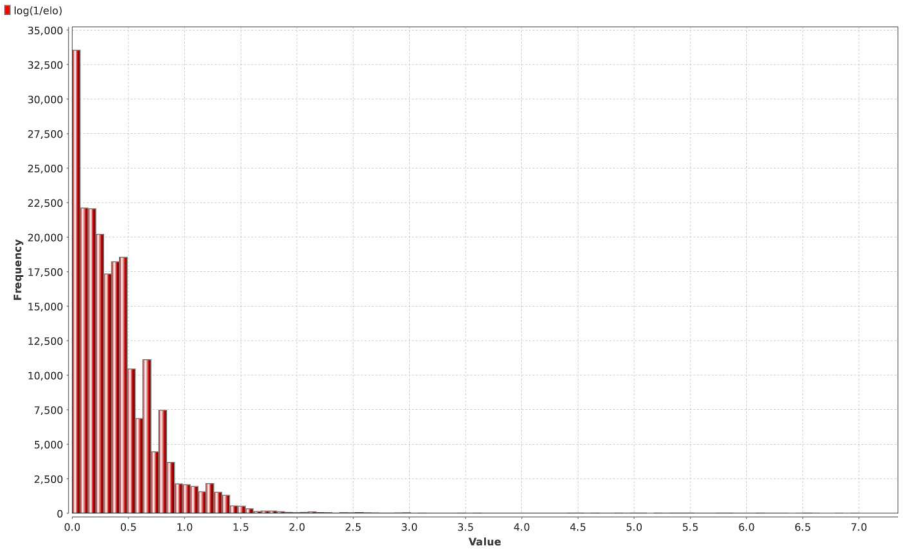
- **Logarithm of normalized height** [6] **and width** [7]. These two features are computed as follows:

$$\log \frac{\text{page's height}}{\text{component's height}}$$

[3] LogNormalDist(height)
[4] center.x/src.cols
[5] center.y/src.rows
[6] log(src.rows/height)
[7] log(src.cols/width)

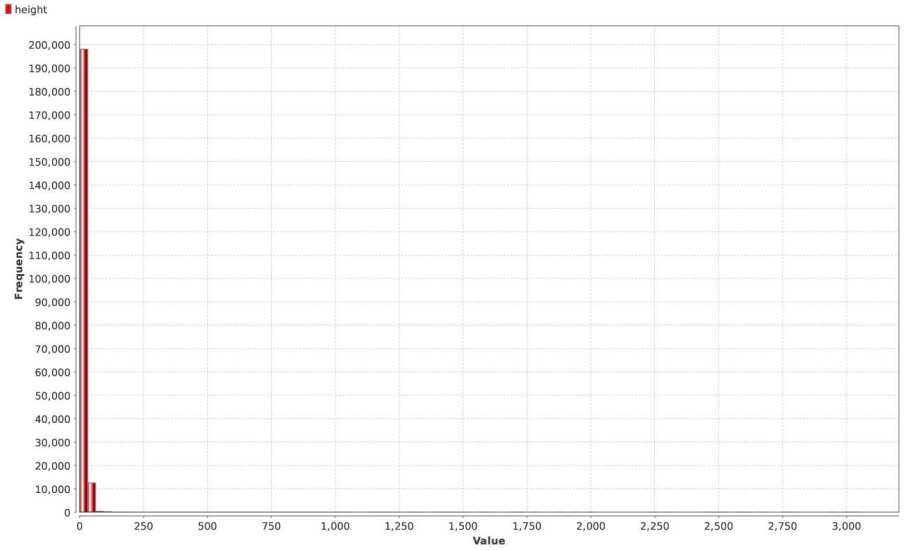


(a) Histogram of 1/elongation of text components on the whole corpus.

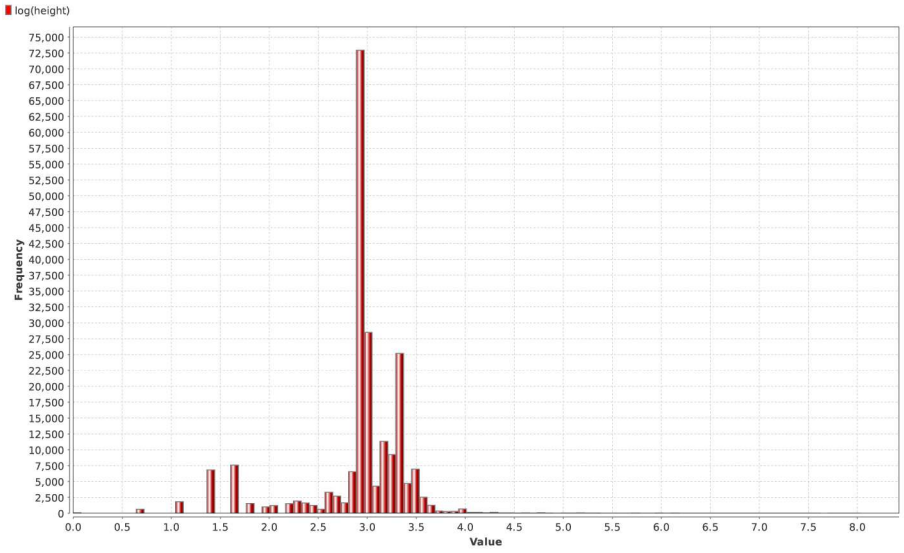


(b) Histogram of log(1/elongation) of text components on the whole corpus.

Figure 3.4: Histograms of 1/elongation and log(1/elongation) of text components on the whole corpus.



(a) Histogram of height of text components on the whole corpus.



(b) Histogram of log(height) of text components on the whole corpus.

Figure 3.5: Histograms of height and log(height) of text components on the whole corpus.

$$\log \frac{\text{page's width}}{\text{component's width}}$$

They are normalized for the reason that document pages of our training dataset come in different resolutions and sizes.

- **Logarithm of 1/solidity** [8] **and 1/elongation** [9]. The importance of solidity and elongation is noted above. Instead of computing the mean or standard deviation of this feature for components of a page, these two features are provided as raw and they allow the classifier to compare these values globally for all images of the training dataset.
- **Logarithm of Hu-moments of the component's pixels** [10]. Image moments are particular weighted average of the image pixels' intensities. They usually are chosen to have some attractive properties or interpretation. Hu moments are special moments that are proved to be invariant to the image scale, rotation and reflection. There are a total of seven popular moments in the literature. Of these moments, the seventh one is not invariant to reflection. Empirically we found that the first four moments improve the classification results.
- **Parents, children and siblings**. Usually text characters that appear on a page are not contained or do not contain any other components except when they belong to a table. On the other hand large drawings often contain many broken pieces. If a component contain another component, the former is assigned the role of a parent and the latter is assigned the role of a child. Any component that has a parent might have some other siblings. By counting the number of parents, children and siblings they serve as perfect features for our purpose.

3.3 Feature analysis

Having some features, it would make sense to know the contribution that each feature may bring to the task of classification. Moreover, if a feature has no correlation with the true labeling of the data, it may do more harm than good and thus should be pruned. To do so, we apply some feature selection methods to our dataset and their true labels and each method assigns a weight to each feature which indicates the significance of the feature based on the method. All parts of this analysis is carried out by an open-source software called *RapidMiner* [63] and for this purpose we use the dataset from ICDAR2009 page segmentation competition for the reason that it contains a good amount of irregular graphical components and tables.

Tabled 3.1 summaries the result of feature analysis by showing the obtained weights for four feature analysis methods. The first method calculates the relevance of a feature by computing the information gain in class distribution.

[8] log(1/solidity)
[9] log(1/elo)
[10] log(HuMomentX+1)

The second method calculates the correlation of each feature with the label and returns the absolute value as its weight. The third method calculates the relevance of a feature by measuring the symmetrical uncertainty with respect to the class label and the fourth method uses the coefficients of the normal vector of a linear SVM as weights.

Based on the obtained weights and considering the nature of each methods, the most important opinion comes from the weights of SVM, however there is one feature (normalized X center) that all methods have a negative opinion about. This feature should be removed before we move to classifier selection.

Table 3.1: ANALYSIS OF FEATURES FOR TEXT GRAPHICS SEPARATION

Feature	Information gain	Correlation	Uncertainty	SVM
LogNormalDist(1/solidity)	0.024	0.086	0.076	0.131
LogNormalDist(1/elo)	0.147	0.217	0.06	0.135
LogNormalDist(height)	0.112	0.131	0.015	0.042
center.x/src.cols	0	0	0	0
center.y/src.rows	0.055	0.099	0.007	0.012
log(src.rows/height)	0.586	0.32	0.105	0.201
log(src.cols/width)	0.186	0.067	0.063	0.288
log(1/solidity)	0.231	0.126	0.076	0.315
log(1/elo)	0.152	0.324	0.067	0.099
log(HuMoment1+1)	0.28	0.196	0.178	0.711
log(HuMoment2+1)	0.158	0.394	0.214	0.727
log(HuMoment3+1)	0.094	0.279	0.11	0.323
log(HuMoment4+1)	0.108	0.286	0.112	0.134
parents	1	1	1	0.607
childs	0.014	0.096	0.013	0.075
siblings	1	0.919	0.897	0.355

3.4 Classifier selection

The goal in classifier selection is to find a classifier for our feature set that generalize well with the data. In other words it should avoid over-fitting and perform well for unseen data. To do so, we choose several available classifiers and train each classifier on our dataset to find the most fitted one for our purpose.

The dataset used for this mean is again ICDAR2009 dataset for page segmentation competition. This dataset contains 55 documents which translates to 211585 connected components of which about 6000 are graphical components. Clearly, the dataset is biased toward text components and the amount of text components is unnecessary large. To speed up the computation we keep all the graphical components but we sample 12000 components from those with true text label. To evaluate classifiers, it is necessary to use cross-validation. Here we use a 5 folds cross-validation. In a 5 folds cross-validation the whole dataset is divided randomly into 5 equal parts. Each time, one part is used for testing purpose and the other four parts are used for training. This process occurs five times and the average performance is used for the result of each classifier evaluation.

Table 3.2: AVERAGE PERFORMANCE OF SEVERAL CLASSIFIERS FOR TEXT GRAPHICS SEPARATION

Classifier	Text recall	Graphics recall	Text precision	Graphics precision	Accuracy	Comments
LogitBoost	93.08	93.03	92.43	92.38	92.73%	500 weak learner
Decision Tree (gini-index)	95.33	88.19	90.47	94.14	92.94%+/-0.66	
GentleBoost	93.32	90.63	90.92	93.12	91.97	
GiniBoost	93.24	90.01	90.34	93.01	91.62	
AutoMLP	95.32	80.54	90.7	89.64	90.38%+/-0.45	51 neurons
Linear C-SVC	98.21	57.06	81.99	94.12	84.45%+/-0.33	SV: 6470, C = 1.0
Decision Tree (accuracy)	96.65	60.69	83.03	90.1	84.62%+/-2.61	
LDA	98.99	46.86	78.76	95.89	81.56%+/-0.48	
Nave Bayes	98.67	43.26	77.58	94.22	80.14%+/-0.77	
Linear C-SVC	99.81	37.68	76.12	99	79.03%+/-0.31	SV: 7942, C = 0.0
Decision Tree (gain-ratio)	99.82	15.43	70.12	97.68	71.57%+/-6.34	

Based on our evaluation, LogitBoost classifier is selected.

3.5 LogitBoost for classification

Boosting is a machine learning algorithm for performing supervised learning. It iteratively combines the performance of many weak classifiers to produce a final strong classifier. A weak classifier is only required to be better than chance, however the combination of them, results in a strong classifier which often outperforms most strong classifiers such as Neural Networks and SVMs [37].

We use boosted decision trees which are popular weak classifiers used in boosting scheme. Our boosted model is based on a large number of training examples (x_i, y_i) with $x_i \in \mathbb{R}^K$ and $y_i \in \{-1, +1\}$. x_i is a vector with K elements ($K = 15$ is the number of our features). The desired two-class output is encoded as -1 and $+1$.

Various boosting methods are known such as AdaBoost [35], LogitBoost and GentleBoost [36]. Regarding the overall structure, all of them are similar. Initially each instance of data in our training dataset is assigned the same weight as the other data points. Then a weak classifier is trained on the weighted training data. A weight indicates the importance of a data point for the classification. In each iteration the weights of misclassified data are increased and the weights of each correctly classified sample are decreased. As a result the new classifier focuses on the examples which have eluded correct classification in previous iterations.

LogitBoost is very similar to the original AdaBoost algorithm. If one considers AdaBoost as a generalized additive model and then applies the cost functional of logistic regression, LogitBoost algorithm can be derived. The implementation of LogitBoost comes from OpenCV machine learning library.

LogitBoost algorithm is as follows:

1. Given N instances (x_i, y_i) with $x_i \in \mathbb{R}^K, y_i \in \{-1, +1\}$
2. Start with weights $w_i = 1/N, i = 1, ..., N, F(x) = 0$ and probability estimates $p(x_i) = \frac{1}{2}$

3. Repeat for $m = 1, 2, ..., M$

   - (a) Compute the working response and weights

$$z_i = \frac{y_i^* - p(x_i)}{p(x_i)(1 - p(x_i))}.$$

$$w_i = p(x_i)(1 - p(x_i)).$$

   - Fit the classifier $f_m(x) \in \{-1, +1\}$, by a weighted least-squares regression of z_i to x_i using weights w_i.
   - Update $F(x) \leftarrow F(x) + \frac{1}{2} f_m(x)$ and $p(x) \leftarrow e^{F(x)}/(e^{F(x)} + e^{-F(x)})$.

4. Output the classifier $\text{sign}[F(x)] = \text{sign}[\sum_{m=1}^{M} f_m(x)]$.

3.6 Post processing

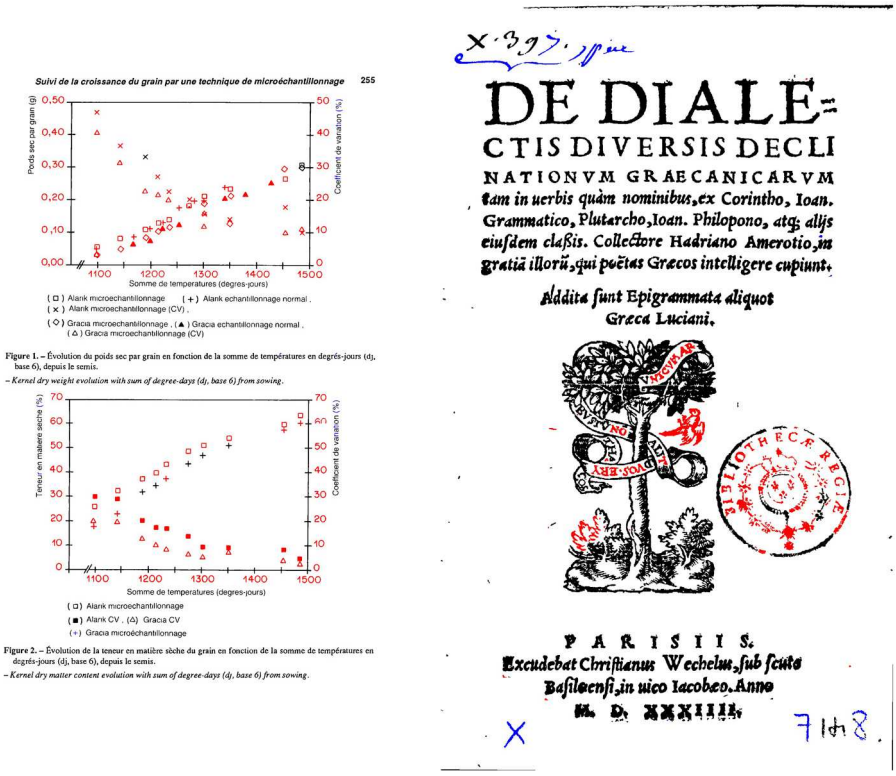
Based on our experiments, whenever a component is classified as graphics and contains many other components, it goes into one of the two possible scenarios. Either the component is a large graphics that contains many other broken parts, or it is a frame or table that holds text characters. Because tables and frames occupy a large area but the actual black pixels are far much less than this area, they can be easily identified with their solidity feature. We choose a small solidity value as the threshold that recognize tables from graphics. If the component has a larger solidity than the threshold, it is identified as graphics and all its children should also assigned a graphics label regardless of the label that are assigned to them by LogitBoost Classifier. Otherwise the children retain their original assigned labels.

3.7 Results

Results of our text/graphics separation comes in two flavors. In the first part we evaluate the performance of our classifier on two datasets; dataset for ICDAR2009 page segmentation competition and dataset for ICDAR2011 historical document layout analysis competition. Table 3.3 summaries these results. Note that the cross-dataset evaluation results is also provided in the table. In cross-dataset evaluation the system is trained on one dataset but is tested on the other with totally different type of graphic structure. The indicated values are component based and measure the precision and recall of component classification regarding its class label.

Table 3.3: EVALUATION OF LOGITBOOST ON TWO DATASETS FOR TEXT/GRAPHICS SEPARATION BEFORE POST-PROCESSING

Trained on	Tested on	Text recall	Graphics recall	Text precision	Graphics precision	Text Accuracy
ICDAR2009	ICDAR2009	99.88%	62.66%	98.91%	94.02%	98.82%
ICDAR2009	ICDAR2011	81.31%	57.2%	96.23%	18.53%	79.65%
ICDAR2011	ICDAR2011	99.15%	57.69%	96.92%	83.6%	96.29%
ICDAR2011	ICDAR2009	98.40%	45.928%	98.4%	41.98%	96.65%



(a) 73.74% (b) 79.98%

Figure 3.6: Two documents that have obtained the lowest accuracy rate for text/graphics separation. Black components are labelled correctly. Red components should have been assigned a graphics label but they are incorrectly labelled as text. Blue components on the other hand have text label in ground-truth, however they are misclassified as graphics.

In the second part of the evaluation, we report the accuracy of text/graphics separation per document. 78 documents from our own corpus are selected and after applying text/graphics separation, each document obtains an accuracy rate that indicates the percentage of components that are labelled correctly. The average accuracy rate for 78 documents is 96.30% according to area weighted match counting A criterion. Figure 3.6 displays two documents that have obtained the lowest accuracy of 73.74% and 79.98%. In this figure all components in black are labelled correctly. Red or blue components indicate that the label was supposed to be graphics or text respectively, but they are labelled incorrectly.

The majority of errors are either due to misclassification of noise, punctuations or part of drawings classified as text. The low graphics recall rates are mostly due to broken drawings and the majority of them are corrected in post-processing stage. Figure 3.7 displays example of errors that occasionally happen in documents.

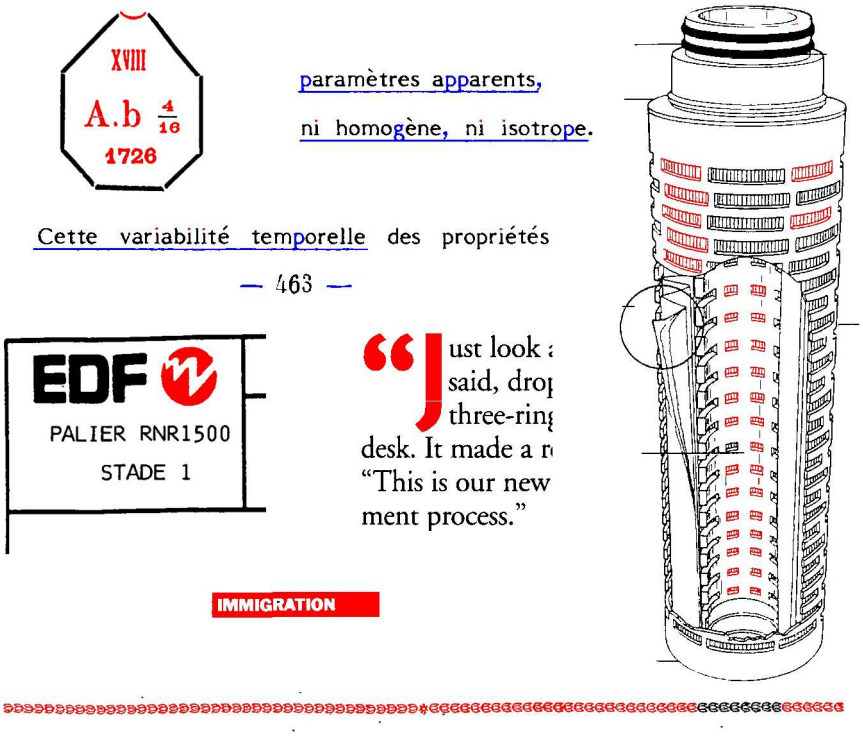


Figure 3.7: Some of the misclassified components gathered from our own documents. Black components are labelled correctly. Red components should have been assigned a graphics label but they are incorrectly labelled as text. Blue components on the other hand have text label in ground-truth, however they are misclassified as graphics.

A serious challenge in some documents is a problem that arises due to underlines. Underlines that appear in the middle of a text region as shown in figure 3.7, pose two problems. These underlines are treated as graphical components and are removed from the set of text components, but in text region detection, they are utilized to separate region of text. This behavior is expected from a true graphical component, but an underline in the middle of a text region may split the region into two which is an understandable side effect in this situation. Moreover, in some situations where text characters are attached to the underline, not only the underline disappear from the text region, it takes some characters with it and leaves large gaps in the middle of a text region. This has a negative effect on our region detection stage when it happens.

Here is another comparison between the results of the method , described here and the results of text and graphics separation from Tesseract-OCR and EPITA methods. The classifier for our method is trained on 26 documents, selected from both ICDAR2011 and our corpus datasets. Tables 3.4,3.5 and 3.6 show the results.

In conclusion, this chapter provides a method for separating text/graphics components with good separation accuracy.

Table 3.4: COMPARISON OF TEXT/GRAPHICS SEPARATION WITH LOGITBOOST, TESSERACT-OCR AND EPITA ON ICDAR2009 (61 DOCUMENTS)

Method	Text precision	Text recall	Graphics precision	Graphics recall	Text Accuracy
LogitBoost	97.45	98.04	79.21	88.00	97.52
TesseractOCR	93.32	95.44	88.52	85.87	92.96
Epita	94.95	96.25	81.62	92.45	95.78

Table 3.5: COMPARISON OF TEXT/GRAPHICS SEPARATION WITH LOGITBOOST, TESSERACT-OCR AND EPITA ON ICDAR2011 (100 DOCUMENTS)

Method	Text precision	Text recall	Graphics precision	Graphics recall	Text Accuracy
LogitBoost	98.05	93.42	56.58	73.52	94.22
TesseractOCR	94.76	87.60	84.66	94.08	90.16
Epita	97.85	95.43	62.33	85.29	95.23

Table 3.6: COMPARISON OF TEXT/GRAPHICS SEPARATION WITH LOGITBOOST, TESSERACT-OCR AND EPITA ON OUR CORPUS (97 DOCUMENTS)

Method	Text precision	Text recall	Graphics precision	Graphics recall	Text Accuracy
LogitBoost	93.82	93.62	59.41	74.11	92.40
TesseractOCR	88.58	95.90	76.80	63.15	89.90
Epita	95.75	90.20	61.28	85.23	91.20

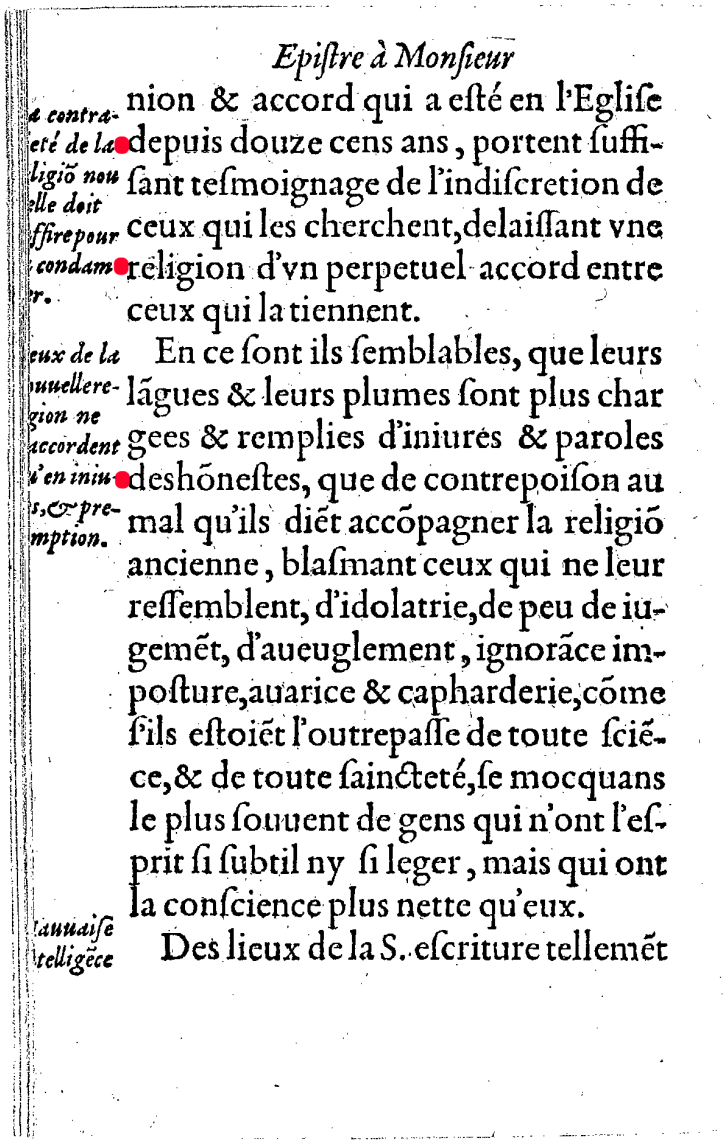
Chapter 4

Region detection

After obtaining two separate images, one containing textual components and the other containing graphical components, we apply a region detection method to separate text regions. It is already mentioned that many authors do not separate text and graphics. They directly apply their methods to segment regions of text and the decision to assign which region is text and which is graphics, comes after segmentation. As mentioned before, when parts of a graphical drawing are located close to text characters and are aligned with them, they may be merged incorrectly as part of the text regions. Other authors do apply a text and graphics separation, and then overlook all graphical components and apply text line detection or region detection only on textual components. Our method incorporates both textual components as well as graphics. The method is designed to take advantages of graphical components such as rule lines to separate text regions.

Although in many situations, rule lines may be helpful to separate text regions or columns of text, there are many examples where rulers are simply not available. Many authors have overcome this problem by using a distance-based approach or analysis of white space with success. However, still the challenge exists when columns of text are located very close to one another. One hard case of such an example is to separate side notes from the main text body. The problem is that in most documents from our corpus, side notes appear so close to the main text that a distance-based method alone, fails to separate them. We propose a framework based on two-dimensional *Conditional random fields* (CRFs) to separate regions of text from one another that also aims to separate side notes and text strings inside a table structure.

The first motivation to use CRFs compared to other locally trained machine learning methods is the long-distance communication between sites in different parts of the image. In figure 4.1 three regions are shown in red. By only considering the local information around each of these regions separately, it would be difficult to recognize these regions as gaps between side notes and the main text. However, by taking advantage of the CRFs' long-distance message passing between these regions and considering the alignment of same-labelled regions, column separators can be easily detected.



Epiſtre à Monſieur

e contra-eté de la ligiõ nou elle doit ffire pour condam r.

nion & accord qui a eſté en l'Egliſe depuis douze cens ans, portent ſuffiſant teſmoignage de l'indiſcretion de ceux qui les cherchent, delaiſſant vne religion d'vn perpetuel accord entre ceux qui la tiennent.

eux de la uuelle re- gion ne accordent 'en iniu s, & pre- mption.

En ce ſont ils ſemblables, que leurs lãgues & leurs plumes ſont plus chargees & remplies d'iniures & paroles deshõneſtes, que de contrepoiſon au mal qu'ils diẽt accõpagner la religiõ ancienne, blaſmant ceux qui ne leur reſſemblent, d'idolatrie, de peu de iugemẽt, d'aueuglement, ignorãce impoſture, auarice & capharderie, cõme ſ'ils eſtoiẽt l'outrepaſſe de toute ſciẽce, & de toute ſainctẽté, ſe mocquans le plus ſouuent de gens qui n'ont l'eſprit ſi ſubtil ny ſi leger, mais qui ont la conſcience plus nette qu'eux.

lauuaiſe telligẽce

Des lieux de la S. eſcriture tellemẽt

Figure 4.1: Long-distance communication between image sites in CRFs.

The next advantage of using CRFs is that they provide a foundation that easily induces global, local and regional knowledge of the document in the process. Because of the conditional nature of the system, any interaction between the fields' labels in a Markov network can be learned from available knowledge.

In this chapter we first introduce two-dimensional CRFs. Next, we describe feature functions which are the most important building blocks of CRFs. Then we note different types of observations that we extract from a document image. These observation are used inside our feature functions. This chapter also includes methods for decoding optimal label configuration for two-dimensional CRFs and methods for training parameters of the model. Finally, we note the results.

4.1 Conditional random fields (CRFs)

First appeared in the domain of natural language processing, conditional random fields are proposed by Lafferty et al. [50] as a framework for building probabilistic models to segment and label sequence data. Example of such sequence data can be found in a wide variety of problems in text and speech processing such as part-of-speech (POS) tagging.

Among probabilistic models that perform the same task, we can name *Hidden Markov Models* (HMMs) [78] that are well understood and widely used throughout the literature. HMMs identify the most likely label sequence for any given observation sequence. They assign a joint probability $p(x, y)$ to pairs of observation (x) and label (y) sequences. To define a joint probability of this type, models must enumerate all possible observation sequences which is intractable for most domains, unless the observation elements are independent of each other within the observation sequence. Although this assumption is appropriate for simple toy examples, most practical observations are best represented in terms of multiple features with long-range dependencies. CRFs address this issue by using a conditional probability $p(y|x)$ over label sequence given an observation sequence, rather than a joint distribution over both label and observation sequences.

CRFs first appeared in the form of chain-conditional random fields. In other words, several fields are connected in a sequential format and the label of each field depends on the label of the field on its left and on the whole observation sequence. This model best fits for applications in signal and natural language processing in which the data appear naturally in a row format. In our application, we deal with images, which can be expressed naturally in two dimensions. Thus, we are interested in two-dimensional conditional random fields.

To obtain our two-dimensional random fields, we first divide the document image into rectangular blocks with equal heights and widths. We call each block a site. Contrary to other CRFs that use sites with fix sizes for all document images, we choose the height and width of sites to be identical to half of the mean of heights and widths of all text characters on the document, respectively.

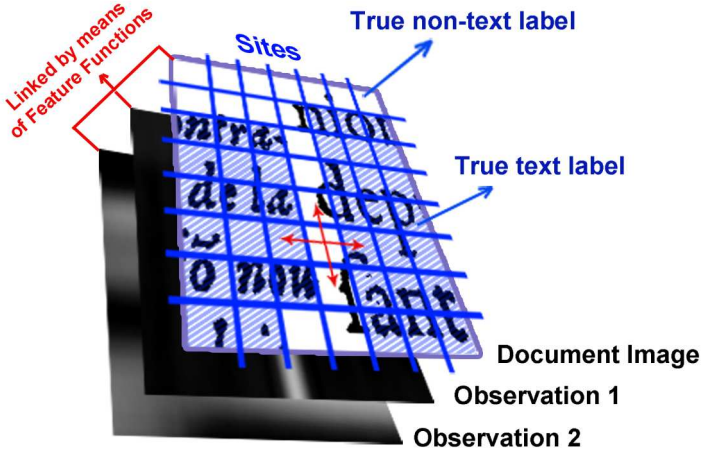


Figure 4.2: This figure shows our two-dimensional conditional random fields model. Blue lines represent the boundaries between site. We divide the document image into rectangular blocks with equal heights and equal widths. The label on each site depends on the label of sites in its vicinity and observations defined for that site. Sites depend on observations by means of feature functions. The ground truth label for each site is also available for the purpose of training and evaluation. Note that the area that is shown on the image is part of a document that shows text lines from the main text and part of a side note. Width and height of sites in this image are not correct according and are shown for visualization purposes only.

The reason is that documents come in different sizes and resolutions, and the size of each site must be normalized and the width should be small enough to pass between side notes and the main body. Each site may take one of the two labels; "text" or "non-text". However, the label of each site depends on the labels of the sites in its vicinity; this includes the sites on its left, right, top and bottom. Furthermore, the label of each site may depend on observations from the document image. Generally, in conditional random fields, there is no restriction on where these observations come from. However, we restrict the observations to be the result from several filtering operations on the document image under the current site. Figure 4.2

Let $x_s = \{x_1, ..., x_N\}$ be the vector of observations available to site s and y_s be one of the labels $\{\text{text}, \text{non-text}\}$. For each site s, the conditional probability of having a label y_s given observations x_s is defined as:

$$p_s(y_s|x_s) \propto exp\left(\sum_{k=1}^{F^e} \lambda_k f_k^e(y_{i,i\in N(s)}, y_s, x_{i,i\in N(s)}, x_s) + \sum_{k=1}^{F^n} \mu_k f_k^n(y_s, x_s)\right)$$

where f^n and f^e are the node and edge feature functions,respectively, which are discussed later in section 4.2. F^e is the total number of edge feature functions and F^n is the total number of node feature functions. $N(s)$ is the set of neighbors of the site s and is often called Markovian blanket of s. λ and μ are

weights that are assigned to edge and node feature functions, respectively.

A node feature function f^n judges how likely a label y_s is assigned to site s given the observations x_s. An edge feature function f^e indicates how likely two adjacent sites have particular label configuration given the observations at both sites. Feature functions and their weights can take any arbitrary real value. With a slight abuse of notation, it is usual to write this probability in a compact form as below:

$$p_s(y_s|x_s) \propto exp\left(\sum_{k=1}^{F} \lambda_k f_k(y_{i,i\in N(s)}, y_s, x_{i,i\in N(s)}, x_s)\right)$$

where $F = F^e + F^n$ is the total number of feature functions. According to the Hammersley-Clifford theorem (1971), the conditional probability of a set of sites given a set of observations is proportional to the product of potential functions on cliques of the graph. If we take cliques to be pair of adjacent sites then:

$$p(y|x) = \frac{1}{Z}\prod_{s\in S} exp\left(\sum_{k=1}^{F} \lambda_k f_k(y_{i,i\in N(s)}, y_s, x_s, x_{i,i\in N(s)}, x_s)\right)$$

The scalar Z is the normalization factor, or partition function, to ensure that the probability is valid. It is defined as the sum of exponential terms for all possible label configurations. Computing such partition function is intractable for most applications due to computational costs. As a consequence, by assuming conditional independence of y's given x's, the conditional probability of the CRF model for the whole image can be defined:

$$p(y|x) = \prod_{s\in S} \frac{1}{Z_s} exp\left(\sum_{k=1}^{F} \lambda_k f_k(y_{i,i\in N(s)}, y_s, x_s, x_{i,i\in N(s)}, x_s)\right)$$

Then Z for each site becomes:

$$Z_s = \sum_{(y_1,y_2)\in Y^2} exp\left(\sum_{i=1}^{F} \lambda_i f_i(y_1, y_2, x_s, x_{N(s)})\right)$$

where $Y = \{\text{text}, \text{non-text}\}$. Notice that the summation in partition function is still composed of 512 terms for a 3×3 sites.

4.1.1 The three basic problems for CRFs

Given the CRF model of the previous section, there are three basic problems of interest that must be solved for the model to be useful in real-world applications. These problems are the following:

- **Problem 1:** Given all the observations x, the label configuration y and a model $\psi = (f, \lambda)$, how do we efficiently compute $p(y|x, \psi)$, the conditional probability of label configuration given the model? (Marginal inference)

- **Problem 2:** Given the observations and the model $\psi = (f, \lambda)$, how do we choose a corresponding label configuration $y^* = y_1^* y_2^* ... y_s^*$ which is optimal in some sense (i.e., best "explains" the observations)? (Label decoding)

$$y^* = \arg\max_y p(y|x, \psi)$$

- **Problem 3:** Given the observations, the label configuration $y = y_1 y_2 ... y_S$ and a model $\psi = (f, \lambda)$, how do we adjust the model parameters λ to maximize $P(y|x, \psi)$? (Training)

We can also view the first problem as one of scoring how well a given model matches a given observation sequence. This viewpoint is useful for the purpose of recognition where we are trying to choose among several competing models the model that best matches the observations. Thus, it does not play any part in our system where we want to segment regions by labeling the observations.

Problem 2 is the one in which we attempt to find the "correct" label configuration. Since we are using two labels text, non-text for each site, the problem becomes to assign one of these labels to each site on the image. This is what we want to do every time that we process a document for region detection. In case of linear-chain conditional random fields, this operation is easily achieved by using the Viterbi algorithm [97] that can assign the optimal labels using dynamic programming. However, in two-dimensional random fields approximate techniques should be used.

Problem 3 is the one in which we attempt to optimize the model parameters to best describe how the given observations and label configuration come about. It is called training, and various methods have been proposed for that. Among these, *Limited Memory Broyden-Fletcher-Goldfarb-Shanno* (L-BFGS or LM-BFGS) [20] and *Collin's Voted Perceptron* [26] are two popular chooses.

4.2 Feature functions

Perhaps feature functions are the most important components of the model. The general form of a feature function is $f(y_{i,i\in N(s)}, y_s, x_{i,i\in N(s)}, x_s)$, which looks at a pair of adjacent sites to indicate how likely the given label configuration is correct given the observations at both sites. Because this feature function depends on two sites, it is called an edge feature function. However, if the feature function only depends on the label and observation at one site, it is called a node feature function. The value of the function is a real value that may depend on labels and observations, including any non-linear combination of them.

The term "feature function" is different from features and feature extraction we are familiar with in image processing. Each feature function must be tied to label configurations. For example, we can define a simple edge feature function f_1 which produces binary values: it is 1 if the label of the current site is "text" and the label of the site on its left is "non-text".

$$f_1(y_s, y_{i,i\in N(s)}) = \begin{cases} 1 & \text{if } y_s = \text{text and } y_{\leftarrow} = \text{non-text} \\ 0 & \text{otherwise} \end{cases}$$

where $y_{\leftarrow}$ is the label for the site on the left of s. Of course in the training phase, the weight associated with feature function f_1 has a correlation with the number of times that a site s has a "text" label and the site of its left has a "non-text" label given all sites in the training dataset. If the training algorithm finds that the number of times a site with a "non-text" label appears at the left side of a site with a "text" label, is greater than all other label configurations combined, then the associated weight for this feature obtains a positive value. In any other case, the weight would be zero or negative.

Now we look at a more complex feature which produces real values. In the previous chapter, all text and graphical elements were separated. We render two separate images; one for each. Imagine that we pick the image (G) with all graphic components which also includes rule lines and table lines. Every pixel on this image has a value of 1 for every pixel of the graphical components, and the rest of the pixels have a value of 0. In the new node feature function f_2, if the current site has the "non-text" label, the value of the feature function is the average of the intensity values of the pixels of the graphical image covered by the current site.

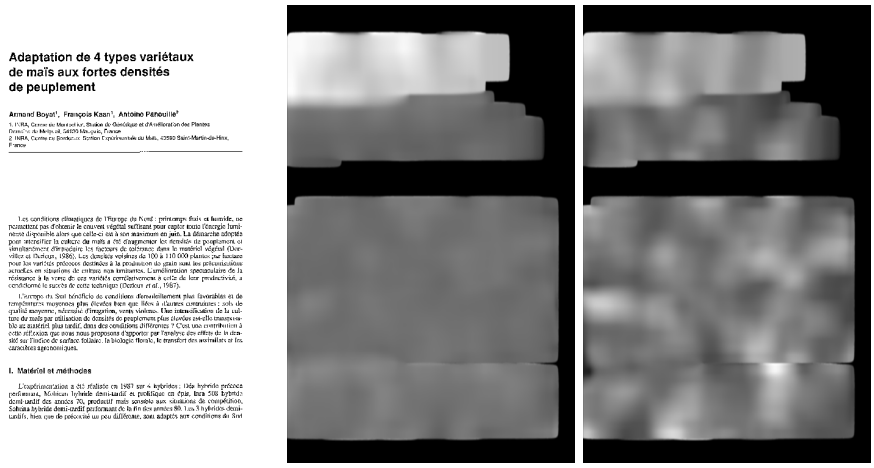
$$f_2(y_s, G) = \begin{cases} G_s & \text{if } y_s = \text{non-text} \\ 0 & \text{otherwise} \end{cases}$$

where G_s is the average value of pixels on image G covered by the site s. The value of feature function f_2 is always positive for sites that are located on graphical drawing and zero elsewhere. Thus, the λ_2 weight for this function is guaranteed to have a positive value. Suppose that we change the non-text label in f_2 to text, then the λ_2 is guaranteed to have a negative value after training.

Often good feature function engineering can significantly increase the labeling accuracy of the model. We will describe our observations thoroughly later in this chapter.

4.3 Observations

We described that feature functions in CRFs are functions that depend on both the observations and labels at one or two sites. In other words, labels and observations are tied to each other in a feature function. But, before we design our feature functions, we need to explain where our observations come from. Given a document image, we often perform operations such as filtering or run-length analysis on the image. Then each site in our model has access to the results from these operations and by knowing the location of all the pixels on the site, it makes an average estimate of the pixel values from the result of each operation and generates a vector of observations. Later, we will bind these observations with appropriate labels to be used in our feature functions. In this section, we describe all the operations that lead to our observations.



(a) Original image (b) Height map (c) Width map

Figure 4.3: Height and width maps

4.3.1 Height and width maps

These two maps represent the local average height and width of connected components in a document image. We do not use them directly in feature functions; instead, we use them as part of the computation for a distance-based feature. Each pixel of a width/height map is the weighted average of the height or width of all text-labeled connected components in a 300×300 block around it.

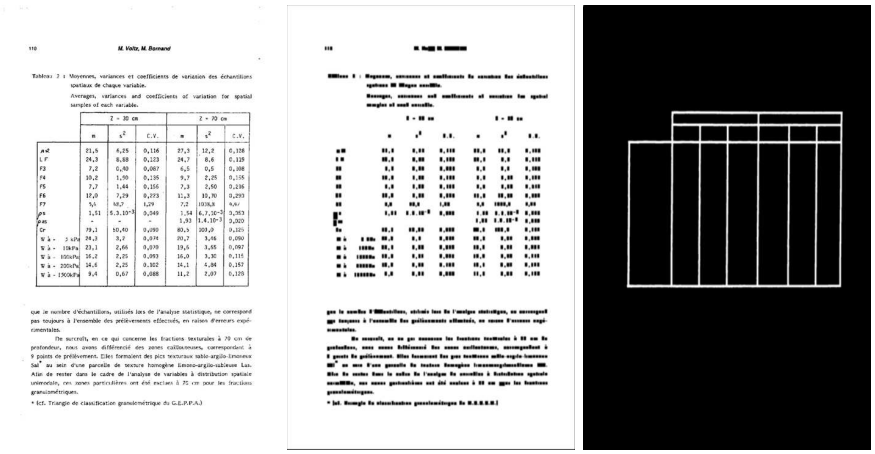
Each map can be computed using the results of two image convolutions that are divided pixel by pixel. Consider that our goal is to generate the height map. We know the location of pixels for all text-labeled components of the page. We first prepare two sparse images. The first image A has height of a component for pixels of text components, and the rest of pixels are zero. The second image B has a value of one for every pixel of text components, and the rest of the pixels are zero. We also have a weighting function K of size 300×300 pixels defined as:

$$K(x, y) = \cos(\frac{\pi i}{300}) \cos(\frac{\pi j}{300})$$

where x and y are the coordinates of pixels with $(0, 0)$ being the center of the window. Then

$$\text{HeightMap(i,j)} = \frac{(K * A)_{i,j}}{(K * B)_{i,j}}$$

The width map can be computed in the same way but instead of having the height of connected components in the A image, we use widths. Figure 4.3 displays one document image and its height and width map. We also normalize our height and width maps by dividing every pixel's value to the height and width of the document, respectively.



(a) Document (b) Filled text components (c) Graphical components

Figure 4.4: A document image, its filled text and graphical components, separated using our text/graphics separation method.

4.3.2 Text components

We rendered image of text components. A morphological hole-filling algorithm is applied to text components without any restrictions to ensure that there are no holes in them. This also enhances the results of applied Gabor filters that aim to capture text lines. Figure 4.4 shows a document image with all its filled components. Note that tables are not filled because they are already taken out for being graphical components.

4.3.3 Graphical components

Just like text components we also use graphical components in our feature function. Figure 4.4 also displays the extracted graphical components. Pixels that belong to graphical components have a value of one, and the rest of the pixels have a value of zero.

4.3.4 Horizontal and vertical run-lengths

Vertical and horizontal run-lengths are two other features computed for each document image. Consider a document image containing plain black text on a solid white background. In vertical run-length map, each white pixel of the background is replaced with the number of white runs, which are confined vertically between two black pixels of text. Then we normalize this map by dividing each pixel by the height of the image.

According to this definition, pixels that belong to page margins have a maximum possible value of 1 and pixels that belong to text components have a value of zero. We use the same strategy to compute horizontal run-lengths. By

thresholding vertical run-lengths, we are able to find the margins of the page. Margin map is another observation that we compute by thresholding the vertical run-lengths map to find its maximal regions. We use margine map separately in our feature functions. Figure 4.5 shows all these features for a sample document image.

There is also another modification that we consider when computing vertical and horizontal features. The raw vertical features can become a problem since sometimes long vertical run-lengths penetrate in part of text regions. To minimize those penetrations, we automatically change those values to zero whenever the horizontal run-lengths are less than twice the value of the width map at the same pixel location. Width map locally represents the width of connected components and horizontal run-length represent the distance between components. Changing the values to zero ensures that vertical run-lengths are unable to penetrate in text regions, when the distance between connected components is less than twice the width of components.

4.3.5 Gabor features

We use Gabor filters to generate several observation maps. Two sets of Gabor filters are used. One set tries to capture text lines and the other set captures vertical spaces between columns of text. It has been found that 2D Gabor filters are particularly appropriate for texture representation and discrimination. They consist of a Gaussian kernel function modulated by a sinusoidal carrier.

The equation of a real 2D Gabor filter is:

$$g(x, y; \lambda, \theta, \phi, \psi, \sigma, \gamma) = \frac{\gamma}{2\pi\sigma^2} exp\left(-\frac{x'^2 + \gamma^2 y'^2}{2\sigma^2}\right) cos\left(2\pi\frac{x'}{\lambda} + \psi\right)$$

where

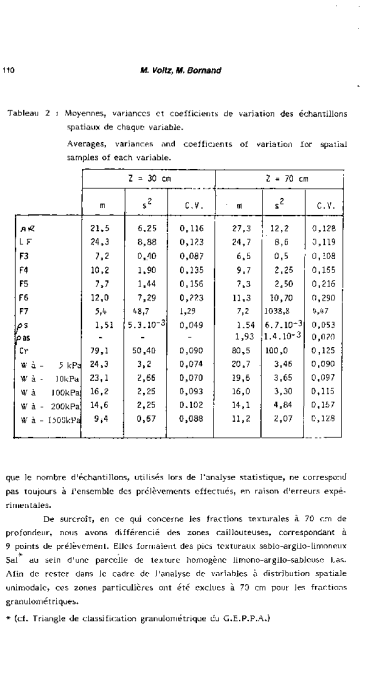
$$x' = xcos\theta + ysin\theta$$

and

$$y' = -xsin\theta + ycos\theta$$

In the above equations, λ represents the wavelength of the carrier, θ represents the orientation of the sinusoidal strips, ψ is the phase offset, σ controls the width of the Gaussian envelope and γ specifies the ellipticity of the Gabor function.

We need to consider two issues here. The first issue is that we only want one sinusoidal peak in g. Thus, we have to adjust σ in a way that it only allows one peak to appear and attenuates all other peaks of the sinusoidal function. This ensures that our kernel only captures one text line. We do not attempt to capture several text lines with one kernel, and the reason is that the height of text lines, and the space between them are not equal. In fact, it is easy to



110 M. Voltz, M. Bornand

Tableau 2 : Moyennes, variances et coefficients de variation des échantillons spatiaux de chaque variable.

Averages, variances and coefficients of variation for spatial samples of each variable.

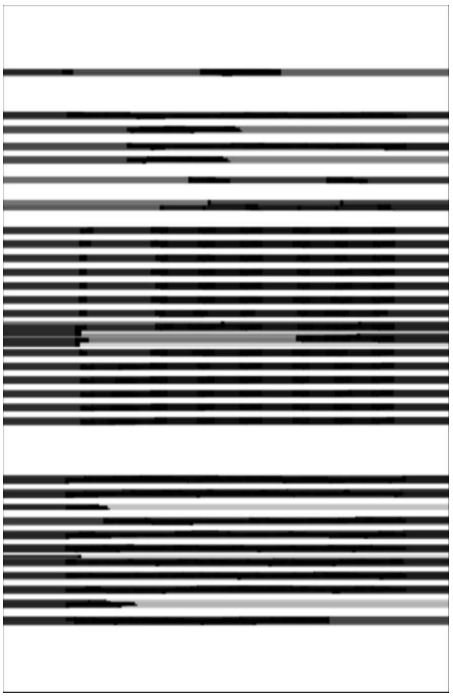
	Z = 30 cm			Z = 70 cm		
	m	s^2	C.V.	m	s^2	C.V.
A*K	21,5	6,25	0,116	27,3	12,2	0,128
L F	24,3	8,88	0,123	24,7	8,6	0,119
F3	7,2	0,40	0,087	6,5	0,5	0,108
F4	10,2	1,90	0,135	9,7	2,25	0,155
F5	7,7	1,44	0,156	7,3	2,50	0,216
F6	12,0	7,29	0,223	11,3	10,70	0,290
F7	54	48,7	1,29	7,2	1038,8	5,47
ρs	1,51	$5.3.10^{-3}$	0,049	1.54	$6.7.10^{-3}$	0,053
ρas	-	-	-	1,93	$1.4.10^{-3}$	0,020
Cr	79,1	50,40	0,090	80,5	100,0	0,125
W à - 5 kPa	24,3	3,2	0,074	20,7	3,46	0,090
W à - 10kPa	23,1	2,66	0,070	19,6	3,65	0,097
W à 100kPa	16,2	2,25	0,093	16,0	3,30	0,115
W à - 200kPa	14,6	2,25	0.102	14,1	4,84	0,157
W à - 1500kPa	9,4	0,67	0,088	11,2	2,07	0,128

que le nombre d'échantillons, utilisés lors de l'analyse statistique, ne correspond pas toujours à l'ensemble des prélèvements effectués, en raison d'erreurs expérimentales.

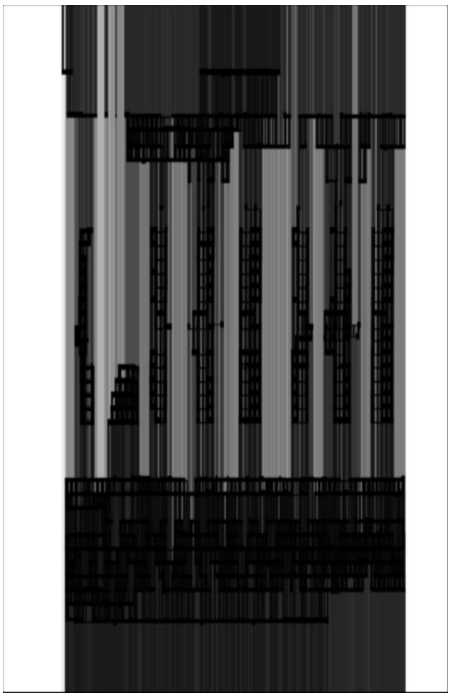
De surcroît, en ce qui concerne les fractions texturales à 70 cm de profondeur, nous avons différencié des zones caillouteuses, correspondant à 9 points de prélèvement. Elles formaient des pics texturaux sablo-argilo-limoneux Sal* au sein d'une parcelle de texture homogène limono-argilo-sableuse Las. Afin de rester dans le cadre de l'analyse de variables à distribution spatiale unimodale, ces zones particulières ont été exclues à 70 cm pour les fractions granulométriques.

* (cf. Triangle de classification granulométrique du G.E.P.P.A.)

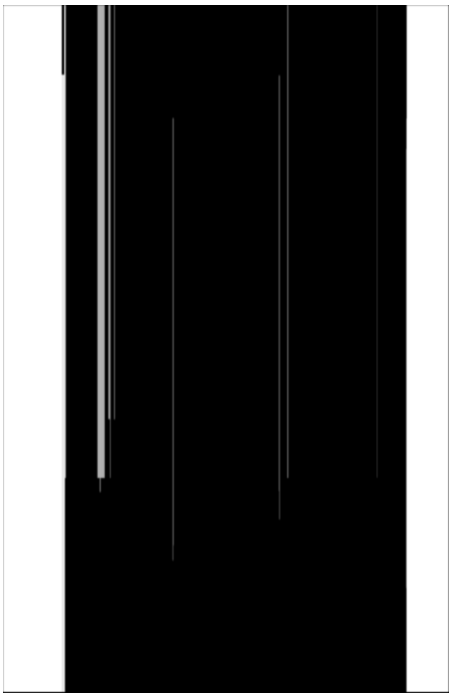
(a) Document image



(b) Horizontal run-length



(c) Vertical run-length



(d) Page margins

Figure 4.5: A document image and its computed vertical, horizontal and marginal run-length maps features. Note that maps are computed using only text components and table structure is ignored.

determine the average height of text lines from text connected components, but much harder to determine the spaces between lines. Furthermore, line-spacing varies from one document to the other. To ensure that our kernel only captures one text line, the value of σ should be tied to λ. The size of the kernel should depend on the σ parameter. The size should normally be selected large enough so that kernel coefficients of the border rows and columns contribute very little to the sum of coefficients. As a rule of thumb the size (in pixels) of the kernel window should be six times the value of σ or larger to ensure that the power of coefficient values at the borders of the kernel window subside to 1% or lower than the power of center coefficients. The second issue is that we want the λ parameter to be dependent on the size of the text line. Our parameters are:

$$\begin{aligned}
\lambda &= 2 \times \text{Avg. text height or width} \\
\sigma &= \frac{\lambda}{3.5} \\
\gamma &= 0.7 \\
\psi &= \begin{cases} \pi & \text{To capture text lines} \\ 0 & \text{To capture white space} \end{cases}
\end{aligned}$$

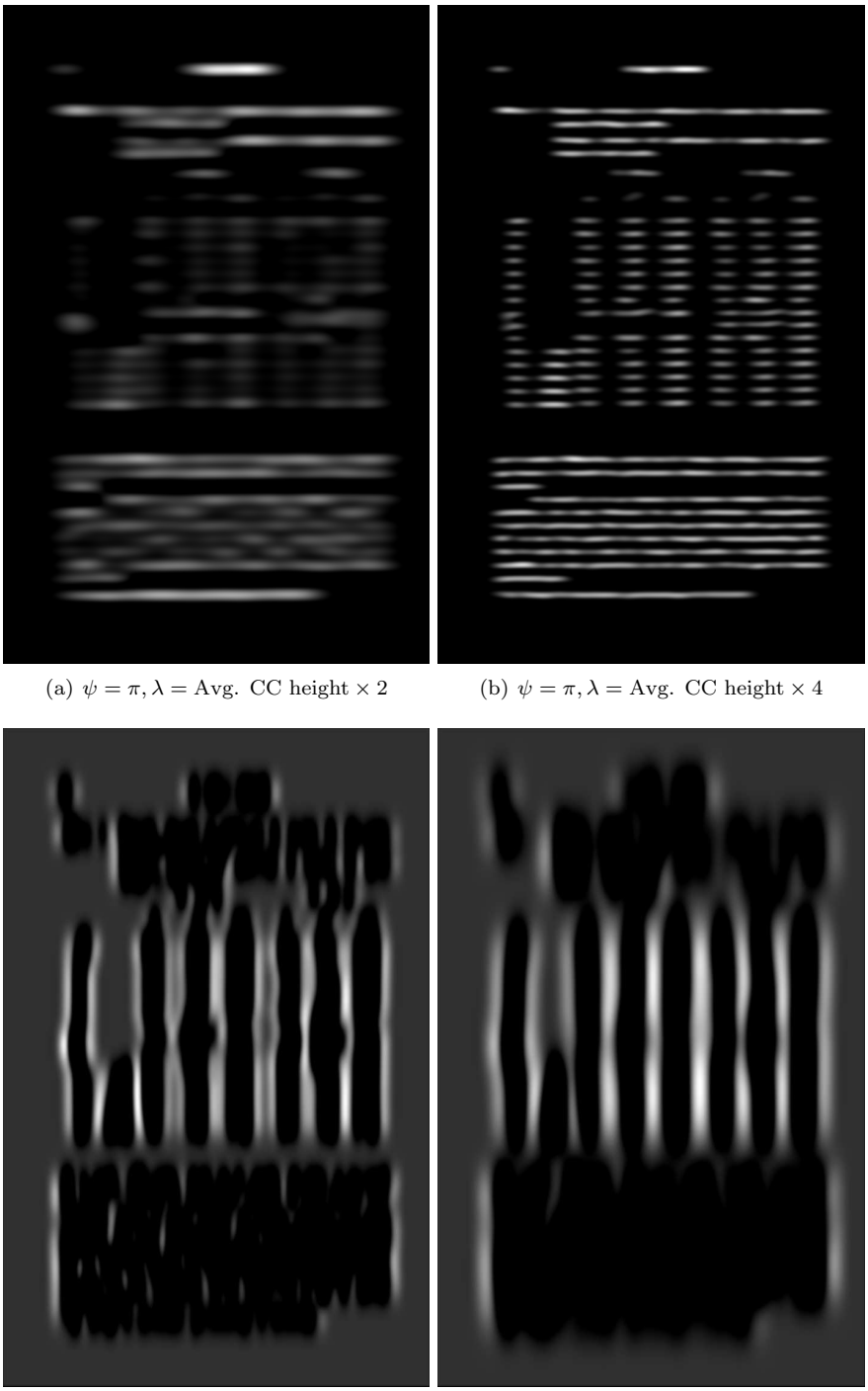
Unfortunately, there is no efficient implementation of Gabor filter in which the λ parameter varies locally. We could manually crop patches from a document image and use a Gabor kernel for each patch that matches text heights locally, but it would take a huge amount of time. Therefore, Gabor filtering using kernel multiplication in frequency space and fixed parameters for each kernel per document is still the only available option. As a consequence two different Gabor kernels are used to capture text lines. For these kernels, the λ parameters are set to $2 \times$ Average Text Height and $4 \times$ Average Text Height. Moreover, to capture white space gaps, three Gabor kernels are used. The λ parameters are set to $1.5 \times$ Text Width, $3.5 \times$ Average Text Width and $5.5 \times$ Average Text Width. Figure 4.6 shows the results of applying the mentioned Gabor filters to a document in figure 4.5.

Figure 4.7 displays two additional examples of filtered images, highlighting the effect of using various λ parameters to capture text lines of different font sizes.

4.4 Feature functions

Each feature function has two parts. The first part depends on the label of the site or a combination of the labels from its neighbors. The second part depends on the observations. Theoretically, these observations can be generated from anywhere on the image; however, we restrict them to be computed from and around the site in question.

We described many features that we extract from document images. In order to generate observations from these features, we compute mean and variance for each feature map at the same site. These statistics serve as observations at each



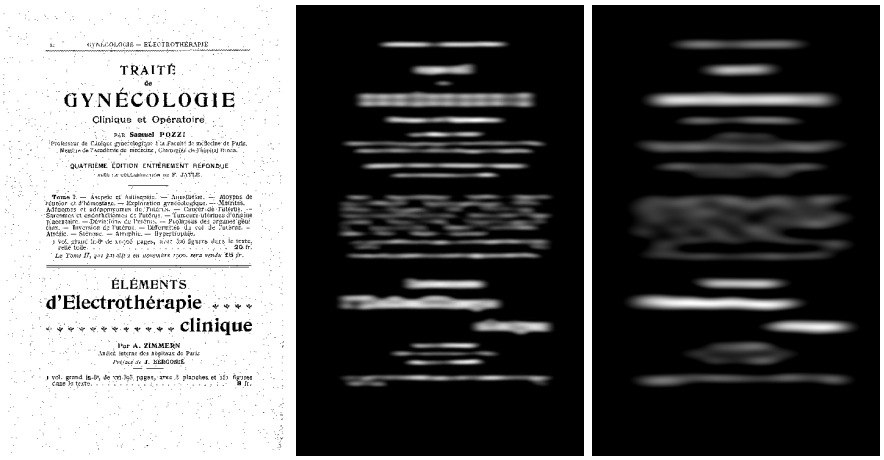
(a) $\psi = \pi, \lambda =$ Avg. CC height $\times$ 2

(b) $\psi = \pi, \lambda =$ Avg. CC height $\times$ 4

(c) $\psi = 0, \lambda =$ Avg. CC width $\times$ 3.5

(d) $\psi = 0, \lambda =$ Avg.n CC width $\times$ 5.5

Figure 4.6: Results of applying two sets of Gabor filters to a document image. **(a)** and **b** display the results of the first set with two Gabor filters that try to capture text lines with different heights. **c** and **d** show the results of two additional Gabor filters that capture gaps between text columns.



(a) Document image (b) Small Gabor kernel (c) Large Gabor kernel

Figure 4.7: Results of applying two Gabor filters with different kernel sizes to a document image. This clearly shows the ability of Gabor filters to capture text lines of different font sizes. The result in **b** belongs to a Gabor filter with medium kernel size. As the size of the Gabor filter gets larger in **c**, it reveals larger text lines on the page.

site. To reduce the effect of scale, mean and variance are not computed using the same height and width as the site but by using a patch centered on the site but with a size proportional to the local height of text components (which comes from the height map) at that site.

Some global feature functions are used that do not depend on any observation. These functions are noted below. In these function y_c refers to the label of the current site at the center and y_t, y_l and y_{tl} refer to labels on the top, left and top-left of the site, respectively. Labels may be 1 for text and 0 for non-textual sites. Note that these functions are separate independent feature functions that each takes its own weight while training. Thus, they cannot be merged into a single function.

$$
\begin{aligned}
f &= [y_c = y_l] \\
f &= [y_c = y_t] \\
f &= [y_c = 0] \times [y_l = 0] \\
f &= [y_c = 0] \times [y_l = 1] \\
f &= [y_c = 1] \times [y_l = 0] \\
f &= [y_c = 1] \times [y_l = 1] \\
f &= [y_c = 0] \times [y_t = 0] \\
f &= [y_c = 0] \times [y_t = 1] \\
f &= [y_c = 1] \times [y_t = 0] \\
f &= [y_c = 1] \times [y_t = 1]
\end{aligned}
$$

$$
\begin{aligned}
f &= [y_c = 0] \times [y_{tl} = 0] \\
f &= [y_c = 0] \times [y_{tl} = 1] \\
f &= [y_c = 1] \times [y_{tl} = 0] \\
f &= [y_c = 1] \times [y_{tl} = 1]
\end{aligned}
$$

So far, the global feature functions are described. The weights of these feature functions control the smoothness and continuity of the labels. For example if the weight for the first feature is positive and large enough compared to other weights, it encourages the current site to have the same label as the site on its left. On the other hand a negative weight for this feature discourages a site to have the same label as its neighbor site on its left.

Apart from global feature functions, there are many number of feature functions that are tied to observations. These feature functions are:

$$
\begin{aligned}
f &= [y_c = 0] \times (x_{tl} + 1) \\
f &= [y_c = 0] \times (x_t + 1) \\
f &= [y_c = 0] \times (x_{tr} + 1) \\
f &= [y_c = 0] \times (x_l + 1) \\
f &= [y_c = 0] \times (x_c + 1) \\
f &= [y_c = 0] \times (x_r + 1) \\
f &= [y_c = 0] \times (x_{bl} + 1) \\
f &= [y_c = 0] \times (x_b + 1) \\
f &= [y_c = 0] \times (x_{br} + 1) \\
f &= [y_c = 0] \times (x_{tl} - 1) \\
f &= [y_c = 0] \times (x_t - 1) \\
f &= [y_c = 0] \times (x_{tr} - 1) \\
f &= [y_c = 0] \times (x_l - 1) \\
f &= [y_c = 0] \times (x_c - 1) \\
f &= [y_c = 0] \times (x_r - 1) \\
f &= [y_c = 0] \times (x_{bl} - 1) \\
f &= [y_c = 0] \times (x_b - 1) \\
f &= [y_c = 0] \times (x_{br} - 1) \\
f &= [y_c = 0] \times [y_t = 0] \times (x_c + x_t) \\
f &= [y_c = 0] \times [y_t = 1] \times (x_c + x_t) \\
f &= [y_c = 1] \times [y_t = 0] \times (x_c + x_t) \\
f &= [y_c = 1] \times [y_t = 1] \times (x_c + x_t) \\
f &= [y_c = 0] \times [y_l = 0] \times (x_c + x_l) \\
f &= [y_c = 0] \times [y_l = 1] \times (x_c + x_l) \\
f &= [y_c = 1] \times [y_l = 0] \times (x_c + x_l)
\end{aligned}
$$

$$f = [y_c = 1] \times [y_l = 1] \times (x_c + x_l)$$

The mentioned feature function prototypes are designed empirically with extensive testing. Again, we note that due to a limitation in our inference method, the label of a site only depends on the labels of sites on its left or top; however, observations do not have this restriction. The observation x in these functions can be substitute by the normalized mean and variance of any extracted feature on the previous sections.

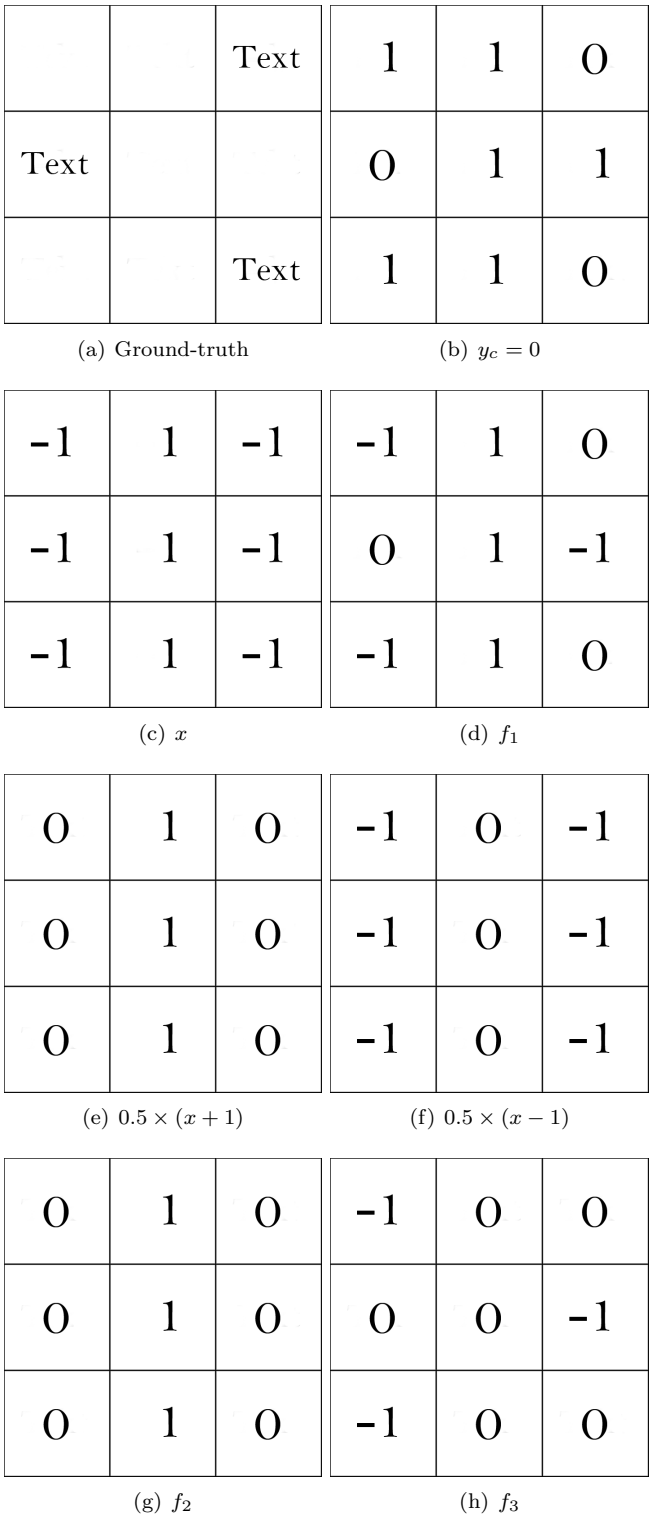
There is one issue that deserves an explanation. In these feature functions, we add +1 and -1 to each observation. Note that observations are normalized between 1 and -1. The question is: what are the advantages of using two feature functions for an observation instead of one? To explain this we turn into a toy example. First, we define f_1 which is a feature function that detects non-text areas with a normalized observation. Equivalently, we have f_2 and f_3. We argue that using f_2 and f_3 instead of f_1 is more effective for the overall feature space.

$$\begin{aligned} f_1 &= [y_c = 0] \times x \\ f_2 &= [y_c = 0] \times 0.5 \times (x + 1) \\ f_3 &= [y_c = 0] \times 0.5 \times (x - 1) \end{aligned}$$

In figure 4.8, a ground-truth image with 3×3 sites is shown. This ground-truth shows that this particular toy example has three textual sites and six non-textual sites. In addition, the extracted observation x is shown. Having said that, all the feature functions f_1,f_2 and f_3 only interact with non-textual areas. In other words, their values for text sites are zero. The sum of the values of f_1 for non-textual sites is zero. It means that the values of x for non-textual sites cancel each other and a training algorithm such as voted percepton assigns a zero weight to this feature function. However, the sum of the values of f_2 and f_3 are 3 and -3, respectively. It means that f_2 positively contributes to some sites for having a "non-text" label and f_3 with a negative sign contributes to some other sites for having a "non-text" label. Finally, the training algorithm assigns a positive weight to f_2 and a negative weight to f_3 and both feature functions would contribute positively to the overall classification.

4.5 Label decoding

Label decoding is one of the crucial steps in training a conditional random field's model and detecting text regions. For now, we assume that we have a trained model in which every parameter in our model is set according to our training dataset. Given a document image, we divide our image into sites and calculate observation vectors for each site. Label decoding refers to the process of assigning "text" or "non-text" labels to each size in such a way that the conditional



(a) Ground-truth (b) $y_c = 0$

(c) x (d) f_1

(e) $0.5 \times (x + 1)$ (f) $0.5 \times (x - 1)$

(g) f_2 (h) f_3

Figure 4.8: A toy example that shows why a normalized observation is better to appear in two feature functions instead of just one.

probability $p(y|x)$ is maximized according to the trained CRF model.

Finding the optimal label configuration in a linear-chain conditional random field or graphical models with tree-like structure is a straightforward process using the Viterbi algorithm. However, in two-dimensional CRF, no exact method exists for label decoding. Nearly all methods that perform decoding in two-dimensional CRFs are approximation methods that iteratively perform inference and change label configuration until they reach to a predefined number of iterations or until they converge. Examples of such methods are *Monte Carlo methods* [3], *Loopy Belief Propagation* [64], variational methods [43]. While the success of these methods is proven in some application like Turbo decoding [67], in the domain of computer vision [31], the precise conditions under which these methods will converge are still not well understood. Furthermore, since we work on high-resolution document images with typically 300 dpi, methods that demand high number of iterations to converge, are impractical.

Another simple to implement solution is *Iterated Conditional Models* (ICM). ICM is an iterative method proposed by Besag in 1986 [9]. Instead of maximizing the probability as a whole, the method tries to maximize the conditional probability of each site by considering only its neighbors. At each iteration, the algorithm chooses sites from left to right and top to bottom and estimates the probability for all possible label configurations for the site in question. Then it picks the label that maximizes the local probability. Finding the global maximum is not guaranteed, but the method converges to a local maximum. Moreover, if instead of using all the neighbors of a site, we restrict the label of the site to be dependent merely on the sites that we have already visited in one iteration, then it takes just one iteration for the ICM to converge. Therefore, ICM is the method that we choose for label decoding.

4.6 Training (parameter/weight estimation)

In this section, we discuss how to estimate the parameters λ_j of a conditional random field. In general, conditional random fields may be trained with latent variables or for structure learning. However, we are provided with fully labeled data, which is the simplest case for training.

Maximum likelihood is the foundation for training CRFs. Weights are chosen such that the training data has the highest probability under the model. The conditional log-likelihood of a set of training sites (s_s, y_s) using λ as parameters is given by:

$$\ell_\lambda = \sum_{s \in S} \left(\sum_{k=1}^{F} \lambda_k f_k(y_{i,i \in N(s)}, y_s, x_{i,i \in N(s)}, x_s) - \log Z(x_s, \lambda) \right).$$

where S is the total number of sites in training dataset and F is the total number of feature functions. Differentiating the log-likelihood function with respect to parameter λ_k is given by:

$$
\begin{aligned}
\frac{\partial \ell_\lambda}{\partial \lambda_k} &= \sum_{s \in S} \left(f_k(y_s, x_s) - \frac{\sum_{y \in Y} f_k(y_s, x_s) \exp \sum_{i=1}^{F} \lambda_i f_i(y_s, x_s)}{Z(x_s, \lambda)} \right) \\
&= \sum_{s \in S} \left(f_k(y_s, x_s) - \sum_{y \in Y} f_k(y_s, x_s) P(y|x_s) \right) \\
&= \sum_{s \in S} \left(f_k(y_s, x_s) - E_{P(y|x_s)}[f_k(y_s, x_s)] \right)
\end{aligned}
$$

where $Y = \{\text{text}, \text{non-text}\}$ and $E_{p()}[.]$ is the expected value of the model under the conditional probability distribution. For maximum likelihood solution, the equation will equal zero, and therefore the expectation of the feature f_k with respect to the model distribution must be equal to the expected value of f_k with respect to the empirical distribution. However, calculating the expectation requires the enumeration of all the y labels. In linear-chain models, inference techniques based on a variation of forward-backward algorithms can be performed to efficiently compute this expectation. However, in two-dimensional CRFs, approximation techniques are needed to simplify the computations. One solution is to use a *Voted Perceptron Method*.

4.6.1 Collin's voted perceptron method

Perceptrons [81] use an approximation of the gradient of the unregularized conditional log-likelihood. Perceptron-based training methods consider one misclassified instance at a time, along with its contribution to the gradient. The expectation of features are further approximated by a point estimate of the feature function vector at the best possible labeling. The approximation for the i^{th} instance and the k^{th} feature function can be written as:

$$
\nabla_k \ell(\lambda) \approx \left(f_k(y^i, x^i) - f_k(\hat{y}^i, x^k) \right)
$$

where

$$
\hat{y}^i = \arg\max_y \lambda_k f_k(y, x^i)
$$

Using this approximate gradient, the following first order update rule can be used for maximization:

$$
\lambda_k^{t+1} = \lambda_k^t + \alpha \left(f_k(y^i, x^i) - f_k(\hat{y}^i, x^i) \right).
$$

where α is the learning rate. This update step is applied once for each misclassified instance x^i in the training set and multiple passes are made over the training dataset. Thought, it has been noted that the final obtained weights suffer from over-fitting. As a solution, Collins [26] suggests a voting scheme, where, in a particular pass of the training data, all the updates are collected,

and their unweighed average is used as an update to the set of weights in each iteration. So the whole update-rule from iteration t to iteration $t+1$ becomes:

$$\lambda_k^{t+1} = \lambda_k^t + \alpha \frac{1}{|S|} \sum_{s \in S} \left(f_k(y_{N(s)}, y_s, x_s, x_{N(s)}) - f_k(\hat{y}_{N(s)}, \hat{y}_s, x_s, x_{N(s)}) \right).$$

where y_s and $y_{N(s)}$ are true labels from ground-truth and $\hat{y}_s$ and $\hat{y}_{N(s)}$ are labels, computed from the model using label decoding at each iteration. The Collin's voted perceptron is shown to achieve lower errors in fewer numbers of iterations.

4.6.2 Loopy belief propagation

Another approach to take instead of Collin's voted perceptron is to use L-BFGS optimization method which is quite popular for training conditional random fields because it takes few number of iterations to converge. But before we are able to utilize it, we have to compute an approximation of marginal probabilities for our model. One way of performing marginal inference on graphical models is a message passing algorithm called *Belief propagation*, also known as *Sum-product algorithm*. this algorithm was first proposed by Judea Pearl in 1982 [76] for trees. This algorithm computes exact marginals and terminate after 2 steps. In the first step, messages are passed inwards: staring at the leaves, each node passes a message along the (unique) edge towards the root node. It is possible to obtain messages from all other adjoining nodes until the root has obtained messages from all of its adjoining nodes.

The same algorithm is used in general graphs, sometimes called "loopy" belief propagation [31]. The new modified algorithm works by passing messages around the network defined by four-connected sites. Each message is a vector of two dimensions, given by the number of possible labels. Let m_{pq}^t be the message that site p sends to a neighboring site q at time t. Using negative *log* probabilities all entries in m_{pq}^0 are initialized to zero. At each iteration new messages are computed according to the update rule below:

$$m_{pq}^t(y_p) = \min_{y_p} \left(V(y_p, y_q) + D_p(y_p) + \sum_{s \in N(p) \setminus \{q\}} m_{sp}^{t-1}(y_p) \right).$$

where $N(p)\backslash\{q\}$ denotes the neighbors of p other than q. $V(y_p, y_q)$, the negated sum of our edge feature functions, is the cost of assigning labels y_p and y_q to two neighboring sites p and q. $D_p(y_p)$, the negated sum of our node feature functions, is the cost of assigning label y_p to site p. After T iterations a belief vector of two-dimensions is computed for each site.

$$b_q(y_q) = D_q(y_q) + \sum_{p \in N(q)} m_{pq}^T(y_q).$$

Beliefs may be used as the expected value of model parameters to compute gradients for each feature function. It is known that on graphs containing a single loop it will always converge, but the probabilities obtained might be

incorrect [100]. Several sufficient (but not necessary) conditions for convergence of loopy belief propagation to a unique fixed point exist [6]. Even so, the precise conditions under which loopy belief propagation will converge are still not well understood in its application for computer vision. As a consequence, through our experiments we found that the usage of some feature functions will result to no answer from L-BFGS optimization.

4.6.3 L-BFGS

Perhaps the simplest approach to optimize $\ell(\lambda)$ is by using a steepest ascent algorithm, but in practice, this requires too many iterations. Newton's method converges much faster because it takes into account the second derivatives of the likelihood function, but it needs to compute *Hessian*, the matrix of all second derivatives. The size of the Hessian matrix is quadratic in the number of parameters and since real-world applications often use thousands (e.g. computer vision) or even millions (e.g. natural language processing) of parameters, even storing the full Hessian matrix in memory is impractical.

Instead, the latest optimization methods make approximate use of second-order information. *Quasi-Newton* methods such as *BFGS*, which compute an approximation of Hessian from only the first derivative of the objective function, have been successfully applied [55]. A full $K \times K$ approximation to the Hessian still requires quadratic size. Therefore, a limited-memory version of the method (L-BFGS) is developed [20].

L-BFGS can simply be treated as a black-box optimization procedure, requiring only that one provides the value and first-derivative of the function to be optimized. The important note is that the use of L-BFGS does not make it easier to compute the partition function. Recall that the partition function $Z(x, \lambda)$ and the marginal distributions $p(y_s|x_s, \lambda)$ in the gradients, depend on λ; thus, every training instance in each iteration has a different partition function and marginal. Even in linear-chain CRFs that the partition function Z can be computed efficiently by forward-backward algorithms, it is reported that on a part-of-speech (POS) tagging data set, with 45 labels and one million words of training data, training requires over a week [94].

Our implementation of L-BFGS is based on libLBFGS library [72] in C++.

4.7 Experimental results

Our training dataset consists of 28 pages from a collection of our corpus. It contains a large variety of page layouts including articles, journals, forms and multi-columns documents with or without side notes and graphical components.

For each page in our dataset we generated a ground-truth image and a map that indicates the location of between-columns in multi-column documents. The former is used only for tracking the rate of errors in those areas and is not part of a training routine. Figure 4.9 shows two pages in our training dataset along with their corresponding ground-truth. Note that three colors are used as part

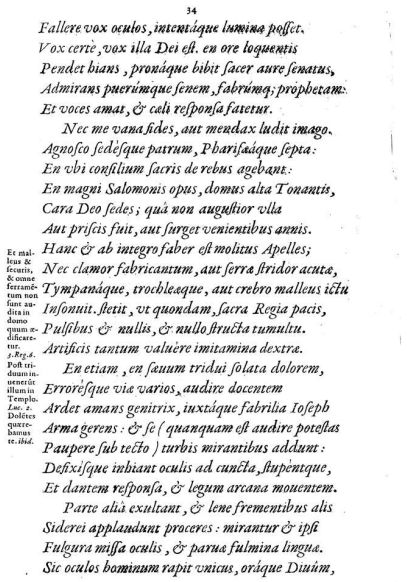
of each ground-truth image. Black and white areas represent text and background regions respectively. Gray areas however represents areas that could be either text or background. In other words we ignore those areas and do not penalize training algorithms for making a misclassification in those regions.

In our first attempts to train our 2D CRF model, we used Collin's Voted Perceptron. The voted perceptron algorithm does not converge to a global minimum. Generally, depending on the learning rate, it starts with a large classification error, and traverses some local minima of the likelihood function until it goes through a cycle. By analyzing the classification results for each iteration, it became clear that in most of the iterations, the algorithm struggles with one row or one column of pixels around the borders of text regions, which occupy most of the training time without gaining any noticeable results for the human eye. Our idea to resolve this situation is to ignore sites that are located on the borders of text regions in our ground-truth images. We label the sites that are not completely located on text or non-text regions as "don't care" regions.

The use of "don't care" regions has two advantages. The first advantage is that computed errors indicate information for the sites that are visually important for us and more reliably reflect the quality of the segmentation. The second advantage is that the Voted perceptron method performs more smoothly and well behaved during the training. Whenever we process a "don't care" site, we simply do not count the error for that site, and we prevent that site to have any influence on the approximate gradient. The drawback is that we still decode their labels and use those labels in the feature functions of the active surrounding sites that demand it. Figure 4.10 illustrates the number of misclassified sites in 400 iterations of training by Collin's Voted Perceptron with or without considering "don't care" sites. The error rates with or without considering "don't care" sites are shown in **red** and **blue**, respectively.

In each iteration of training, a large vector of weights are stored. The dilemma is which weight vector to choose. Figure 4.10 clearly shows that as the number of misclassified text sites starts to decrease, there is an increase in the classification error for sites that are located between text columns. In other words, those sites, located between text columns, should not be labeled with text, but iteration after iteration text labels start to be build-up in those regions. Considering both numbers, it seems reasonable to use the obtained weights around iteration 147 as the trained parameters for our model.

Some obtained results are shown in figure 4.11. We have particularly chosen examples that have more errors. These results are showing the output of CRF without any post-processing. Though, in many cases there are holes inside text regions that can be easily corrected in post-processing. In some cases penetration of non-text labels happen (sub-figure e) at the boundaries of text regions. These penetrations do pose a problem if they divide a single text region into two separate regions. It happens because some feature functions, responsible for capturing gap between side notes, have obtained a large weight. Other type of problem that often occurs (sub-figure g) is touching of a small region between side notes and the main text. There is always a trade-off between misclassification in text regions and gaps between side-notes.



34

Fallere vox oculos, interúaque lumina posset.
Vox certè, vox illa Dei est. en ore loquentis
Pendet hians, pronáque bibit sacer aure senatus,
Admirans puerúmque senem, fabrúmq; prophetam:
Et voces amat, & cæli responsa fatetur.
Nec me vana fides, aut mendax ludit imago.
Agnosco sedésque patrum, Pharisæáque septa:
En vbi consilium sacris de rebus agebant:
En magni Salomonis opus, domus alta Tonantis,
Cara Deo sedes; quá non augustior vlla
Aut prisci fuit, aut surget venientibus annis.
Et malleus & fecuris, & omne ferramẽtum non sunt audita in domo quum ædificaretur. 3. Reg. 6.
Hanc & ab integro faber est molitus Apelles;
Nec clamor fabricantum, aut serræ stridor acutæ,
Tympanáque, trochleaque, aut crebro malleus ictu
Insonuit. stetit, vt quondam, sacra Regia pacis,
Pulsibus & nullis, & nullo structa tumultu.
Artificis tantum valuêre imitamina dextræ.
Post triduum inuenerũt illum in Templo. Luc. 2. Dolẽtes quærebamus te. ibid.
En etiam, en sæuum tridui solata dolorem,
Errorésque viæ varios, audire docentem
Ardet amans genitrix, iuxtáque fabrilia Ioseph
Arma gerens: & se (quanquam est audire potestas
Paupere sub tecto) turbis mirantibus addunt:
Defixíque inhiant oculis ad cuncta, stupentque,
Et dantem responsa, & legum arcana mouentem.
Parte aliâ exultant, & lene frementibus alis
Siderei applaudunt proceres: mirantur & ipsi
Fulgura missa oculis, & paruæ fulmina linguæ.
Sic oculos hominum rapit vnicus, orsque Diuúm,

676 *AFFECTIONS DU NERF OPTIQUE*

Etiologie. — La malformation crânienne paraît la conséquence d'une soudure précoce des sutures en rapport avec une affection durémérienne (Virchow). La névrite optique serait sous la dépendance de l'inflammation méningée (Virchow, Hirschberg), dont la cause première n'a pas encore été établie.

Il n'est pas rare de rencontrer des cas d'oxycéphalie sans aucune lésion des nerfs optiques.

Fig. 386. — Crâne en tour avec atrophie post-névritique des nerfs optiques et cécité.

Diagnostic. — L'examen radiographique fournira des données importantes en montrant notamment l'amincissement irrégulier de la calotte crânienne.

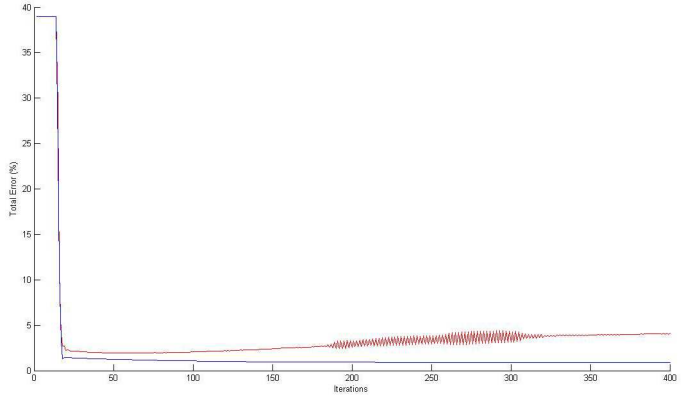
Traitement. — Si l'on est consulté au moment du développement des lésions, on instituera un traitement spécifique. A la période d'atrophie des nerfs, toute thérapeutique sera inutile. On conseillera aux parents de donner à leur enfant un enseignement spécial.

Atrophie héréditaire du nerf optique

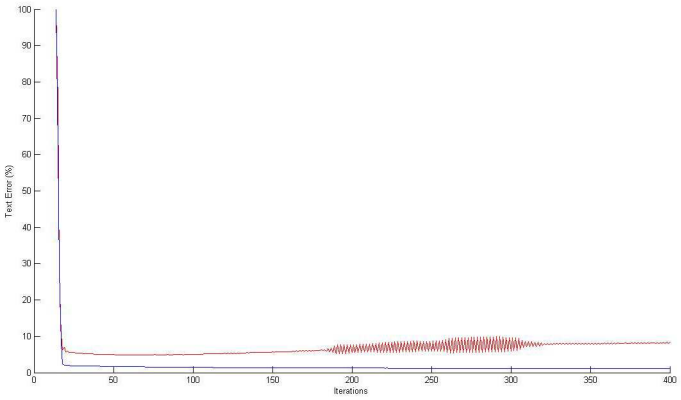
L'affection du nerf optique, différenciée par Leber et décrite sous le nom d'*atrophie héréditaire*, évolue suivant le type de la névrite rétrobulbaire et apparaît le plus souvent à l'époque de la puberté. Elle peut atteindre plusieurs membres des générations successives d'une même famille.

Symptômes. — L'affection se manifeste par un trouble visuel à début brusque ou progressif, essentiellement caractérisé par une altération marquée de l'acuité visuelle centrale sans modification de la vision périphérique. Il y a parfois des céphalées à ce moment seulement. L'acuité visuelle s'abaisse à 5/40 ou à 3/50 et l'épreuve stéréoscopique montre un scotome central pour les cou-

Figure 4.9: Two sample pages from our training dataset along with their ground-truth images.

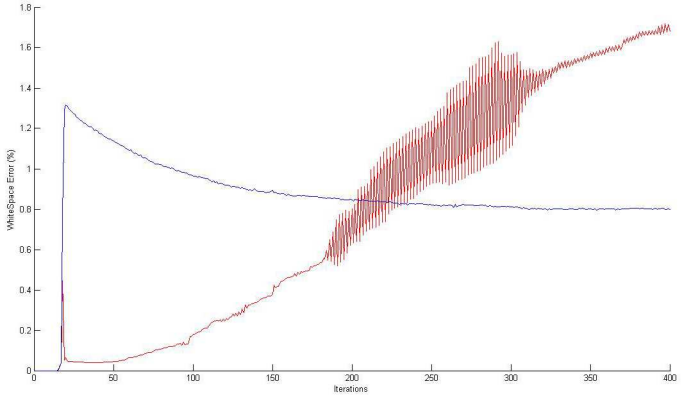


(a) Number of misclassified sites for all sites (%)

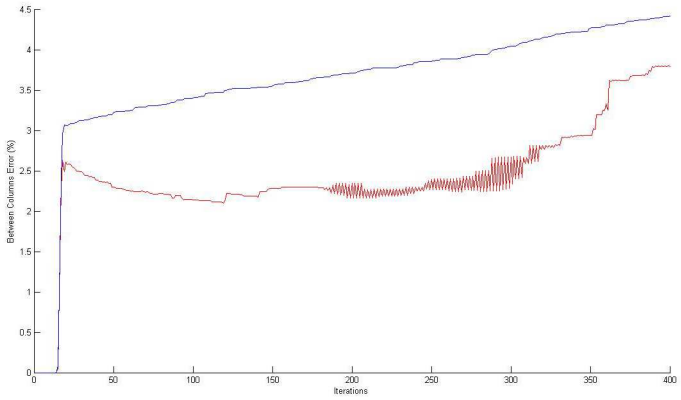


(b) Number of misclassified text sites (%)

Figure 4.10: Number of misclassified sites per iteration in Voted Perceptron training. The error rates with or without considering "don't care" sites are shown in **red** and **blue**, respectively.



(a) Number of misclassified non-textual sites (%)



(b) Number of misclassified sites between text columns (%)

Figure 4.10: Number of misclassified sites per iteration in Voted Perceptron training. The error rates with or without considering "don't care" sites are shown in **red** and **blue**, respectively.

4.7.1 Post-processing

Three post-processing steps are applied to the output of the CRF:

- Removing regions whose width or height are smaller than the width of height of the average character.
- Opening each regions separately from other regions. Since side notes are very close to main text body, care should be taken not to merge two text regions together.
- Applying a hole-filling method to fill holes inside each text region separately.

Four pages are shown in figure 4.12 after applying these three post-processing steps.

4.8 Results and discussion

At this point, regions of text are detected and ready for text line detection, but paragraphs are yet to be found. Paragraphs may or may not be separated depending on the distance between them. However since in most ground-truth data for competitions such as *ICDAR2011 Historical Document Layout Competition*, paragraphs are annotated separately, evaluation of the results based on region matching with the true ground-truth data are meaningless for the purpose of comparison.

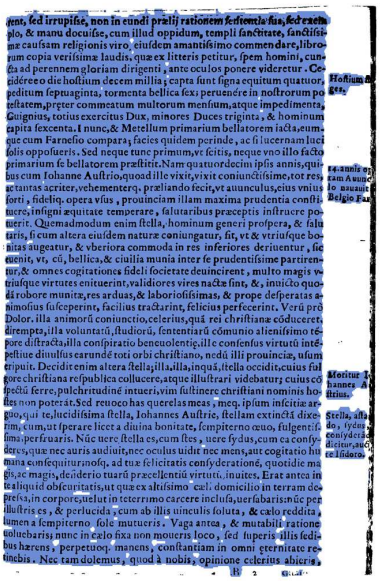
We report the current success rate for site-wise classification. The statistics aim to show how far the results are from the closest acceptable region segmentation for the means of text line detection. This means that instead of preparing a ground-truth that separates all the paragraphs, we generate the ground-truth data by correcting the segmentation results to make them acceptable for text line detection.

Table 4.1 indicates number of misclassified sites from the output of our CRF model.

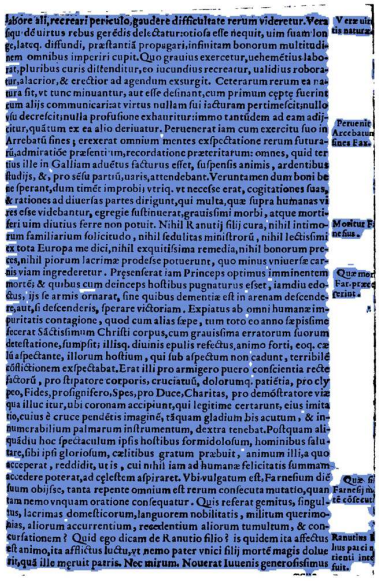
Table 4.1: NUMBER OF MISCLASSIFIED SITES (%) FROM THE OUTPUT OF OUR CRF MODEL

Total sites (%)	Textual sites (%)	Non-textual sites (%)	Gap between columns (%)
0.97	1.32	0.88	3.5

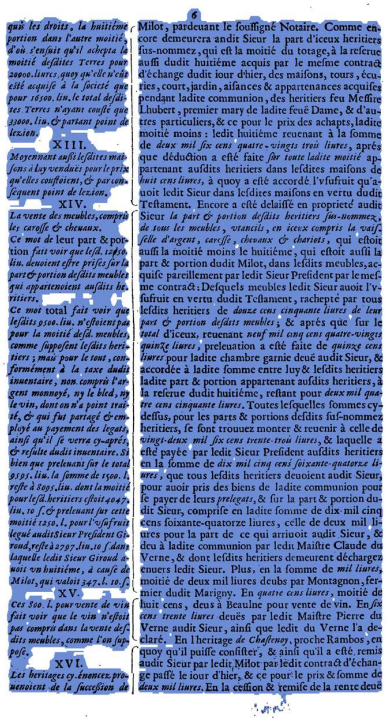
Two other tables 4.2 and 4.3 show region segmentation success rates for different images. The indicated rates are computed between the segmentation output and the closest acceptable segmentation for text line detection. The closest acceptable segmentation is a segmentation that is capable of producing



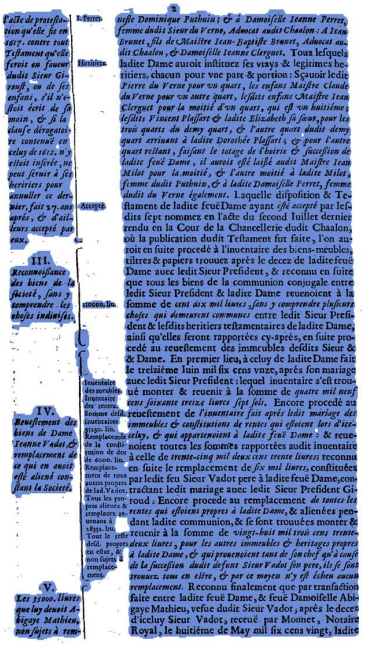
(c) 37531011459550 00011



(d) 37531011459550 00023

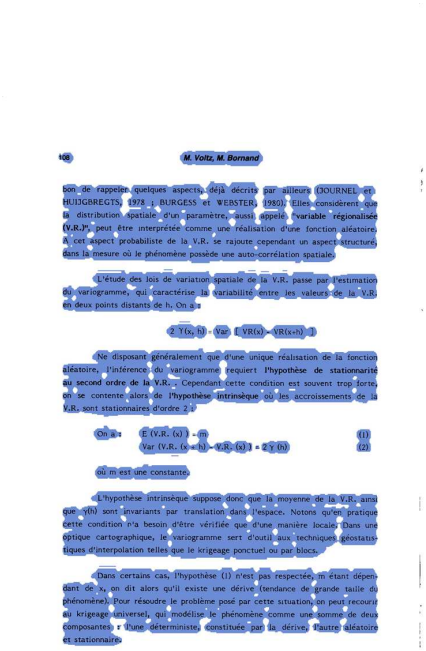


(e) 37531022215512 00006

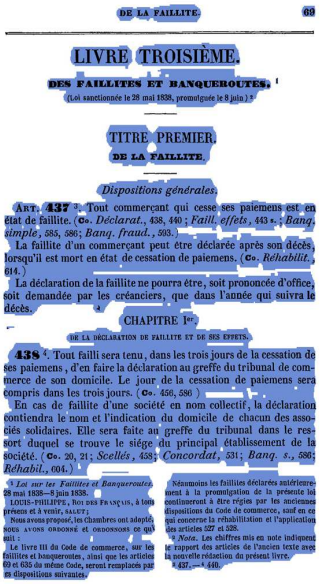


(f) 37531022215512 00002

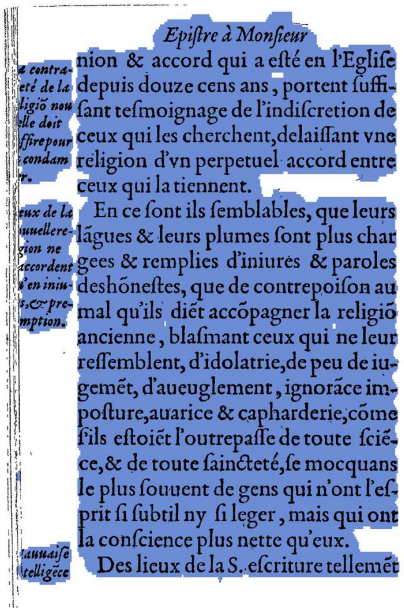
Figure 4.11: Several obtained results for text region detection without post-processing.



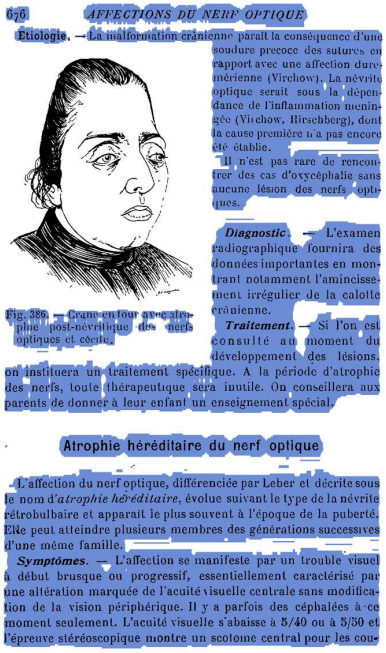
(a) 9782738000064 0110



(b) 37502043571332 00074

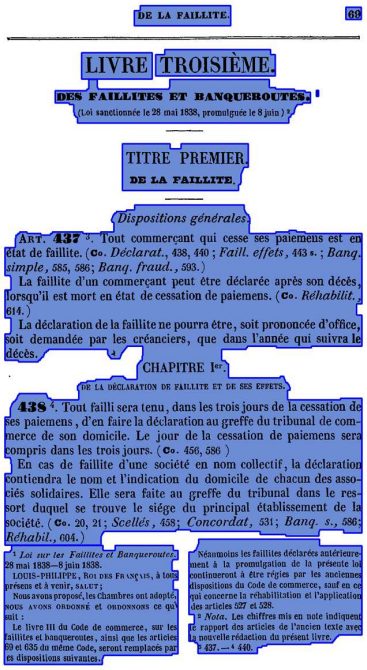


(c) 37531011465979 00019



(d) 37531032730088 00703

Figure 4.11: Several obtained results for text region detection without post-processing.



DE LA FAILLITE. 68

LIVRE TROISIÈME.

DES FAILLITES ET BANQUEROUTES.

(Loi sanctionnée le 28 mai 1838, promulguée le 8 juin.)

TITRE PREMIER.

DE LA FAILLITE.

Dispositions générales.

ART. 437. Tout commerçant qui cesse ses paiemens est en état de faillite. (Co. *Déclarat.*, 438, 440; *Faill. effets*, 443 s.; *Banq. simple*, 585, 586; *Banq. fraud.*, 593.)

La faillite d'un commerçant peut être déclarée après son décès, lorsqu'il est mort en état de cessation de paiemens. (Co. *Réhabilit.* 614.)

La déclaration de la faillite ne pourra être, soit prononcée d'office, soit demandée par les créanciers, que dans l'année qui suivra le décès.

CHAPITRE Ier.

DE LA DÉCLARATION DE FAILLITE ET DE SES EFFETS.

438. Tout failli sera tenu, dans les trois jours de la cessation de ses paiemens, d'en faire la déclaration au greffe du tribunal de commerce de son domicile. Le jour de la cessation de paiemens sera compris dans les trois jours. (Co. 456, 586.)

En cas de faillite d'une société en nom collectif, la déclaration contiendra le nom et l'indication du domicile de chacun des associés solidaires. Elle sera faite au greffe du tribunal dans le ressort duquel se trouve le siége du principal établissement de la société. (Co. 20, 21; *Scellés*, 458; *Concordat*, 531; *Banq. s.*, 586; *Réhabil.*, 604.)

[1] *Loi sur les Faillites et Banqueroutes*, 28 mai 1838—8 juin 1838.

LOUIS-PHILIPPE, ROI DES FRANÇAIS, à tous présens et à venir, SALUT;

Nous avons proposé, les chambres ont adopté, NOUS AVONS ORDONNÉ et ORDONNONS ce qui suit :

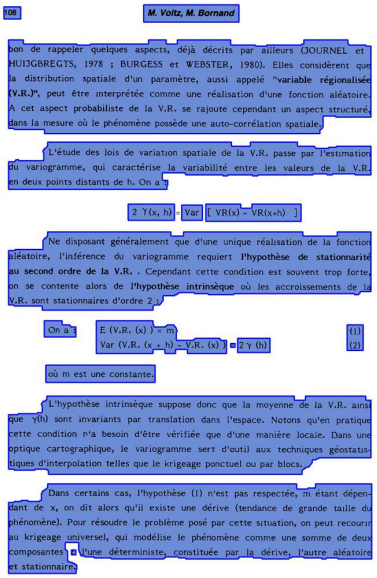
Le livre III du Code de commerce, sur les faillites et banqueroutes, ainsi que les articles 69 et 635 du même Code, seront remplacés par les dispositions suivantes.

Néanmoins les faillites déclarées antérieurement à la promulgation de la présente loi continueront à être régies par les anciennes dispositions du Code de commerce, sauf en ce qui concerne la réhabilitation et l'application des articles 527 et 528.

[2] *Nota.* Les chiffres mis en note indiquent le rapport des articles de l'ancien texte avec la nouvelle rédaction du présent livre.

[3] 437. — [4] 440.

(e) 37502043571332 00074



108 M. Voltz, M. Bornand

bon de rappeler quelques aspects, déjà décrits par ailleurs (JOURNEL et HUIJGBREGTS, 1978 ; BURGESS et WEBSTER, 1980). Elles considèrent que la distribution spatiale d'un paramètre, aussi appelé "variable régionalisée (V.R.)", peut être interprétée comme une réalisation d'une fonction aléatoire. A cet aspect probabiliste de la V.R. se rajoute cependant un aspect structural, dans la mesure où le phénomène possède une auto-corrélation spatiale.

L'étude des lois de variation spatiale de la V.R. passe par l'estimation du variogramme, qui caractérise la variabilité entre les valeurs de la V.R. en deux points distants de h. On a :

$2\,\gamma(x, h) = \mathrm{Var}\,[\,VR(x) - VR(x+h)\,]$

Ne disposant généralement que d'une unique réalisation de la fonction aléatoire, l'inférence du variogramme requiert l'hypothèse de stationnarité au second ordre de la V.R. . Cependant cette condition est souvent trop forte, on se contente alors de l'hypothèse intrinsèque où les accroissements de la V.R. sont stationnaires d'ordre 2 :

On a : $E\,(V.R.\,(x)) = m$ (1)

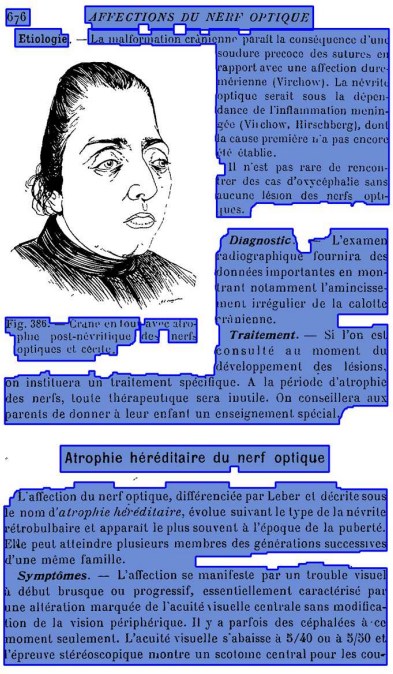
$\mathrm{Var}\,(V.R.\,(x+h) - V.R.\,(x)) = 2\,\gamma\,(h)$ (2)

où m est une constante.

L'hypothèse intrinsèque suppose donc que la moyenne de la V.R. ainsi que $\gamma(h)$ sont invariants par translation dans l'espace. Notons qu'en pratique cette condition n'a besoin d'être vérifiée que d'une manière locale. Dans une optique cartographique, le variogramme sert d'outil aux techniques géostatistiques d'interpolation telles que le krigeage ponctuel ou par blocs.

Dans certains cas, l'hypothèse (1) n'est pas respectée, m étant dépendant de x, on dit alors qu'il existe une dérive (tendance de grande taille du phénomène). Pour résoudre le problème posé par cette situation, on peut recourir au krigeage universel, qui modélise le phénomène comme une somme de deux composantes : l'une déterministe, constituée par la dérive, l'autre aléatoire et stationnaire.

(f) 9782738000064 0110



676 AFFECTIONS DU NERF OPTIQUE

Etiologie. — La malformation crânienne paraît la conséquence d'une soudure precoce des sutures en rapport avec une affection durérienne (Virchow). La névrite optique serait sous la dépendance de l'inflammation méningée (Virchow, Hirschberg), dont la cause première n'a pas encore été établie.

Il n'est pas rare de rencontrer des cas d'oxycéphalie sans aucune lésion des nerfs optiques.

Fig. 388. — Crane en tour avec atrophie post-névritique des nerfs optiques et cécité.

Diagnostic. — L'examen radiographique fournira des données importantes en montrant notamment l'amincissement irrégulier de la calotte crânienne.

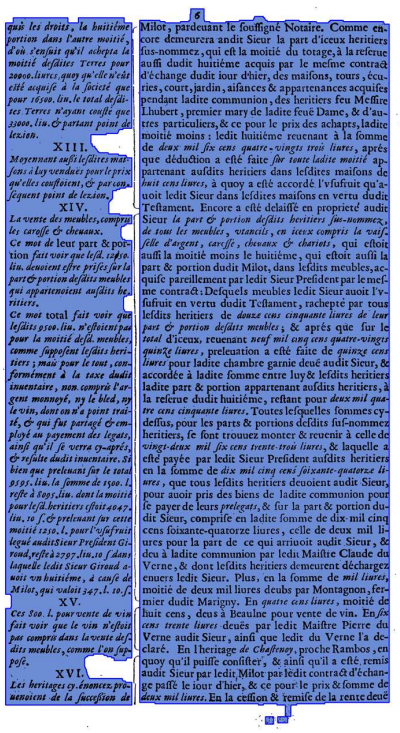
Traitement. — Si l'on est consulté au moment du développement des lésions, on instituera un traitement spécifique. A la période d'atrophie des nerfs, toute thérapeutique sera inutile. On conseillera aux parents de donner à leur enfant un enseignement spécial.

Atrophie héréditaire du nerf optique

L'affection du nerf optique, différenciée par Leber et décrite sous le nom d'*atrophie héréditaire*, évolue suivant le type de la névrite rétrobulbaire et apparaît le plus souvent à l'époque de la puberté. Elle peut atteindre plusieurs membres des générations successives d'une même famille.

Symptômes. — L'affection se manifeste par un trouble visuel à début brusque ou progressif, essentiellement caractérisé par une altération marquée de l'acuité visuelle centrale sans modification de la vision périphérique. Il y a parfois des céphalées à ce moment seulement. L'acuité visuelle s'abaisse à 5/40 ou à 5/50 et l'épreuve stéréoscopique montre un scotome central pour les cou-

(g) 37531032730088 00703



6

Milot, pardeuant le soussigné Notaire. Comme encore demeurera audit Sieur la part d'iceux heritiers sus-nommez, qui est la moitié du totage, à la reserue aussi dudit huitiéme acquis par le mesme contract d'échange dudit iour d'hier, des maisons, tours, écuries, court, jardin, aisances & appartenances acquises pendant ladite communion, des heritiers feu Messire Lhubert, premier mary de ladite feuë Dame, & d'autres particuliers, & ce pour le prix des achapts, ladite moitié moins; ledit huitiéme reuenant à la femme de *deux mil six cens quatre-vingts trois liures*, après que déduction a esté faite *sur toute ladite moitié* appartenant audits heritiers dans lesdites maisons de *huit cens liures*, à quoy a esté accordé l'vsufruit qu'auoit ledit Sieur dans lesdites maisons en vertu dudit Testament. Encore a été delaissé en propriété audit Sieur *la part & portion desdits heritiers sus-nommez de tous les meubles, vtancils, en iceux compris la vaisselle d'argent, carosse, cheuaux & charriots*, qui estoit aussi la moitié moins le huitiéme, qui estoit aussi la part & portion dudit Milot, dans lesdits meubles, acquise pareillement par ledit Sieur President par le mesme contract. Desquels meubles ledit Sieur auoit l'vsufruit en vertu dudit Testament, rachepté par tous lesdits heritiers de *douze cens cinquante liures de leur part & portion desdits meubles*; & après que sur le *total* d'iceux, reuenant *neuf mil cinq cens quatre-vingts quinze liures*, preleuation a esté faite de *quinze cens liures* pour ladite chambre garnie deuë audit Sieur, & accordée à ladite femme entre luy & lesdits heritiers ladite part & portion appartenant audits heritiers, à la reserue dudit huitiéme, restant pour *deux mil quatre cens cinquante liures*. Toutes lesquelles sommes cy-dessus, pour les parts & portions desdits sus-nommez heritiers, se font trouuez monter & reuenir à celle de *vingt-deux mil six cens trente trois liures*, & laquelle a esté payée par ledit Sieur President audits heritiers en la femme de *dix mil cinq cens soixante-quatorze liures*, que tous lesdits heritiers deuoient audit Sieur, pour auoir pris des biens de ladite communion pour le payer de leurs *prelegats*, & sur la part & portion dudit Sieur, compriſe en ladite somme de dix mil cinq cens soixante-quatorze liures, celle de deux mil liures pour la part de ce qui arriuoit audit Sieur, & deu à ladite communion par ledit Maiſtre Claude du Verne, & dont lesdits heritiers demeurent déchargez enuers ledit Sieur. Plus, en la somme de *mil liures*, moitié de deux mil liures deubs par Montagnon, fermier dudit Marigny. En *quatre cens liures*, moitié de huit cens, deus à Beaulne pour vente de vin. En *six cens trente liures* deuës par ledit Maiſtre Pierre du Verne audit Sieur, ainſi que ledit du Verne l'a declaré. En l'heritage *de Chaſteney*, proche Rambos, en quoy qu'il puiſſe conſiſter, & ainſi qu'il a eſté remis audit Sieur par ledit Milot par ledit contract d'échange paſſé le iour d'hier, & ce pour le prix & ſomme de *deux mil liures*. En la ceſſion & remiſe de la rente deuë

que les droits, la huitiéme partion dans l'autre moitié, d'où s'enſuit qu'il achepta la moitié deſdites Terres pour 20000. liures, quoy qu'elle n'eût eſté acquiſe à la ſocieté que pour 18500. liu. le total deſdites Terres n'ayant couſté que 37000. liu. & partant point de lezion.

XIII.

Moyennant auſſi leſdites maiſons à luy vendues pour le prix qu'elles conſtoient, & partant point de lezion.

XIV.

La vente des meubles, compris les caroſſe & cheuaux.

Ce mot de leur part & portion *fait voir que leſd. 18500. liu. deuoient eſtre priſes ſur la part & portion deſdits meubles qui appartenoient auſdits heritiers.*

Ce mot total *fait voir que leſdits 9500. liu. n'eſtoient pas pour la moitié deſd. meubles, comme ſuppoſent leſdits heritiers; mais pour le tout, conformément à la vaxe dudit inuentaire, non compris l'argent monnoyé, ny le bled, ny le vin, dont on n'a point traité, & qui fut partagé & employé au payement des legats, ainſi qu'il ſe verra cy-après, & reſulte dudit inuentaire. Et bien que preleuant ſur le total 9595. liu. la ſomme de 1500. l. reſte à 8095. liu. dont la moitié pour leſd. heritiers eſtoit 4047. liu. 10 ſ. & preleuant ſur cette moitié 1210. l. pour l'uſufruit lequel auditSieur Preſident Giraud, reſte à 2797. liu. 10 ſ. dans laquelle ledit Sieur Giroud auoit vn huitiéme, à cauſe de Milot, qui valoit 349. l. 10. ſ.*

XV.

Ces 800 l. pour vente de vin fait voir que le vin n'eſtoit pas compris dans la vente deſdits meubles, comme l'on ſuppoſe.

XVI.

Les heritages cy énoncez prouenoient de la ſucceſſion de

(h) 37531022215512 00006

Figure 4.12: Results of text region detection after post-processing.

prefect results for text line detection. Accuracy rates are reported as received by Prima Layout Evaluation software [25]. Table 4.2 reports the region segmentation success rates based on weighted area. The other table 4.3 reports the success rates for the same images based on weighted count. The weighted area is more reliable because in methods that simply report counts, a tiny region counts the same as a large region that covers most of the page. The weights are set according to ICDAR2011 historical document layout analysis competition [4]. These weights measure the pure segmentation performance. Miss and partial miss errors are considered worst and have the highest weights. The weights for merge and split errors are set to 50%, whereas false detection, as the least important error type, has a weight of 10%.

Table 4.2: AREA WEIGHTED SUCCESS RATES FOR REGION SEGMENTATION

Image	Merge %	Split %	Miss %	Partial miss %	False detection %	Overall success%
9782738000064 0110	0.6	1.32	0	0.06	0	99.66
9782738000064 0112	0	9.14	0	0.28	0.04	97.89
9782738000064 0267	0	12.19	0	0.14	0	97.02
9782738002082 0145	0	0.04	0	0.06	0	99.98
9782738003041 0337	0	0	0	0	0	100
9782738005151 0042	23.44	1.1	0	0.02	0	92.79
37502016547384 00275	0	19.91	0	2.38	0.08	94.03
37502043571332 00074	0	1.35	0	0	0.03	99.76
37502043571332 00129	0	0	0	0.03	0	99.99
37511000967896 00044	0	0	0	0	0	100
37531011459550 00011	0	0	0	0.02	0.04	99.99
37531011459550 00023	4.53	0	0	0.01	0.02	99.1
37531011465763 00009	7.08	0.06	0	0.46	0	98.41
37531011465979 00019	20.08	0	0	0.09	0	94.24
37531017210387 00004	0	12.33	0	1.73	5.73	95.83
37531021715306 00002	0	29.79	0	0.11	0.02	90.09
37531022215512 00002	14.92	0	0	0.01	0	96.14
37531022215512 00006	18.94	0	0	0.02	0.03	94.69
37531022331848 00165	0	0	0	0	0	100
37531022333950 00254	0	0	0	0	0	100
37531023943955 00010	0	0	9.14	0.01	0	99.97
37531023943955 00043	0.16	0.04	0	0.01	0	99.97
37531027315911 00007	0	0	0	0	0	100
37531032730088 00703	0	1.55	0	0.14	0	99.7
37531032730096 00692	0	2.8	0.14	0.96	0	99.3
37531032886898 00116	0	0	0	0.3	0.01	99.99
CIDELOT 00002 000112	0	4.21	0	0.54	0.01	99.09
SEPTLOT 00001 000547	5.63	0.61	0	0.2	0	98.73

Our experiments prove that the choice of training parameters has a major effect on the segmentation results. There is always a trade-off between the number of misclassified textual sites and the number of misclassified non-textual sites, located between text columns. As described before, some features that are responsible for capturing gaps between side notes and some features capture text lines. If the trained model obtains larger weights for features that capture gaps between side notes, then some errors appear on text areas. Four types of errors can be identified. The first type is about non-textual holes that appear in the middle of text regions. Figure 4.13 (A) illustrates this problem. The source of the problem are due to observations that come from Gabor filtering and are responsible for capturing side notes. This problem occurs when there are not enough text characters in a particular location inside a text region. It can be easily fixed in post-processing by applying a hole-filling algorithm.

Table 4.3: COUNT WEIGHTED SUCCESS RATES FOR REGION SEGMENTATION

Image	Merge %	Split %	Miss %	Partial miss %	False detection %	Overall success%
9782738000064 0110	13.33	23.53	0	13.33	9.09	87.85
9782738000064 0112	0	23.33	0	20.69	16.67	86.36
9782738000064 0267	0	34.43	0	23.08	0	83.84
9782738002082 0145	0	12.5	0	22.22	0	91.31
9782738003041 0337	0	0	0	33.33	0	88.41
9782738005151 0042	28.57	21.05	0	28.57	0	81.64
37502016547384 00275	0	86.02	0	17.14	9.09	56.85
37502043571332 00074	0	11.11	0	0	9.09	95.65
37502043571332 00129	0	0	0	40	0	85
37511000967896 00044	0	0	0	0	0	100
37531011459550 00011	0	0	0	28.57	16.67	87.9
37531011459550 00023	27.27	0	0	20	9.09	86.65
37531011465763 00009	33.33	11.11	0	33.33	0	80.27
37531011465979 00019	42.86	0	0	50	0	70.9
37531017210387 00004	0	55.56	0	66.67	67.74	48.64
37531021715306 00002	0	44.44	0	28.57	9.09	77.65
37531022215512 00002	15.38	0	0	26.67	0	88.96
37531022215512 00006	45.45	0	0	66.67	16.67	62.36
37531022331848 00165	0	0	0	0	0	100
37531022333950 00254	0	0	0	0	0	100
37531023943955 00010	0	0	40	40	0	76
37531023943955 00043	50	33.33	0	50	0	65.35
37531027315911 00007	0	0	0	0	0	100
37531032730088 00703	0	22.22	0	46.15	0	78.81
37531032730096 00692	0	22.58	14.29	40	0	80.44
37531032886898 00116	0	0	0	40	16.67	82.96
CIDELOT 00002 000112	0	47.22	0	42.42	9.09	72.17
SEPTLOT 00001 000547	12.9	6.9	0	12.9	0	93.22

A slightly more serious problem is when the same problem occurs near the border of text regions, which leads to the second type of error, namely non-textual penetrations. Figure 4.13 (B) displays penetration of non-textual sites in text regions. This problem is also fixable in post-processing by isolating the text region and applying morphological opening on the region while respecting the borders. Morphological opening should only be applied after isolating a text region, otherwise two text regions may merge incorrectly.

The third problem is of broken titles. Figure 4.14 shows this problem in part of a document image. The title on this page is divided into two parts when there are enough empty space around. This problem is a serious problem and we do not have a fix for that.

The fourth problem is when sides notes are very close to main text and the main text is slightly slanted to one side. Figure 4.15 illustrates this problem. Its clear from the images that Gabor filters are not able to capture gaps between side notes.

4.8.1 Discussion on parameters

The final matter that should be discussed is the choice of parameters and their effects on the results of the region detector. For this purpose we train a CRF on a selected subset of the dataset containing 16 documents. The goal is to determine the effects of changing parameters such as learning rate, overlapping ratio and the maximum number of cycles of ICM on the training error. For this study, we consider the first 100 iterations of the training algorithm and the

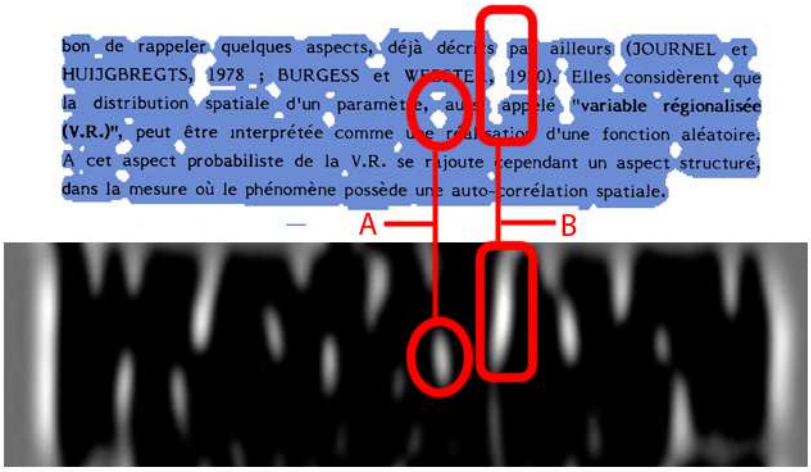


Figure 4.13: This figure displays two types of error that frequently occur in the results of text region detection. They happen when there are gaps between words. If the gap is located in the middle of the text region, it appears as holes A and if it exists near the border of the text region, it causes penetrations B. Both problems can be fixed in the post-processing stage.

Figure 4.14: This figure displays a serious problem when the title of the page divides into two isolated text regions. Currently there is no fix for his problem and the use of morphological opening is not recommended to solve it.

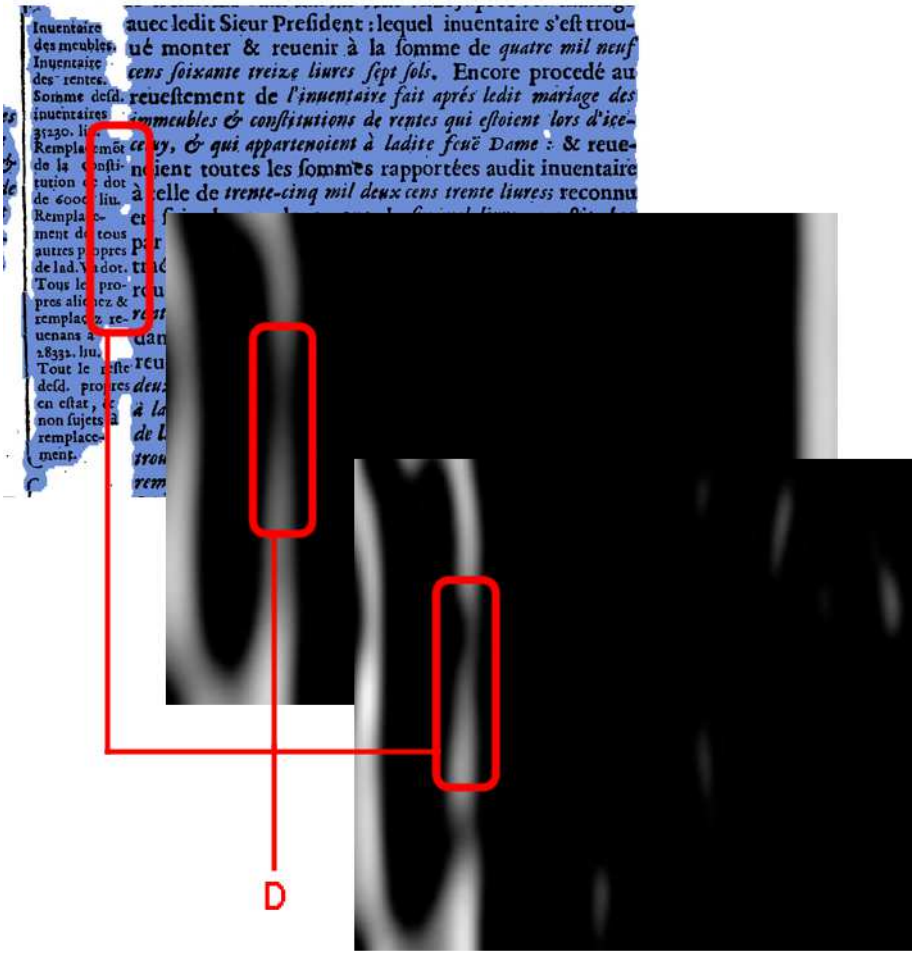


Figure 4.15: This figure illustrates a problem when sides notes are very close to main text and the main text is slightly slanted. In such situation, Gabor filters are not able to capture the small gap between side notes which cause both text region to be merged.

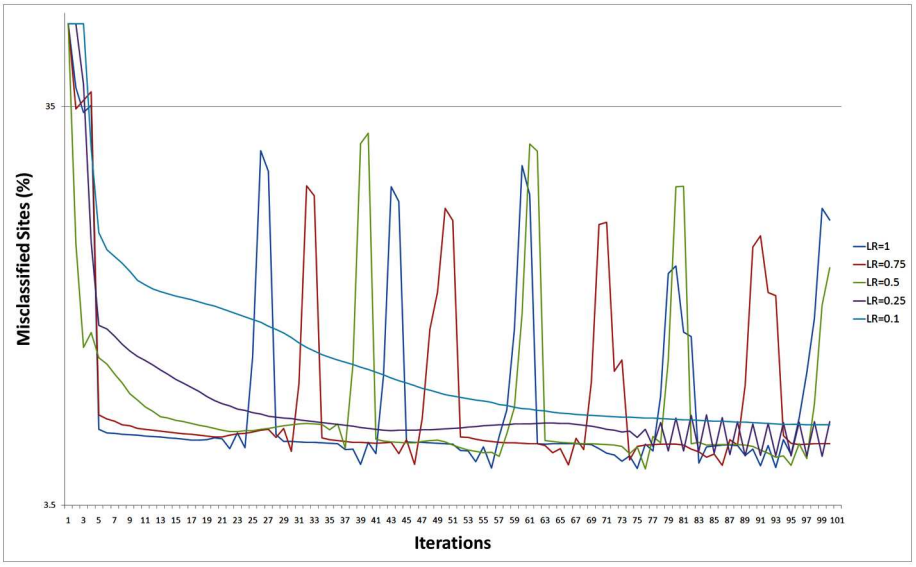


Figure 4.16: This figure shows the evolution of training error rate with 5 different learning rates. All other parameters remain the same.

obtained error in each iteration. Any other parameter of the training remains the same unless if stated otherwise.

Learning rate

Learning rate refers to the learning rate of the voted perceptron training algorithm. It controls the fraction of computed deltas in each iteration of the training that are contributing to the weights of the feature functions. The number of ICM cycles is 10. The block overlap ratio is 0.25 and the width and height of each block is half of the mean width and height of all text characters of the page, respectively.

Figure 4.16 shows the evolution of training error with 5 different learning rates. It suggests that a high value for learning rate forces the training to go into a cycle. On the other hand, the training process is smooth with a small value for learning rate (0.1 in this case) however the process converges to a higher training error. In this particular experiment, a learning rate of 0.25 can be considered as good because the number of misclassified sites decreases reasonably fast without going into cycles.

Unfortunately, there are no rules of thumb to determine a best value for learning rate without considering the number of blocks, number of features functions and their characteristics.

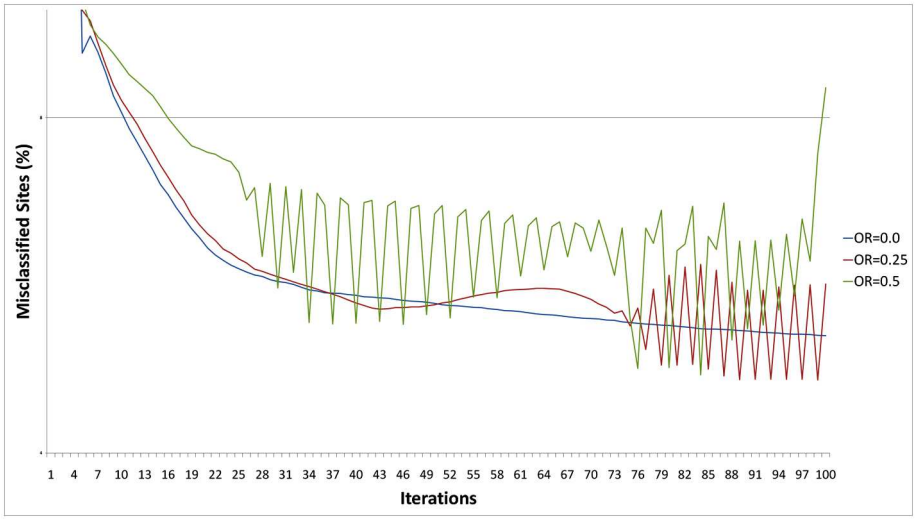


Figure 4.17: This figure shows the evolution of training error rate with 3 different overlapping ratios.

Overlapping ratio

Overlapping ratio refers to the amount that two consecutive blocks overlap. For this experiment we consider three different overlapping ratio; without overlap, with 25% and 50% overlap. Everything else remain the same but the total number of sites.

Figure 4.17 displays the percentage of misclassified sites during the training process with 3 different overlapping ratios. In conclusion, a smaller overlapping ratio results in less number of sites, much faster training process and less number of misclassified sites. However, despite the decrease in the number of misclassified sites, the number of misclassified sites that are located in between columns of text increases. For this reason, it is in our best interest to use a value higher than 0.25 for the overlapping ratio. Values higher than 0.5 result in a huge number of sites and increase the memory consumption and training time substantially.

Maximum number of ICM cycles

Maximum number of ICM cycles refers to the maximum number of cycles that are allowed for the iterated conditional modes inference algorithm before the algorithm converges. The ICM algorithm is supposed to converge to a fixed state of the system after several cycles, however the convergence is not guaranteed. In the former case, a maximum number of cycles is set to terminate the inference algorithm prematurely.

Figure 4.18 shows the progress of the training algorithm with 5 different maximum number of ICM cycles. The results indicate that except of the first experiment with only one cycle, all other experiments perform the same way with slight changes. In conclusion 5 cycles can be considered a good value as a

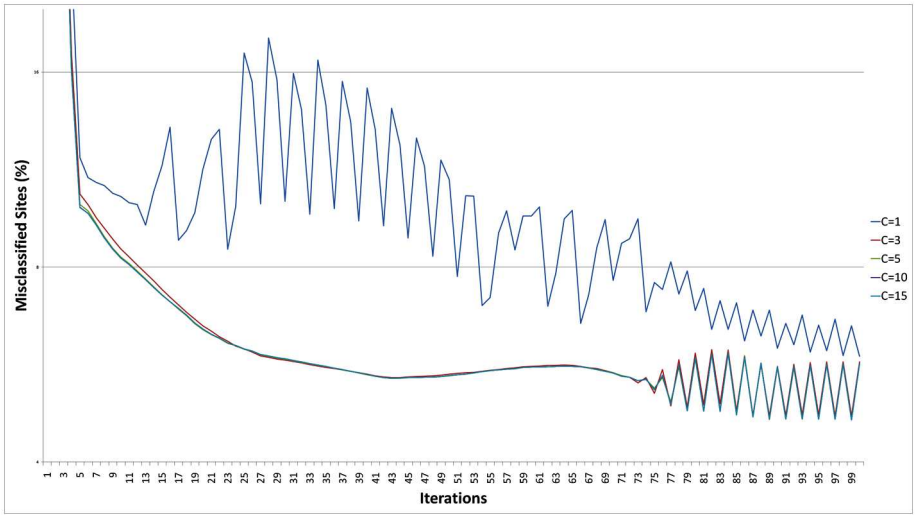


Figure 4.18: This figure displays the number of misclassified sites per iteration for 5 experiments with different maximum number of ICM cycles.

trade-off between speed and accuracy of the training process.

4.9 Final Notes

This chapter provided a method for detecting and separating regions or columns of text. Because we are yet to detect paragraphs within each region, a complete evaluation and comparison of the results will be performed in chapter 6, when other parts of the system are available.

Chapter 5

Text line detection

Text line detection refers to the segmentation of each text region into distinct entities, namely text lines. In chapter 2 we mentioned and analyzed many methods for detecting text line.

Our text line detection method is a variant of the method proposed by Papavassiliou in [75]. The original method segments a document image into non-overlapping vertical zones with equal width. The height of each zone is equal to the height of the document image. Its width is equal to 5% of the width of the document image so as to ignore the effect of skewed text lines, and wide enough to contain decent amount of characters. Also the original method disregards zones situated close to the left and right borders of the page; mainly because they do not contain sufficient amount of text.

Since documents in our corpus contain side notes, it is not wise to dismiss zones that do not contain sufficient amount of text compared to zones in the middle of the document. One reason why the original method neglects these zones is because of the effect of large gaps that affect the overall estimation of model parameters. To solve this problem we ensure that parameters of the model are estimated from detected text regions. Also we ensure that detected lines do not cross from one text region to another as it happens in the original method.

5.1 Initial text line separators

The first step is to calculate the projection profile of each vertical zone onto y axis. Let PR_i be the projection profile of the i^{th} vertical zone onto y axis. Peaks and valleys of PRs give rough indication of the location of text lines; however, in the case where writing style results in large gaps between successive words, a vertical zone many not contain enough foreground pixels for every text line. In order to slake the influence of these instances on PR_i, a smoothed projection profile SPR_i is estimated as a normalized weighted sum of M profiles on either side of the i^{th} zone. The dimension for PR_i and SPR_i is $1 \times$ Page's height. In figures 5.1 and 5.2, the bar chart view for PR_i and SPR_i are rendered at the

same location of i^{th} zone in sub-figure (a) and (b), respectively.

$$SPR_i = \sum_{j=-M}^{M} w_j PR_{i+j}.$$

where the weights are:

$$w_j = \frac{\exp \frac{-3|j|}{M+1}}{\sum_{k=-M}^{M} \exp \frac{-3|k|}{M+1}}$$

Weights are defined to decay exponentially with respect to the distance from the zone in the middle.

Finally, the first derivative of a smoothed projection may be obtained using a symmetric difference equation as follows:

$$\triangle SPR_i(j) = \frac{2}{h(\frac{h}{2}+1)} \sum_{k=1}^{2h} k.\left(SPR_i(j+k) - SPR_i(j-k)\right).$$

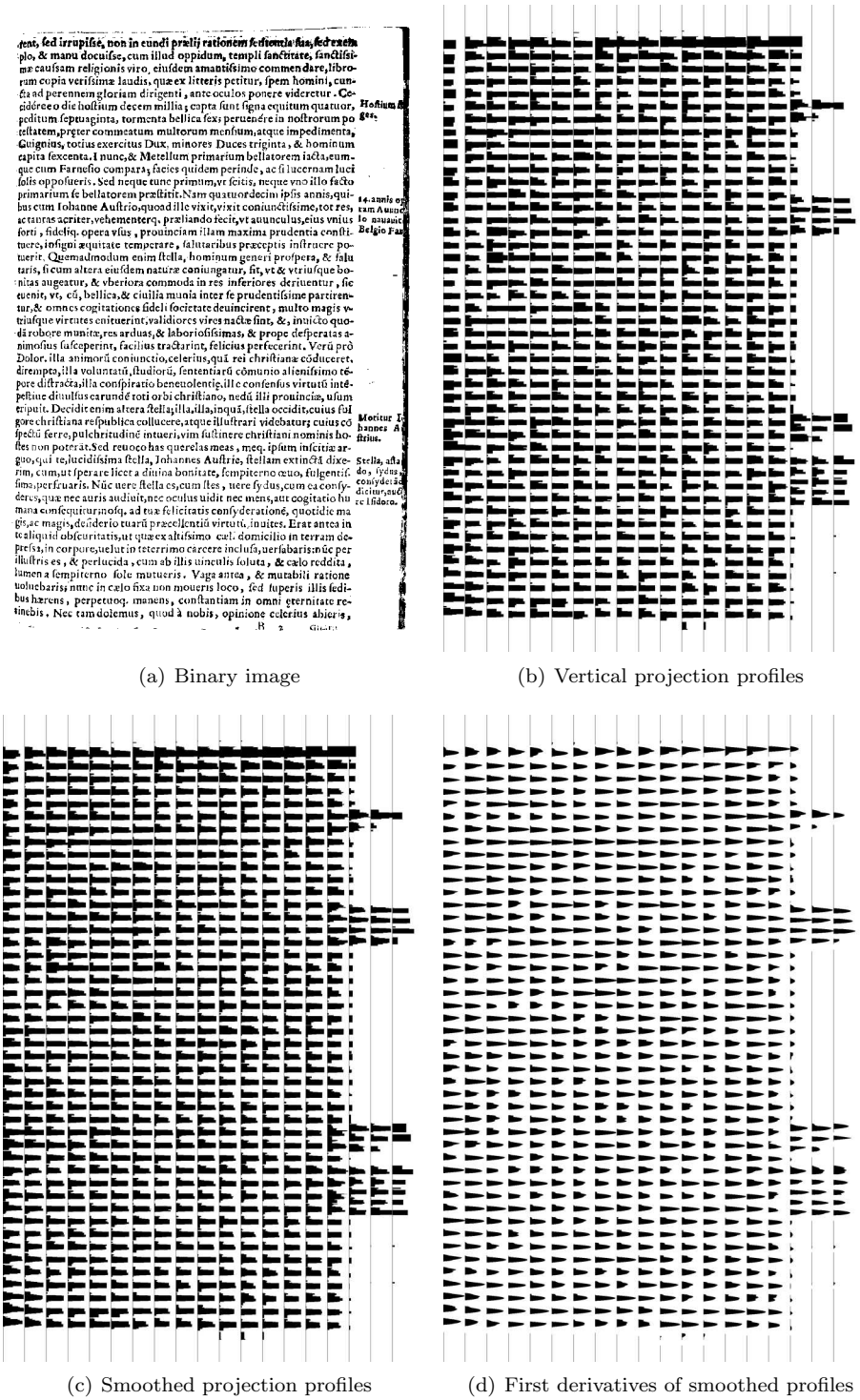
where h is set to the closest odd integer value to the mean height of all CCs. The upper and lower bounds of text lines are the local maxima and local minima of $\triangle SPR_i$ respectively. Likewise gap regions are identified as the areas between consecutive minima and maxima of $\triangle SPR_i$.

Having the first derivatives of the smoothed projection profiles, finding the initial text and gap regions is a matter of applying a threshold to these first derivatives. Figure 5.3 shows the initial text and gap regions for the two image in figures 5.1 and 5.2. Initial separators for separating text lines can be drawn in the middle of each gap in each vertical zone. In this regard, separators are line segments that are defined in the middle of each gap region inside every vertical zone. They are used later to find text lines.

5.2 Refinement of initial text line separators

Initial text line separators contain two types of errors: redundant separators resulting from misclassified text and gaps due to poor local extrema, introduced in the smoothed derivatives and separators that cut through descenders or ascenders of text characters. In order to correct these errors and to locate separators more accurately, a *Hidden Markov Model* (HMM) [78, 91] is formulated with parameters drawn from statistics of the initial text and gap regions. For each document, an HMM is formulated separately on-the-fly and is applied once to each vertical strip of text and gap regions. Viterbi decoding scheme [34] is applied on each zone to obtain a better succession of text and gaps.

We briefly note some formal definitions of HMM. Our HMM is characterized by the following items:



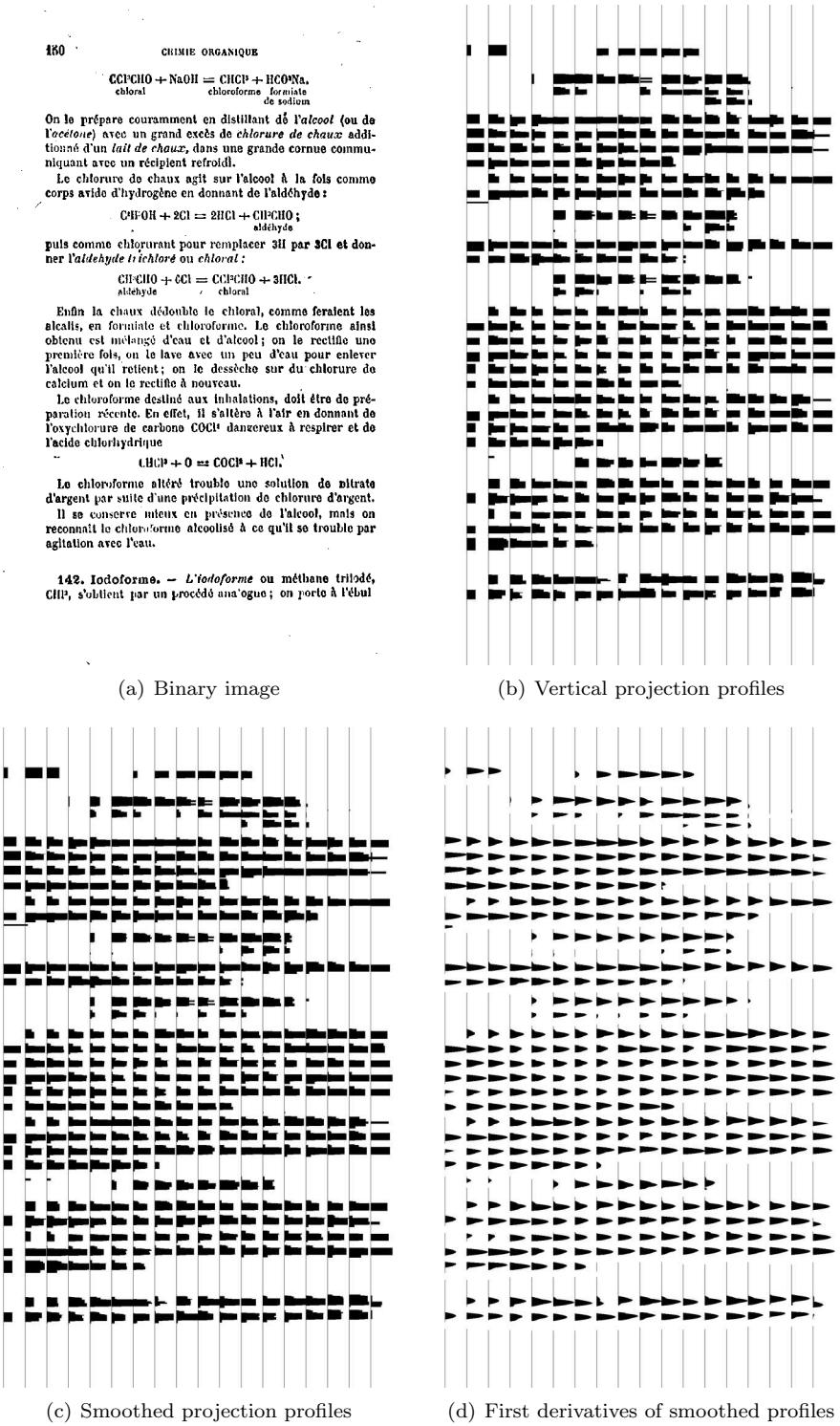
(a) Binary image

(b) Vertical projection profiles

(c) Smoothed projection profiles

(d) First derivatives of smoothed profiles

Figure 5.1: First steps in line detection to obtain initial lines and gaps for a single document image.



(a) Binary image

(b) Vertical projection profiles

(c) Smoothed projection profiles

(d) First derivatives of smoothed profiles

Figure 5.2: First steps in line detection to obtain initial lines and gaps for a single document image.

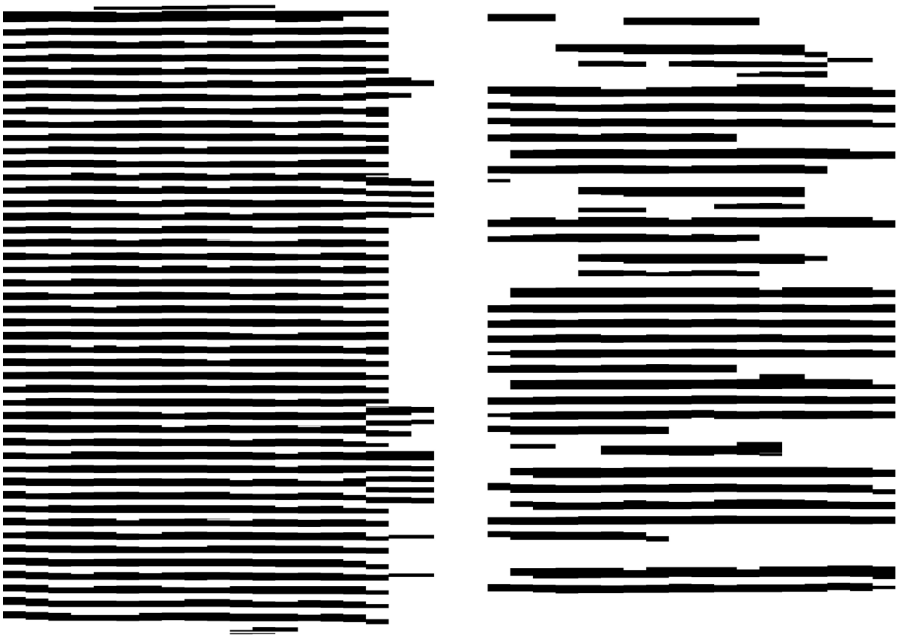


Figure 5.3: Initial text and gap regions for document images in figures 5.1 and 5.2. Text line separators can be drawn in the middle of each gap (white).

- N, the number of states in the model. In the application for text-gap correction, we have two states: text and gap. Typically, the states are interconnected in a way that any state can be reached from any other state. With our two-states model, a text state may remain in text state at each region, or it may transit to a gap state. The same statement is also true for a gap state.

- M, the number of distinct observations per state. Generally, observations are used to train a model and estimate transition and emission probability matrices. In other words, transition and emission probabilities are estimated during the training phase based on training sequences and then during inference the observations are used to find the optimal sequence of states. However, in the method proposed by [75] the HMM model calculates transition and emission probabilities continuously on-the-fly based on the height of each region and the mean of pixels' intensities, covered by the region.

- The state transition probability matrix $A(k)$ for k^{th} region of the vertical strip is modeled by exponential distributions as follows:

$$A(k) = \begin{pmatrix} \exp\left(-\frac{h_k}{m_0}\right) & 1 - \exp\left(-\frac{h_k}{m_0}\right) \\ 1 - \exp\left(-\frac{h_k}{m_1}\right) & \exp\left(-\frac{h_k}{m_1}\right) \end{pmatrix}$$

  where h_k denotes the height of the k^{th} region and $m_j, j \in \{0, 1\}$ is the mean height of all regions of the whole document image in the initial state j. A state transition occurs with high probability for regions that have a height close to the mean height.

- The emission probability matrix $B(k)$ for k^{th} region of the vertical strip is modeled by a log-normal distribution of the foreground (text) pixels density for all regions. The emission matrix is defined by the following:

$$B(k) = \begin{pmatrix} \frac{1}{x_k \sigma_0 \sqrt{2\pi}} \exp\left(-\frac{(\ln x_k - \mu_0)^2}{2\sigma_0^2}\right) \\ \frac{1}{x_k \sigma_1 \sqrt{2\pi}} \exp\left(-\frac{(\ln x_k - \mu_1)^2}{2\sigma_1^2}\right) \end{pmatrix}$$

  where x_k is the foreground pixel density for the k^{th} region. μ_j and σ_j^2 denote the mean and variance of the logarithm of foreground pixels' density for all region in the initial state j. This formulation indicates that the two states may be distinguished by considering two log-normal distributions. For estimation of the parameters of the HMM, regions with reasonable heights, over one fifth of the mean height of all CCs, are used.

- $\pi_j, j \in \{0, 1\}$, the initial state probabilities are set to be equal.

Having specified the model, a vertical zone can be seen as a succession of initial text and gaps that form observations of the model. The corresponding state sequence that characterizes the respective regions as text or gap, results with the application of Viterbi algorithm for maximization of the sequence probability. As mentioned before, separators of the text lines are drawn in the middle of each resulting gap region.

Figure 5.4 shows the initial text and gap regions and the resulting regions after applying the Viterbi algorithm for the two images in figures 5.1 and 5.2.

5.3 Connecting separators across vertical zones

Up to this point, text line separators are found in vertical zones. But these separators are line fragments that do not always meet end-to-end across different vertical zones. The objective of a connecting algorithm is to correct the location of line fragments, connect separators from adjacent vertical zones and extend them until they isolate text lines. Our connecting algorithm is slightly different from the one introduced in the original work. The steps are as follows:

- The first step of our connecting algorithm is to perturb the location of each separator parallel to the x-axis by either moving it up or down until it stays clear of any connected components or text characters. This step is overlooked in the original algorithm. The reason for this perturbation is that although the Viterbi algorithm does a good job on clearing many false separators, it introduces some other false separators over text characters that have a considerably larger height than the average height of CCs.

- The second step is to remove any separator that is located partially or completely outside of any text region detected in the previous chapter.

- The third step is to connect separators across vertical zones to one another. All separators that are not connected from the right side to any separator should search for the closest separator on the vertical zone to their right. The new founded separator should be within half the height of averaged CCs in vertical direction from the main separator, otherwise two separators that separate different text lines may be connected incorrectly. Another search should be performed likewise to connect open-ended separators to their left counterparts. If a search is unsuccessful, the separator should be left open-ended.

- The fourth step is to extend open-ended separators from left and right sides until they either reach to the contours of text regions, or they reach to the borders of the page. After traversing any new zone by extending the separator, a new search should be perform to connect the extended separator to another potentially available separator.

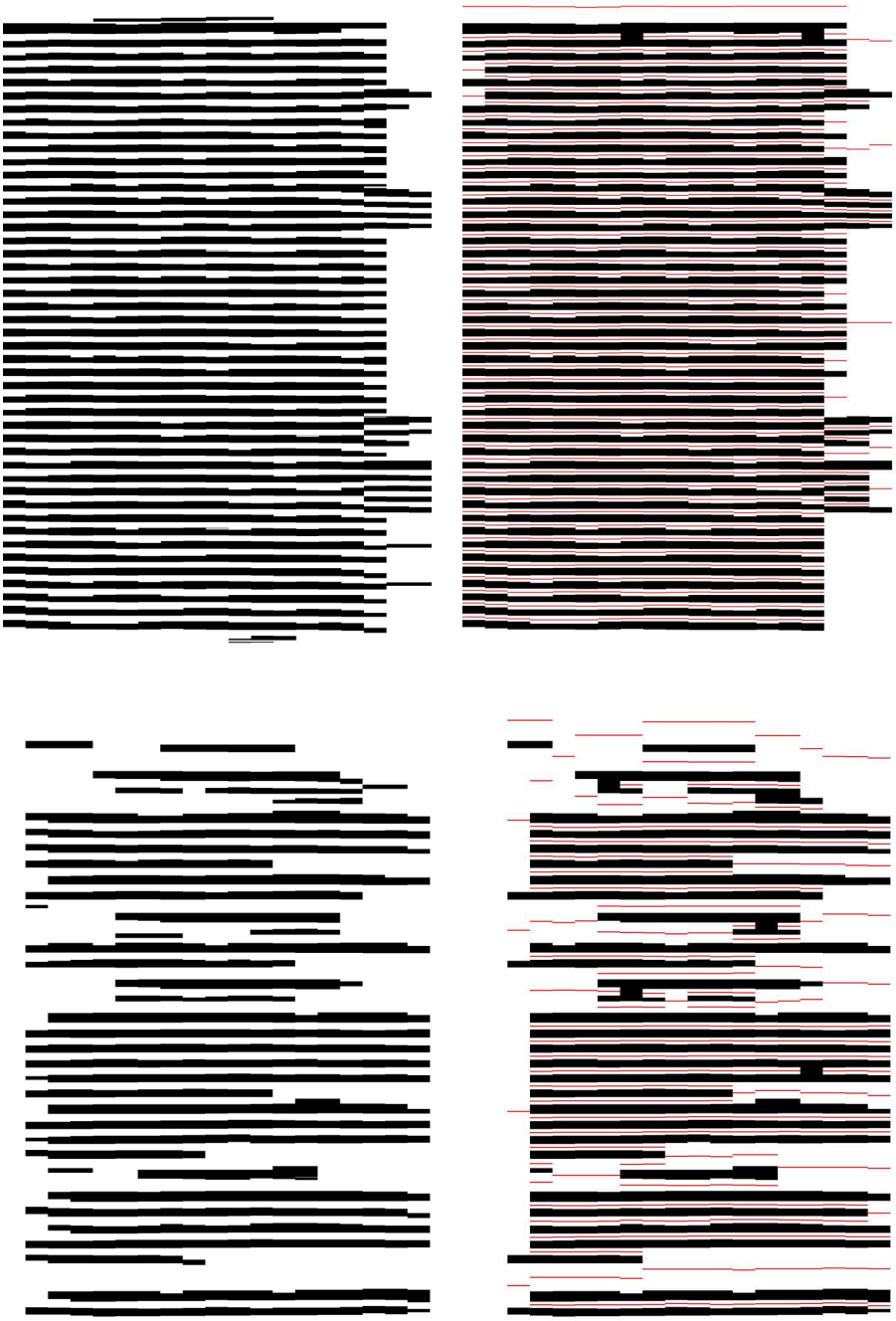


Figure 5.4: Text and gap regions after applying Viterbi algorithm and the resulting text line separators in red.

- The fifth step is to remove extended separators that have cut through more than one connected component. When two text lines are located close to each other and contain touching components, a separator might not be available at that particular location. The lack of this separator should be compensated by extending separators in the adjacent zones to pass over the touching characters and separate the two text lines. However, from time to time a separator may be found that incorrectly cuts all the way through a misaligned text line. These extended separators often cut more than one connected components and should be removed. If left untreated, they will cut a single text line into two separate text lines by passing through the middle of the characters.

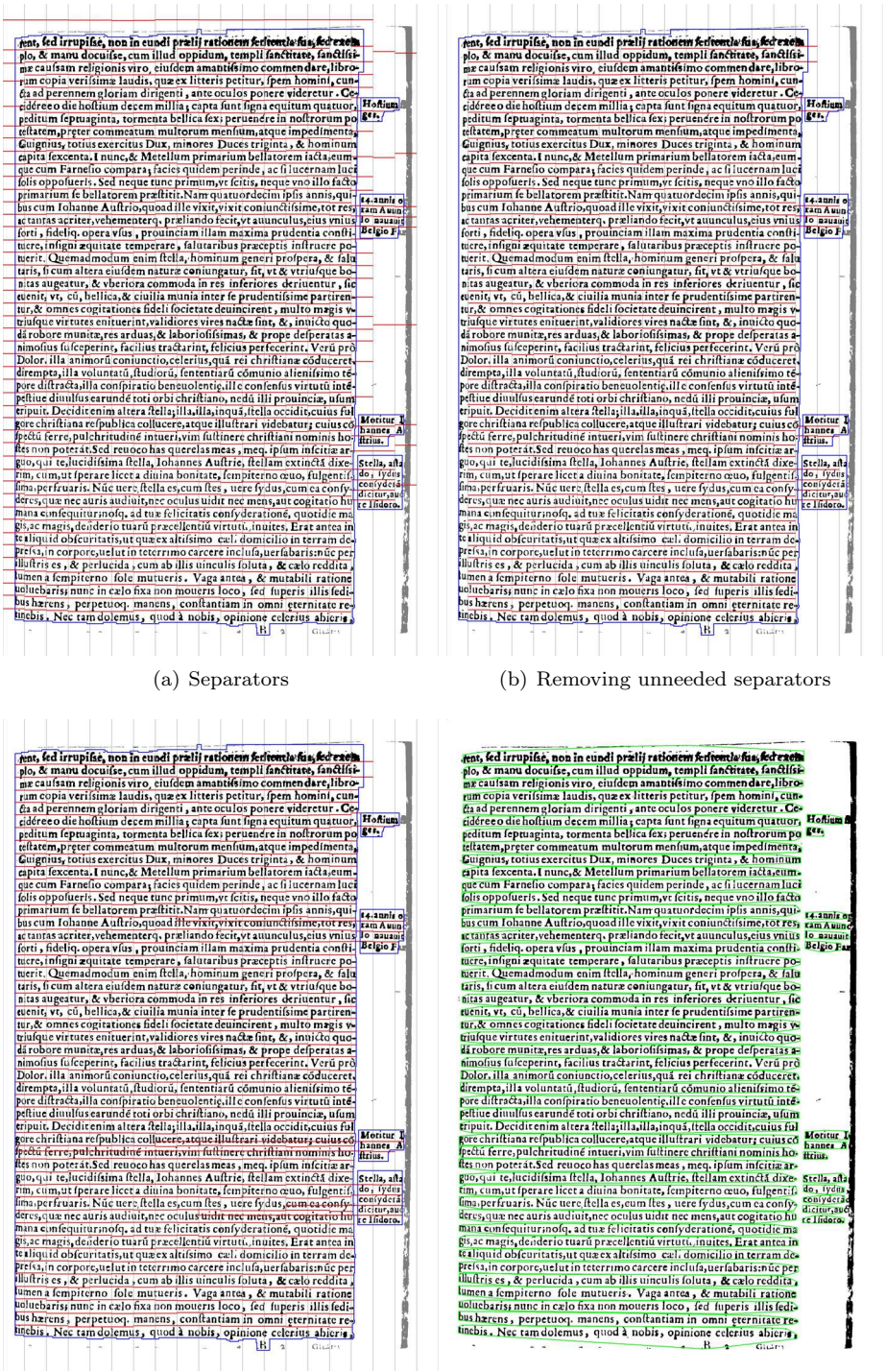
- The final step is to group connected components into text lines. This can be done by generating an image map that only contains text line separators and contours of the text regions, computed in the previous chapter. A labeling strategy is performed on this map to assign a region id to the pixels of each isolated region. Next, connected components of the page should be analyzed. If a component belongs to two regions it should be assigned to the region that includes the greater part of the component, however if the two regions have roughly equal amount of the component, as it happens in the case of touching text lines, then that component should be divided into smaller pieces and each region takes its own piece.

Figure 5.5 displays some steps mentioned above for two document images, illustrating how we get from text line separators to text lines.

5.4 Results

It is important to analyze the results of text line segmentation and know where the errors come from. Unfortunately none of the methods are comparable. For example Papavassiliou [75] has applied his method on ICDAR07 handwritten segmentation contest [39], but the dataset used in the competition contains simple double-spaced handwritten documents. These documents do not contain any side notes or multi-column text regions. So applying our method to this dataset, yields the same results as the method developed by Papavassiliou.

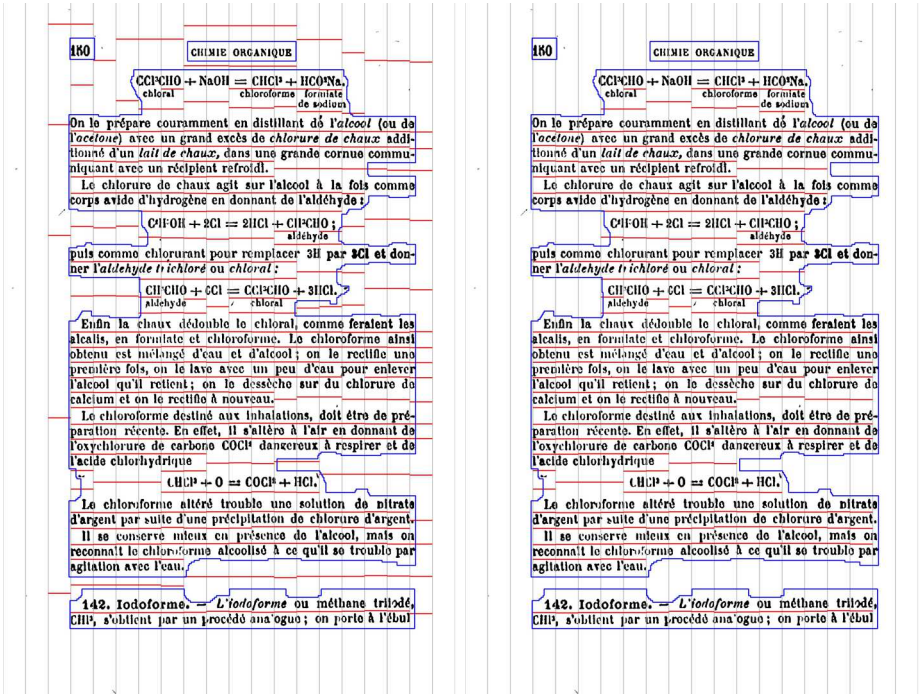
However, we apply our method on two datasets; the dataset for ICDAR2009 *Page Segmentation Competition* and the dataset for ICDAR2011 *Historical Document Layout Competition* [4]. The first dataset contains 60 document pages which reflect commonly occurring everyday documents that are likely to be scanned including pages from magazines and technical journals. The second dataset contains 100 historical documents from most national and major libraries in Europe that are very similar to our own corpus. This dataset consists of handwritten and printed documents of various types, such as books, newspapers, journals and legal documents. The 100 documents for the competition are selected as a representative of different document types with aging artifices,



(a) Separators (b) Removing unneeded separators

(c) Extending separators (d) Final text lines

Figure 5.5: Some steps to get from separators to text lines



(a) Separators (b) Removing unneeded separators

(c) Extending separators (d) Final text lines

Figure 5.5: Some steps to get from separators to text lines

dense printing, irregular spacing, side notes and varying text column widths.

Apart from our results, we managed to apply Tesseract-OCR engine to both datasets. The Tesseract-OCR engine was one of the top 3 engines in the 1995 UNLV Accuracy test. Since then it has been improved by Google and is one of the most accurate open source OCR engines available. Currently, it is one of the most popular engines that has been used by many software and methods including EPITA method that holds the second place in Historical Document Layout Competition [4]. It is worth noting that the provided command line for Tesseract-OCR can only produce layout segmentation results in hOCR format, an open standard which defines a data format for representation of OCR output by embedding this data into a standard HTML format.

Evaluation of the results is carried out by Prima Layout Evaluation Tool [25]. The parameters of the evaluation are configured to measure the pure segmentation performance. Therefore, missing and partial missing text lines are considered worst and having the highest weights of 1. The weights for merge and split errors are set to 0.5, whereas false detected text lines, as the least important error type, has a weight of 0.1. Appropriately, the errors are also weighted by the size of the affected area (excluding background pixels). In this way, a partially missed text line, having one or two missed characters, has less influence on the overall result than missing of a whole text line. These settings are taken from [4] for the evaluation of paragraph detection; however, they are also reasonable for evaluation of text line detection.

Table 5.1: LINE DETECTION SUCCESS RATES FOR 61 DOCUMENTS OF ICDAR2009 DATASET

	Area weighted error %					Area weighted %	Count weighted %
Method	Merge	Split	Miss	Partial miss	False detection	Success	Success
Our Method	5.73	4.69	3.13	4.63	3.89	93.57	71.99
Tesseract	7.53	0.95	3.32	1.67	2.00	95.01	80.21

Table 5.2: LINE DETECTION SUCCESS RATES FOR 100 DOCUMENTS OF ICDAR2011 DATASET

	Area weighted error %					Area weighted %	Count weighted %
Method	Merge	Split	Miss	Partial miss	False detection	Success	Success
Our Method	09.35	3.37	0.21	1.65	0.85	95.53	69.52
Tesseract	29.92	7.92	0.83	0.68	3.72	85.46	51.03

Table 5.3: LINE DETECTION SUCCESS RATES FOR 100 DOCUMENTS OF OUR CORPUS

	Area weighted error %					Area weighted %	Count weighted %
Method	Merge	Split	Miss	Partial miss	False detection	Success	Success
Our Method	04.80	6.96	0.42	3.23	2.28	95.04	77.43
Tesseract	14.18	2.29	0.96	2.12	6.52	92.02	66.21

The majority of errors in our method happen because of the segmentation results of region detection from the previous stage. Errors due to merging of

text lines can be divided into two categories. Merging of adjacent text lines from different columns are due to errors of region segmentation. Currently the most occurring scenario that could contribute to this error is that in some documents, pieces of broken rule lines that are supposed to separate two columns of text, are mistaken for 'I','i' and 'l' and remain as text components. These misclassified components merge two columns of text in region detection. In this situation text line separators from text line detection stage of the method extends to the boundaries of the region and eventually text lines from different columns of text are merged. Merging of parallel adjacent text lines that are located on top of each other is an error that may happen due to text line detection algorithm where two text lines are intertwined. However chances of this happening is very slim.

Table 5.1 shows that the Tesseract-OCR engine performs a slightly better job on clean on clean and well-formatted documents. However, tables 5.2 and 5.3 indicate that as datasets move toward historical documents with handwritten text lines, our method perform significantly better than Tesseract-OCT.

Chapter 6

Paragraph detection

Saving specified all text lines, in this stage of the method, we focus on regrouping these lines to produce paragraphs. At this point it is expected that text lines from different columns or marginal notes are segmented properly and they belong to separate text regions. However the text region detection method described in chapter 4 has its focus on separating text regions that are located far from one another or belong to different columns. Several factors that are overlooked are the size of the text lines and the paragraph model that includes them. For example, a text line that has a considerably larger text components compared to its adjacent text lines is expected to be a title, header or sub-header. These text lines should be separated from the rest of the text lines in a region. Moreover there are several paragraph models that exist on either handwritten or printed historical documents. They can be immediately recognized from documents based on the geometric location of text lines and their indentations. The simplest paragraph model is the model for articles, magazines and technical journals that the first line of the paragraph has a larger left indentation than the rest of the text lines. More complicated paragraph models can be found in historical documents. One such paragraph model is that except for the first text lines, all remaining text lines of the paragraph have a large left indentation. Another paragraph model which can be seen mostly in poems, has text lines that are center justified. The ending of text lines may or may not be aligned. The paragraph detection module that we describe in this section, should have the capability to handle all these cases.

Our paragraph model should be applies to each text region independently. An overview of different stages for paragraph detection is as follows:

- A *minimum spanning tree* (MST) is applied to connect all text lines in a way that only one link exists between each pair of lines. The weights of the MST is specified in a way that the natural reading order of the text lines are preserved.

- The MST is converted to a binary partition tree of text lines. The leaves of the tree represent individual text lines. The remaining nodes represent group of text lines that are obtained by merging text lines represented by

the two children. The root node represents the entire text lines of the text region.

- A set of features is extracted for each node of the binary tree. The cost to preserve a node as a paragraph is a Gaussian weighted mixture of these features. The cost to remove a node is equal to the sum of costs of preserving both child nodes.

- At each node of the tree from root to the leaves, if the cost of preserving a node is less than the cost of removing the node, that node is marked to be preserved and the rest of the children for that particular node are ignored. A dynamic programming framework is utilized to estimate these costs.

- Training is performed using a training scheme similar to Collin's voted perceptron. It tunes weights to ensure that the cost of preserving a paragraph (node) of the ground truth is less than that of its children. The cost of removing a leaf node is ∞ which ensures that no leaf node can be removed regardless of the cost for preserving the lead node.

6.1 Minimum spanning tree (MST)

All text lines in one text region are fully connected with links between them. To compute the MST, first we need to compute a weight for each link. Consider that all the text lines in figure 6.1 are part of one text region. The weight for the link between the first text line (p) and the second line (q) is defined as:

$$W_{mst}(p,q) = (1 + d(p,d))(1 + \sin(\angle_{min}(p,q))).$$

where $d(p,q)$ is the Euclidean distances between the convex hull of the text line p (red marks) and that of the text line q (blue marks). $\angle_{min}(p,q)$ is the minimum positive angle between the axis of p and axis of q. The second term ensures that if accidentally a vertical line is included in the text region, it would be the last text line to join the spanning tree.

6.2 Binary partition tree (BPT)

The goal of converting a minimum spanning tree to a binary partition tree is illustrated in figure 6.2. The root node contains all the text lines. Then at each node based on a criterion that we describe below, the algorithm chooses one of the remaining links inside the MST. The link is removed and the MST ensures that the tree breaks into two set of text lines that are not connected with any other links. Each set of lines are assigned to a child node until we reach to the

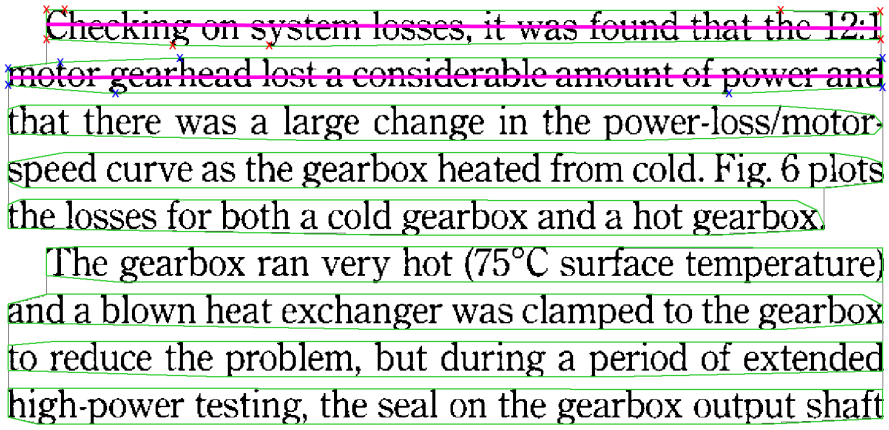


Figure 6.1: A fictional text region including its text lines .

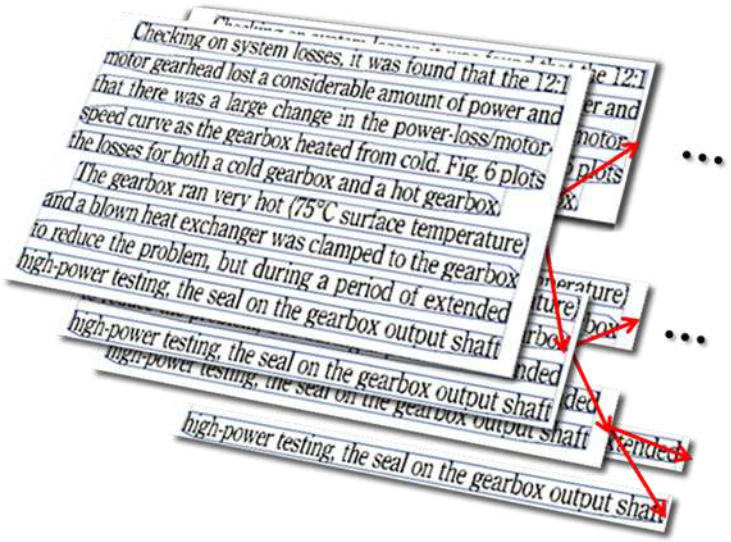


Figure 6.2: Binary partition tree generated from couple of text lines.

leaves of the tree which contain only one text line.

For each pair of lines p and q, let $a = (x_1^L, y_1^L)$ and $b = (x_1^R, y_1^R)$ be the coordinates of the axis points of the line at the top and let $e = (x_2^L, y_2^L)$ and $f = (x_2^R, y_2^R)$ be the coordinates of the axis points of the second line. Then $LL = d(a, e)$ is the Euclidean distance between the two points on the left and $RR = d(b, f)$ is the Euclidean distance between the two points on the right. The criterion for selecting a link at each node of the BPT is:

$$W_{bpt}(p, q) = (1 + \min\{LL, RR\})(1 + W_{mst}(p, q)).$$

Starting from the root node, at each level the algorithm selects the link with maximum W_{bpt} and removes that link. The two resulting set of lines are passed to the child nodes of the node in process.

6.3 Paragraph features

Now that the BPT is generated and each node of the BPT contains a potential paragraph, a set of features should be extracted from each paragraph. Features are as follows:

- Left indentation of the first line. All computations of the indentation are done in regard to the bounding box of the paragraph in question.
- Right indentation of the first line.
- Indentation ratio of the first line which is the minimum value between left and right indentations divided by the maximum value between them.
- Difference between left and right indentations of the first line.
- Left indentation of the last line.
- Right indentation of the last line.
- Indentation ratio of the last line.
- Difference between left and right indentations of the last line.
- Average value of left indentations of the text lines in the paragraph excluding the first and the last line.
- Average value of right indentations of the text lines in the paragraph excluding the first and the last line.
- Average indentation radio which is computed based on the average left and right indentations of the text lines in the middle of the paragraph.
- Maximum left indentation of the text lines in the middle of the paragraph.
- Minimum left indentation of the text lines in the middle of the paragraph.

- Maximum right indentation of the text lines in the middle of the paragraph.
- Minimum right indentation of the text lines in the middle of the paragraph.
- Average vertical distance between text lines.
- Average height of text lines.
- Standard deviation of height for text lines.
- Average width of text lines.
- Standard deviation of width fro text lines.
- Average size of connected components.
- Standard deviation of size of connected components.
- Number of connected components.
- Paragraph's width.

6.4 State decoding

State decoding is a procedure that determine the state of each node given their weights. Two possible states are defined for each node: "Preserve" or "Remove". If a node takes the "Preserve" state, that node and all its children should be grouped as one paragraph. On the other hand a "Remove" state indicates that the two children of that node should not be grouped as one paragraph. Leaves of the tree are single text lines and should always be preserved. The algorithm is described in algorithm 1. W is the weight vector that should be trained.

6.5 Training method

The training method for our paragraph model is very similar to Collin's voted perceptron in regard to computing the direction of gradients. The major difference is that the values of the features are not available beforehand and should be computed online while training. For each document image in our trainset, we start from the leaves of the tree where we already know the values of the features. Then we move upward toward the root of the tree. The implementation can be done using dynamic programming where we appoint to the root and the root ask for feature values from its children. The pseudo code for computing Δ values is provided in algorithm 2. In each iteration we initialize Δ to zero and call the procedure for the root node of the binary tree for first document. Then, we continue calling procedures for all other documents in our training dataset. After one passage for all documents, Δ should be divided by the total number of processed nodes in the whole trainset. Then each element of Δ vector should be added to the current weight vector and the next iteration the procedure starts over with the new weights and we continue until convergence.

Algorithm 1 State decoding

procedure DECODESTATES()
  $F_{preserve} \leftarrow$ feature vector from current paragraph
  if this paragraph has no children **then**
    state $\leftarrow$ "Preserve"
    $F \leftarrow F_{preserve}$
  else
    First child $\rightarrow$ *DecodeStates*()
    $F_1 \leftarrow$ feature vector from first child
    Second child $\rightarrow$ *DecodeStates*()
    $F_2 \leftarrow$ feature vector from second child
    $F_{Remove} \leftarrow F_1 + F_2$
    $C_{preserve} \leftarrow W^T F_{preserve}$ ▷ Cost of preserving this paragraph
    $C_{remove} \leftarrow W^T F_{remove}$ ▷ Cost of removing this paragraph
    if $C_{preserve} < C_{remove}$ **then**
      state $\leftarrow$ "Preserve"
      $F \leftarrow F_{preserve}$
    else
      state $\leftarrow$ "Remove"
      $F \leftarrow F_{remove}$
    end if
  end if
end procedure

6.6 Results

We apply our paragraph detection method on three datasets. These datasets include 55 documents from ICDAR2009 page segmentation competition [6], 100 documents from ICDAR2011 historical document layout analysis competition [4] and 100 documents from our own corpus. We also apply the Tesseract-OCR [89] and EPITA [4] page segmentation methods on all datasets for the purpose of comparison.

The evaluations are carried out by Prima Layout Evaluation Tool [25]. The full description of the evaluation method is noted in appendix A. Tables 6.1, 6.2 and 6.3 summarize the results. In the reported percentages, area weighted errors and success rates are more important, because they also account for the area that a violation happens.

Results show that all methods perform roughly the same on well formatted magazines and journals from ICDAR2009 dataset with our method doing slightly better.

On the other hand, on historical documents of ICDAR2011 competition dataset, our method and the EPITA perform significantly better than Tesseract-OCR with our method performing slightly better. According to this result, the reported results in [4] is updated in figure 6.3 computed based on scaled estimates.

Algorithm 2 Calculating delta

procedure CALCDELTA(Δ) ▷ Δ should be initialized with zero.
  //* refers to ground truth
  if state* = "Preserve" or this paragraph has no children **then**
    $F_{gt} \leftarrow F_{preserve}$
  else
    $F_{gt} \leftarrow F_{remove}$
  end if
  $\Delta \leftarrow \Delta + F_{gt} - F$
  if this paragraph has children **then**
    First child $\rightarrow CalcDelta(\Delta)$
    Second child $\rightarrow CalcDelta(\Delta)$
  end if
end procedure

Table 6.1: PARAGRAPH DETECTION SUCCESS RATES FOR 61 DOCUMENTS OF ICDAR2009 DATASET

	Area weighted error %					Area weighted %	Count weighted %
Method	Merge	Split	Miss	Partial Miss	False Detection	Overall	Overall
Our Method	29.45	22.92	2.84	04.31	23.94	75.55	59.82
Tesseract	14.83	35.35	1.78	03.08	11.53	73.22	56.95
EPITA	13.46	02.45	0.22	05.27	53.27	73.65	59.48

Table 6.2: PARAGRAPH DETECTION SUCCESS RATES FOR 100 DOCUMENTS OF ICDAR2011 DATASET

	Area weighted error %					Area weighted %	Count weighted %
Method	Merge	Split	Miss	Partial Miss	False Detection	Overall	Overall
Our Method	32.31	20.11	0.02	1.38	04.16	82.22	66.59
Tesseract	29.62	49.24	0.83	0.98	16.07	72.00	52.28
EPITA	30.30	17.46	1.09	6.40	16.24	81.72	61.06

For the purpose of comparison, figure 6.4 shows the results of paragraph detection on two documents from our corpus using our method and Tesseract-OCR. These documents are selected in a way that the ratio of overall success for our method divide by the ratio of overall success for Tesseract-OCR is maximized. In other words, our method performs way better on these two images compared to Tesseract-OCR.

Results indicate that Tesseract fails at processing indentations, side notes detection and also it fails to make use of rule lines and other clues such as borders and frames to correctly segment the document.

On the other side, figure 6.5 shows the results of paragraph detection on two other documents from our corpus. This time for the selection of documents, the mentioned success ratio is minimum. For the images in sub-figures *a* and *b*, the overall area weighted success rate for Tesseract is 98.8% but that of our method is 79.9%. In the second sub-figures *c* and *d*, our method misses many dots that belong to text lines and due to this problem, the success rate is lower than that

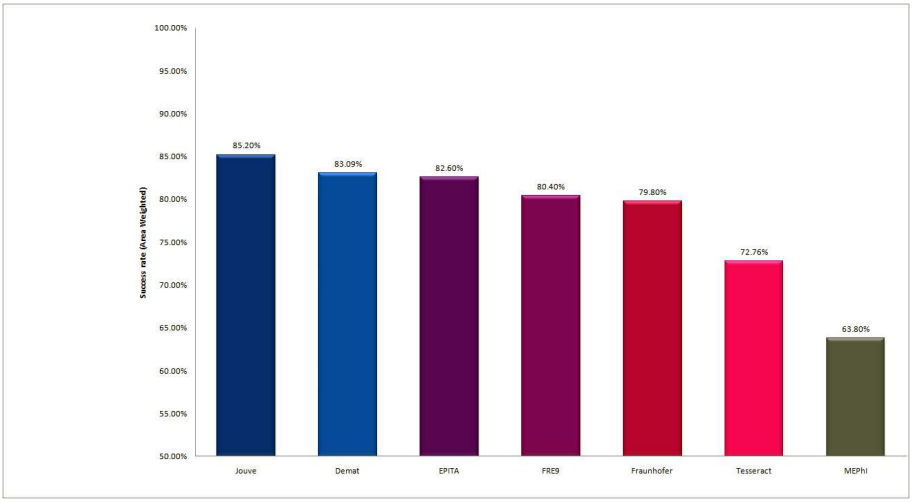
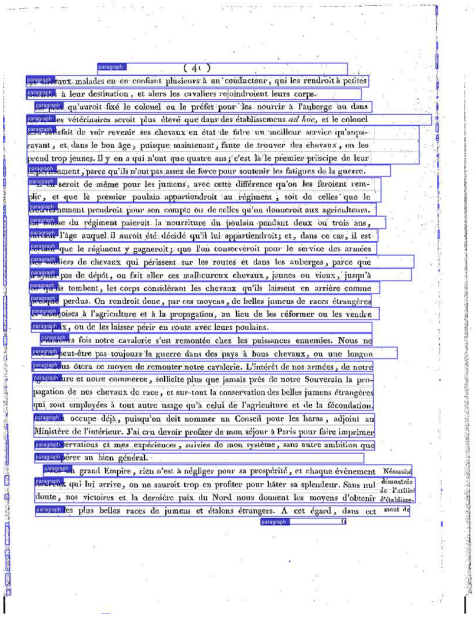


Figure 6.3: Updated results reported in [4] based on scaled estimates. In this figure, the results from our method (Demat) and Tesseract-OCR are added based on scaled estimates by using results of EPITA as a reference.

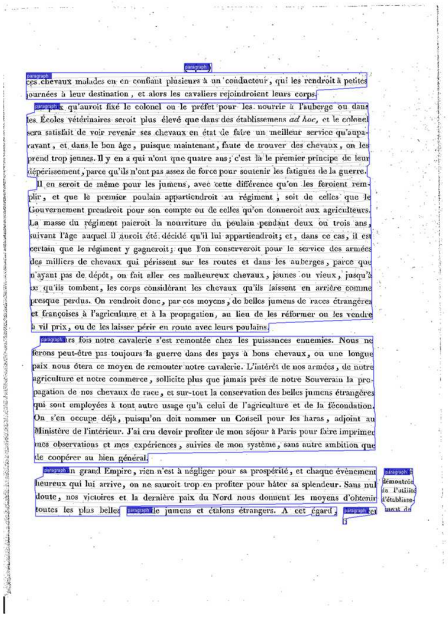
Table 6.3: PARAGRAPH DETECTION SUCCESS RATES FOR 100 DOCUMENTS OF OUR CORPUS

	Area weighted error %					Area weighted %	Count weighted %
Method	Merge	Split	Miss	Partial Miss	False Detection	Overall	Overall
Our Method	23.88	27.08	0.39	3.03	12.23	86.97	72.71
Tesseract	16.11	44.95	0.51	2.38	23.86	81.79	62.55
EPITA	23.08	19.03	0.85	8.82	12.82	88.05	67.84

of the Tessseract-OCR.



(a) Tesseract-OCR

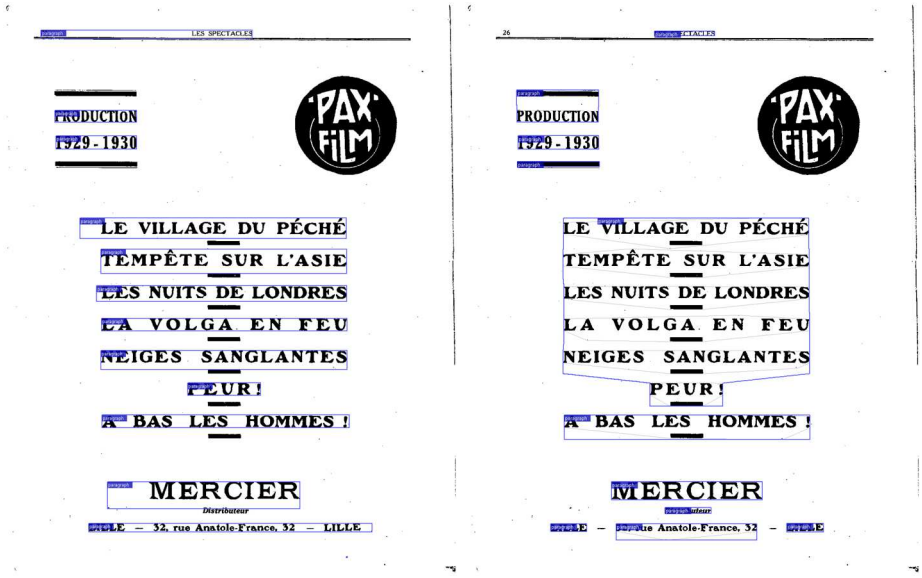


(b) Our method

(c) Tesseract-OCR

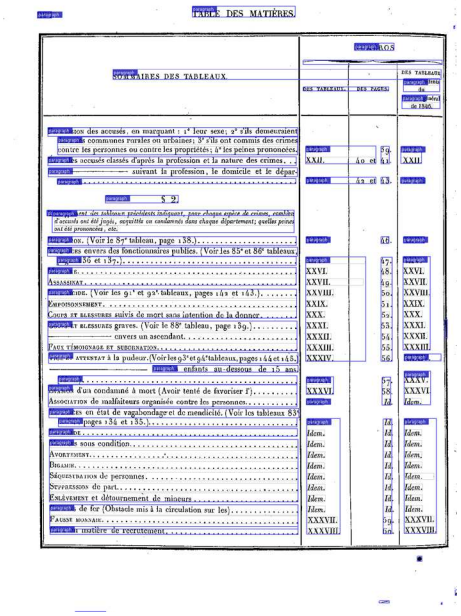
(d) Our method

Figure 6.4: This figure shows the result of paragraph detection for two documents that have obtained the best detection rate using our method and at the same time the worst detection rate using Tesseract-OCR.

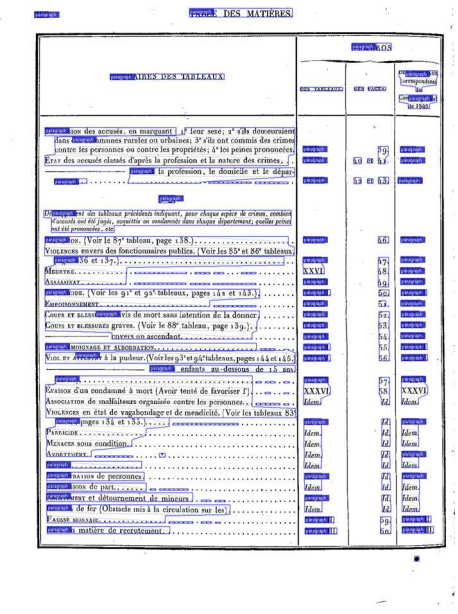


(a) Tesseract-OCR

(b) Our method



(c) Tesseract-OCR



(d) Our method

Figure 6.5: This figure shows the result of paragraph detection for two more documents that have obtained the worst detection ratio using our method than that of Tesseract-OCR.

Chapter 7

Conclusion and future work

In this thesis, we have provided a new framework for document page segmentation that brings some improvements upon current state-of-the-art methods.

Unlike most methods that apply page segmentation on all components of the page, we have presented a method to identify non-textual components and clear them from the document image before applying a page segmentation method. Moreover, the identified non-textual components such as rulers, table structures and advertisement frames are utilized again to separate text regions.

For text region detection, we have introduced a framework based on two-dimensional conditional random fields that can gather various observations from whitespace analysis, graphical components, run-lengths and separate regions of text according to these observations. As a result, we were able to improve the results of separating text columns when one is situated very close to the other. This framework also prevents the contents of a cell in a table to be merged with the contents of other adjacent cells. Furthermore, the method does not allow text regions inside a frame to be merged with other text regions around.

After carefully examining methods for text line detection that can sustain the variety and variation of text lines in handwritten and printed document, we have adopted a variant of Papavassiliou's text line detection method [75] to segment text regions into text lines.

Finally, a novel trainable method based on binary partition tree is proposed to determine the paragraph structures based on the appearances of text lines in a text region.

7.1 Future direction

As with every new work, there are many problems and challenges that can be overcome with an improvement to the methods presented in this thesis. We propose several ideas which would clearly be of benefit to this work. We are

currently working on some of these ideas to improve our results and we decided to set aside some others due to limitation in time.

For text and graphics separation, it would be a benefit to use three labels of "text","graphics" and "text containers" instead of just two labels of "text" and "non-text" components. Text containers refer to tables and frames that contain text. Currently we are doing that by using a threshold on the number of children and the solidity feature of non-textual components. Learning this using a training dataset will make it more robust to errors but it requires a modification to the ground truth data that we use.

For text region detection there is more room for improvement. Currently we are computing several observations with Gabor features with different size of the window and we send these observations to our feature functions. However, to make it more robust and multi-scale what we should do is to change the window size of the Gabor filter locally according to the local height of text connected components. To do this, we have to compute a complete set of Gabor filters and to convolve them with the original image to generate many filtering results. Then for each site, based on the average local height of text components, we have to pick the value from the filleting result that correspond to that local height. Clearly, this approach is unfeasible due to its computation burden. It would be desirable if one could benefit from a formulation and implementation of non-stationary Gabor filters in which the size of the kernel changes locally according to the local height of text components.

Feature functions in our CRF model depend on several observation maps from the image. However, the number of features are not enough to separate sites with great confidence. Also the dependency of these observation are not exploited in our framework. As a result the contribution edge potentials in our CRF model is very limited due to small number of feature functions. One can benefit from more effective features such as Ferns [74, 73] to define feature functions in CRF model.

As for inference in conditional random fields, we are using ICM [9] due to its simplicity and fast training time which is less than 4 to 6 hours. However, it is known that ICM fails to capture long-range interaction between sites' labels. We evaluated loopy belief propagation [64] in this work and it could not converge to a good solution after considerable amount of time. One may benefit from other inference methods to improve results for the CRF mdoel.

Finally, in the method for text line detection, the global median of height of text connected components are used inside the transition and emission probabilities of HMM. One can reformulate these probabilities to use the local average height of text components instead of the global statistics.

Appendix A

Performance evaluation methods

Many segmentation algorithms have been proposed in the literature. However without a common performance evaluation method, it is hard to compare the performance of these methods. Care should be taken in using a performance metric to predict how well a segmentation method will perform on a particular task.

Three popular performance evaluation methods are:

1. Precision and recall
2. Match counting
3. Scenario driven region correspondence

A.1 Precision and recall

Precision and recall are two measures that evaluate the results of a classification task. The precision for a class is the number of **true positive**[1] divided by the total number of objects that are labeled as belonging to the positive class. Recall is defined as the number of true positives divided by the total number of objects that actually belong to the positive class. In simple terms, high recall means that a the classifier returned most of the relevant results. High precision means that a classifier returned more relevant results than irrelevant.

In the context of text and graphics separation, a positive class is text and a negative class is graphics. We define four counts:

- tp (true positive) is the number of correctly classified objects as text.
- fp (false positive) is the number of unexpected text objects.
- tn (true negative) is the number of correctly objects as graphics.

[1] the number of objects that are labeled correctly as belonging to the positive class

- fn (false negative) is the number of unexpected graphical objects.

Then:

$$\begin{aligned} \text{Text precision} &= \frac{tp}{tp+fp} \\ \text{Text recall} &= \frac{tp}{tp+fn} \\ \text{Graphics precision} &= \frac{tn}{tn+fn} \\ \text{Graphics recall} &= \frac{tn}{tn+fp} \end{aligned}$$

We use these measures for performance evaluation of our text/graphics separator. It is visually desirable to use a performance metric that takes into account the area of the objects. Because of this instead of using the number of connected components, we use the number of pixels that belong to those connected components. In this way, a missing component with one pixel does not affect the classifier performance as much as a large component.

A.2 Match counting

First published in 1999, Phillips et al [77] present a methodology for evaluating graphics recognition systems that operates on images containing lines, circles, arcs and text blocks. Their method gained widespread acceptance in many scientific communities including document imaging. [4, 75, 70, 5, 6] are only some of the papers that have reported their results based on this performance evaluation method. In order to compare our results with other results reported in the literature, we also use the same evaluation method. For interested reader the best resource to understand this method in detail is still the original paper [77], but we also briefly describe the method here.

In this method, the accuracy of a segmentation algorithm is measured by counting the number of matches between entities detected by the algorithm and the entities in the ground-truth, the number of misses and under/over segmentation. These entities can be characters, text lines, or text regions. We briefly explain the steps for text lines but it can be applied to any other entities as well. The steps to compute

1. Text lines within the recognition result are indexed.
2. Text lines within the ground-truth are indexed.
3. A $MatchScore$ is computed between each detected line and each line in the ground-truth. Match scores range from 0 to 1, 1 being a perfect match. D being a text line within the set of detected lines and G being a text line from ground-truth, $D \cap G$ is the intersection of D and G. If $D \cap G$ is empty, then the match score is zero. Otherwise, the evaluator

computes the area of D, G and $D \cap G$. And the $MatchScore(d, g)$ is set to:

$$\frac{area(D \cap G)}{max(area(D), area(G))}$$

4. The computed scores are stored in a match-score matrix.

5. The performance evaluator computes a match-count table by computing a two-dimensional table from the computed match-score matrix. The entry $MatchCount(d, g)$ is set to 1 if the $MatchScore(d, g)$ is greater or equal *acceptance threshold*. Then two projection profiles, D-profile and G-profile are computed from the match-count table. The entry $D(d)$ is computed as the sum of the matches in the dth row of the match-count table. Likewise, $G(g)$ is computed as the sum of the matches in the gth row. In the original paper [77] the acceptance threshold for general purpose is set to 0.85 and also it's the same for [70] followed by the original paper. However in [75] the threshold is set to 0.95 for text lines.

6. The evaluator computes the following counts:

   - **one2one**: The number of the one-to-one matches. One to one matches happen when $D(i)$,$MatchCount(i, j)$ and $G(j)$ are all equal to one. If the segmentation algorithm produces a perfect result, all elements in the D-profile and the G-profile will be one.
   - If **g_one2many** is the number of lines in ground-truth that have more than one detected line matches.
   - **d_one2many** is the number of detected lines that have more than one line matches in ground-truths.
   - **g_many2one** is the count that many lines in ground-truth have only one detected match line.
   - **d_many2one** is the count that many detected lines have only one match in ground-truth.

7. Then, the evaluator defines three metrics. The **Detection Rate**, **Recognition Accuracy** and **F-Measure** which can be considered as weighted harmonic mean of detection rate and recognition accuracy [60]. They are defined as follows:

   $DetectionRate(DR) =$

   $$W_1 * \frac{\text{one2one}}{N} + W_2 * \frac{g\_one2many}{N} + W_3 * \frac{g\_many2one}{N}$$

   $RecognitionAccuracy(RA) =$

   $$W_4 * \frac{\text{one2one}}{M} + W_5 * \frac{d\_one2many}{M} + W_6 * \frac{d\_many2one}{M}$$

$$F - Measure = \frac{2 * RA * DR}{RA + DR}$$

where N is the number of lines in ground-truth and M is the number of recognized lines. w_1,w_2,w_3,w_4,w_5,w_6 are predetermined weights that are set to $1, 0.25, 0.25, 1, 0.25, 0.25$ respectively in [75] and to $1, 0.75, 0.75, 1, 0.75, 0.75$ in [5]. The former is more generous towards errors in results.

A.3 Scenario driven region correspondence

Scenario driven region correspondence [25] is another performance evaluation method implemented in Prima Layout Evaluation Tool. It is used as part of the performance evaluation methods in [4] and [6]. The essence of this method from a segmentation point of view is very similar to match counting. The method still computes merge, split, miss/partial miss and false detection errors. However it also takes into account the reading order of the document and assign different weights to errors in each entity depending on the situation in which an error occurs. For example, a merge between two text regions that belong to different text columns is more significant than a merge between two regions that follow each other in a single text column. The weights are set due to the scenario that a evaluation is carried out. All performance evaluations for paragraph detection are done using Prima Evaluation Tool under pure segmentation scenario.

Appendix B

Implementation and software

During the course of this PhD thesis, many applications and command line tools are developed. Some are developed for the purpose of document, data or feature visualization and some are developed to perform computation for different parts of the system. Finally, all the pieces have come together as a single unified program that can be applied on any document image using one command line tool.

The command line tool is written in C++ using both QT, OpenCV and libLBGFS libraries. The cross-platform software with 6800 lines of code is developed on a Windows machine using Microsoft Visual Studio and is ported into Linux for testing and evaluation.

The general syntax of the command line tool is:

- DematSeg [Options] FolderPath

where folder path is the location of .TIFF document images. Without any option, the application opens each document image in the folder path in a multi-threaded framework and applies page segmentation on every document image. Options can be used to generate features or to train different parts of the system. Options are:

- "-gn": This option redirects the application to process all document images in the folder path for the purpose of extracting connected components and generating features for them. All features will come together in a single file that should later be used for training. The application expects to find the corresponding XML ground truth file for each document image in the same folder. XML ground-truths should have a name consisting of the base name of the document plus one of the suffixes: "_GT", "_PrimaGT" or "pc-" as prefix (ICDAR2009 default naming). If a ground truth file is not available, the application simply ignores that document and continues processing the remaining documents in the folder.

- "-tn": This option uses the dataset that is generated with option "-gn" to train LogitBoost classifier for text/graphics separation.
- "-gr": This option redirects the application to process all document images in the folder path for the purpose of generating observations and features for two-dimensional random fields model. Same as option "-gn", the application expects to find the XML ground-truth file for each document image. All features are appended to the database for further training purposes.
- "-tr": This option uses the dataset that is generated with option "-gr" to train the two-dimensional random fields model using either Collin's voted perceptron or L-BFGS training algorithm. Selection of the training method as well as many other details of the training can be changed inside an .ini file that exists on the same folder as the executable command line.
- "-tp": This option redirects the application to process all document images in the folder path for the purpose of training the paragraph detection module. Unlike other training options, features should be computed online and no feature is saved on the storage.

The .ini file along side the executable file can be manipulated to change the behavior of the application. It is worth nothing that binarization methods, different parameters for Sauvola binarization, details of morphological operations, parallelization method, training parameters can be set or changed inside this file.

Bibliography

[1] M. Acharyya and M. K. Kundu. Document image segmentation using Wavelet scale-space features. *IEEE Transactions on Circuits and Systems*, 12(12):1117–1127, 2002.

[2] M. Agrawal and D. Doermann. Voronoi++: a dynamic page segmentation approach based on Voronoi and Docstrum features. *10th International Conference on Document Analysis and Recognition (ICDAR '09)*, pages 1011–1015, 2009.

[3] C. Andrieu, N. De Freitas, A. Doucet, and M. I. Jordan. An introduction to MCMC for machine learning. *Machine Learning*, 50(1):5–43, 2003.

[4] A. Antonacopoulos. Historical document layout analysis competition. *11th International Conference on Document Analysis and Recognition (ICDAR '11)*, 2011.

[5] A. Antonacopoulos, B. Gatos, and D. Bridson. ICDAR2007 Page segmentation competition. In *9th International Conference on Document Analysis and Recognition (ICDAR '07)*, Curtiba, Brazil, 2007.

[6] A. Antonacopoulos, S. Pletschacher, D. Bridson, and C. Papadopoulos. ICDAR2009 Page segmentation competition. In *10th International Conference on Document Analysis and Recognition (ICDAR '09)*, Salford, Manchester, UK, 2009.

[7] M. Arivazhagan, H. Srinivasan, and S. N. Srihari. A statistical approach to line segmentation in handwritten documents. *SPIE*, 2007.

[8] H. S. Baird. Background structure in document images. *International Journal of Pattern Recognition and Artificial Intelligence*, pages 1–18, 1994.

[9] J Besag. On the statistical analysis of dirty pictures. *Journal of the Royal Statistical Society. Series B*, 48(3):259–302, 1986.

[10] D. Bloomberg. Multiresolution morphological approach to document image analysis. *1th International Conference on Document Analysis and Recognition (ICDAR '91)*, 1991.

[11] D. Bradley and G. Roth. Adaptive thresholding using the integral image. *Journal of Graphics, GPU, & Game Tools*, 12(2):13–21, 2007.

[12] T. M. Breuel. Two geometric algorithms for layout analysis. *Lecture Notes In Computer Science; Vol. 2423*, 2002.

[13] T. M. Breuel. An algorithm for finding maximal whitespace rectangles at arbitrary orientations for document layout analysis. *7th International Conference on Document Analysis and Recognition (ICDAR '03)*, page 66, August 2003.

[14] T. M. Breuel. High performance document layout analysis. *2003 Symposium on Document Image Understanding Technology*, 03:209–218, 2003.

[15] S. S. Bukhari. Improved document image segmentation algorithm using multiresolution morphology. *Document Recognition and Retrieval XVIII, SPIE*, 7874:1–10, 2011.

[16] S. S. Bukhari, A. Al Azawi, F. Shafait, and T. M. Breuel. Document image segmentation using discriminative learning over connected components. *8th IAPR International Workshop on Document Analysis Systems DAS 10*, pages 183–190, 2010.

[17] S. S. Bukhari, F. Shafait, and T. M. Breuel. Segmentation of curled textlines using active contours. *DAS '08. The Eighth IAPR International Workshop on Document Analysis Systems, 2008*, pages 270–277, 2008.

[18] S. S. Bukhari, F. Shafait, and T. M. Breuel. Coupled snakelet model for curled textline segmentation of camera-captured document images. *10th International Conference on Document Analysis and Recognition (ICDAR '09)*, pages 61–65, July 2009.

[19] S. S. Bukhari, F. Shafait, and T. M. Breuel. Ridges based curled textline region detection from grayscale camera-captured document. *Lecture Notes In Computer Science*, 5702:173–180, 2009.

[20] R. H. Byrd, P. Lu, J. Nocedal, and C. Zhu. A limited memory algorithm for bound constrained optimization. *SIAM Journal on Scientific Computing*, 16(5):1190, 1995.

[21] L. Caponetti, C. Castiello, and P. Górecki. Document page segmentation using neuro-fuzzy approach. *Journal of Applied Soft Computing*, 8(1):118–126, January 2008.

[22] R. Cattoni, T. Coianiz, S. Messelodi, and C. M. Modena. Geometric layout analysis techniques for document image understanding: a review. *ITC-irst Technical Report*, pages 9703–09, 1998.

[23] S. Cha and S. N. Srihari. System that identifies writers. *7th National Conference on Artificial Intelligence and 20th Conference on Innovative Applications of Artificial Intelligence*, 2000.

[24] R. Chandrashekara, R. H. Mohiaddin, and D. Rueckert. Analysis of 3-D myocardial motion in tagged MR images using nonrigid image registration. *IEEE Transactions on Medical Imaging*, 23(10):1245–50, October 2004.

[25] C. Clausner, S. Pletschacher, and A. Antonacopoulos. Scenario driven in-depth performance evaluation of document layout analysis methods. In *11th International Conference on Document Analysis and Recognition (ICDAR '11)*, pages 1404–1408. IEEE, 2011.

[26] M. Collins. Discriminative training methods for hidden Markov models: theory and experiments with perceptron algorithms. *The ACL02 conference on Empirical methods in natural language processingVolume 10*, 10(July):1–8, 2002.

[27] S. Deng, S. Latifi, and E. Regentova. Document segmentation using polynomial spline Wavelets. *Pattern Recognition*, 34(12):2533–2545, 2001.

[28] D. Doermann. *Document image understanding: integrating recovery and interpretation*. PhD thesis, University of Maryland, College Park, 1993.

[29] X. Du, W. Pan, and T. D. Bui. Text line segmentation in handwritten documents using Mumford-Shah model. *Pattern Recognition*, 42(12):3136–3145, December 2009.

[30] K. Etemad, D. Doermann, and R. Chellappa. Multiscale segmentation of unstructured document pages using soft decision integration. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, l19(1):92–96, 1997.

[31] P. F. Felzenszwalb and D. R. Huttenlocher. Efficient belief propagation for early vision. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition 2004 CVPR 2004*, volume 1 of *Int. J. Comput. Vis. (Netherlands)*, pages 41–54. Springer, 2004.

[32] S. Ferilli, M. Biba, F. Esposito, and T. Basile. A distance-based technique for non-Manhattan layout analysis. *10th International Conference on Document Analysis and Recognition (ICDAR '09)*, pages 231–235, July 2009.

[33] L. A. Fletcher and R. Kasturi. A robust algorithm for text string separation from mixed text/graphics images. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 10(6):910–918, 1988.

[34] G. D. Forney. The Viterbi algorithm. *IEEE*, 61(3):42–52, 1973.

[35] Y. Freund and R. Schapire. A desicion-theoretic generalization of online learning and an application to boosting. *EuroCOLT '95 Proceedings of the Second European Conference on Computational Learning Theory*, 904(1):23–37, 1995.

[36] J. Friedman, T. Hastie, and R. Tibshirani. Additive logistic regression: a statistical view of boosting. *The Annals of Statistics*, 28(2):337–407, 2000.

[37] J. Friedman, T. Hastie, and R. Tibshirani. *The elements of statistical learning*, volume 1 of *Springer Series in Statistics*. Springer, 2001.

[38] B. Gatos, A. Antonacopoulos, and N. Stamatopoulos. Handwriting segmentation contest. *9th International Conference on Document Analysis and Recognition (ICDAR '07)*, 2:1284–1288, 2007.

[39] B. Gatos, A. Antonacopoulos, and N. Stamatopoulos. ICDAR2007 Handwriting segmentation contest. In *9th International Conference on Document Analysis and Recognition (ICDAR '07)*, pages 1284–1288, Curitiba, Brazil, 2007. IEEE Computer Society.

[40] B. Gatos, N. Stamatopoulos, and G. Louloudis. ICDAR2009 Handwriting segmentation contest. *10th International Conference on Document Analysis and Recognition (ICDAR '09)*, 2011.

[41] B. K. P. Horn. Shape from shading: a method for obtaining the shape of a smooth opaque object from one view. Technical report, Massachusetts Institute of Technology, Cambridge, MA, USA, 1970.

[42] R. Jiang, F. Qi, L. Xu, G. Wu, and K. Zhu. A learning-based method to detect and segment text from scene images. *Journal of Zhejiang University Science A*, 8(4):568–574, 2007.

[43] M. I. Jordan, Z. Ghahramani, T. S. Jaakkola, and L. K. Saul. An introduction to variational methods for graphical models. *Machine Learning*, 233:183–233, 1999.

[44] N. Journet, J. Ramel, R. Mullot, and V. Eglin. Document image characterization using a multiresolution analysis of the texture: application to old documents. *International Journal of Document Analysis and Recognition (IJDAR '08)*, 11(1):9–18, June 2008.

[45] K. Khurshid. *Analysis and retrieval of historical document images*. PhD thesis, Universite Paris Descartes, 2009.

[46] K. I. Kim, K. Jung, and J. H. Kim. Texture-based approach for text detection in images using support vector machines and continuously adaptive mean shift algorithm. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 25(12):1631–1639, December 2003.

[47] K. Kise, M. Iwata, and K. Matsumoto. On the application of Voronoi diagrams to page segmentation. In *Workshop on Document Layout Interpretation and its Applications (DLIA99)*, pages 6–9. Citeseer, 1998.

[48] K. Kise, A. Sato, and M. Iwata. Segmentation of page images using the area Voronoi diagram. *Computer Vision and Image Understanding*, 70pp:370–382, 1998.

[49] S. Kumar, R. Gupta, N. Khanna, S. Chaudhury, and S. D. Joshi. Text extraction and document image segmentation using matched wavelets and MRF model. *IEEE Transactions On Image Processing*, 16(8), 2007.

[50] J. Lafferty, A. McCallum, and F. C. N. Pereira. Conditional random fields: probabilistic models for segmenting and labeling sequence data. *8th International Conference on Machine Learning*, pages(1):282–289, 2001.

[51] A. H Land and A. G. Doig. An automatic method of solving discrete programming problems. *Econometrica: Journal of the Econometric Society*, 28(3):497–520, 1960.

[52] Y. Li, Y. Zheng, D. Doermann, and S. Jaeger. A new algorithm for detecting text line in handwritten documents. In *10th International Workshop on Frontiers in Handwriting Recognition (IWFHR 2006)*, pages 35–40, 2006.

[53] Y. Li, Y. Zheng, D. Doermann, and S. Jaeger. Script-independent text line segmentation in freestyle handwritten documents. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 30(8):1313–1329, 2008.

[54] L. Likforman-Sulem, A. Zahour, and B. Taconet. Text line segmentation of historical documents: a survey. *9th International Conference on Document Analysis and Recognition (ICDAR '07)*, 9(2-4):123–138, September 2006.

[55] C. Liu and J. Nocedal. On the limited memory BFGS method for large scale optimization. *Journal of Mathematical Programming*, 45(1-3):503–528, 1989.

[56] G. Louloudis, B. Gatos, I. Pratikakis, and C. Halatsis. A block-based hough transform mapping for text line detection in handwritten documents. In *10th International Workshop on Frontiers in Handwriting Recognition (IWFHR 2006)*, number i, pages 515–520, 2006.

[57] G. Louloudis, B. Gatos, I. Pratikakis, and C. Halatsis. Text line detection in handwritten documents. *Pattern Recognition*, 41(12):3758–3772, December 2008.

[58] G. Louloudis, B. Gatos, I. Pratikakis, and C. Halatsis. Text line and word segmentation of handwritten documents. *Pattern Recognition*, 42(12):3169–3183, December 2009.

[59] S. M. Lucas. ICDAR2005 Text locating competition results. *8th International Conference on Document Analysis and Recognition (ICDAR '05)*, pages 80–84, 2005.

[60] J. Makhoul, F. Kubala, R. Schwartz, and R. Weischedel. Performance measures for information extraction. *In Proceedings of DARPA Broadcast News Workshop*, page 249, 1999.

[61] V. Malleron, V. Eglin, and H. Emptoz. Hierarchical decomposition of handwritten manuscripts layouts. *CAIP '09 Proceedings of the 13th International Conference on Computer Analysis of Images and Patterns*, pages 221–228, 2009.

[62] R. Manmatha and J. Rothfeder. A scale space approach for automatically segmenting words from historical handwritten documents. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, tell27(8):1212–1225, 2005.

[63] I. Mierswa, M. Wurst, R. Klinkenberg, M. Scholz, and T. Euler. YALE: rapid prototyping for complex data mining tasks. In *12th ACM SIGKDD international conference on Knowledge discovery and data mining*, volume 2006 of *KDD '06*, pages 935–940. University of Canterbury, ACM, 2006.

[64] K. P. Murphy, Y. Weiss, and M. I. Jordan. Loopy belief propagation for approximate inference: an empirical study. In *Uncertainty in AI*, volume 9, pages 467–475. Citeseer, 1999.

[65] G. Nagy. Twenty years of document image analysis in PAMI. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(1):38–62, 2000.

[66] G. Nagy, S. Seth, and M. Viswanathan. A prototype document image analysis system for technical journals. *Journal of Computer*, 25(7):10–22, July 1992.

[67] C. Nerrou, A. Glavieux, and P. Thitimajshima. Near Shannon limit error-correcting coding and decoding: Turbo-Codes. *IEEE International Conference on Communnications*, 2(1):1064–1070, 1993.

[68] W. Niblack. *An introduction to digital image processing.* Strandberg Publishing Company Birkeroed, Denmark, October 1985.

[69] S. Nicolas, T. Paquet, and L. Heutte. Markov random field models to extract the layout of complex handwritten documents. In *Tenth International Workshop on Frontiers in Handwriting Recognition (2006)*, 2006.

[70] N. Nikolaou and M. Makridis. Segmentation of historical machine-printed documents using adaptive run length smoothing and skeleton segmentation paths. *Journal of Image and Vision Computing*, 28(4):590–604, April 2010.

[71] L. O'Gorman. The document spectrum for page layout analysis. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 15(11), 1993.

[72] N. Okazaki. libLBFGS: a library of Limited-memory Broyden-Fletcher-Goldfarb-Shanno (L-BFGS), 2010.

[73] M. Ozuysal, M. Calonder, V. Lepetit, and P. Fua. Fast keypoint recognition using random ferns. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 32(3):448–461, 2010.

[74] M. Ozuysal, P. Fua, and V. Lepetit. Fast keypoint recognition in ten lines of code. *2007 IEEE Conference on Computer Vision and Pattern Recognition*, 1(R56):1–8, 2007.

[75] V. Papavassiliou, T. Stafylakis, V. Katsouros, and G. Carayannis. Handwritten document image segmentation into text lines and words. *Pattern Recognition*, 43(1):369–377, January 2010.

[76] J. Pearl. Reverend Bayes on inference engines: a distributed hierarchical approach. In D L Waltz, editor, *The AAAI National Conference on AI*, pages 133–136. Morgan Kaufmann, 1982.

[77] I.T. Phillips and A. K. Chhabra. Empirical performance evaluation of graphics recognition systems. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 21(9):849–870, 1999.

[78] L. R. Rabiner. A tutorial on hidden Markov models and selected applications in speech recognition. *IEEE*, 77(2):257–286, 1989.

[79] Z. Razak, K. Zulkiflee, M. Yamani, I. Idris, E. M. Tamil, M. Noorzaily, M. Noor, R. Salleh, M. Yaacob, and Z. M. Yusof. Offline handwriting text line segmentation: a review. *International Journal of Computer Science and Network Security*, 8(7):12–20, 2008.

[80] M. D. Riley. Time-frequency representations for speech signals. Technical report, 1987.

[81] F. Rosenblatt. The perceptron: a probabilistic model for information storage and organization in the brain. *Psychological Review*, 65(6):386–408, 1958.

[82] J. Sauvola and M. Pietikainen. Adaptive document image binarization. *Pattern Recognition*, 33(2):225–236, 2000.

[83] F. Shafait and T. M. Breuel. Document image dewarping contest. *International Workshop on Camera-Based Document Analysis and Recognition*, 2007.

[84] F. Shafait, D. Keysers, and T. M. Breuel. Performance evaluation and benchmarking of six page segmentation algorithms. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 30(6):941–954, 2008.

[85] F. Shafait, J. van Beusekom, D. Keysers, and T. M. Breuel. Background variability modeling for statistical layout analysis. *2008 19th International Conference on Pattern Recognition*, pages 1–4, December 2008.

[86] F. Shafait, J. Van Beusekom, D. Keysers, and T. M. Breuel. Structural mixtures for statistical layout analysis. In *2008 The Eighth IAPR International Workshop on Document Analysis Systems*, pages 415–422. IEEE, September 2008.

[87] Z. Shi, S. Setlur, and V. Govindaraju. Text extraction from gray scale historical document images using adaptive local connectivity map. *8th International Conference on Document Analysis and Recognition (ICDAR '05)*, pages 794–798, 2005.

[88] Z. Shi, S. Setlur, and V. Govindaraju. A steerable directional local profile technique for extraction of handwritten arabic text lines. *10th International Conference on Document Analysis and Recognition (ICDAR '09)*, 2009.

[89] R. Smith. An overview of the Tesseract OCR engine. *9th International Conference on Document Analysis and Recognition (ICDAR '07)*, 2:629–633, 2007.

[90] T. Stafylakis, V. Papavassiliou, V. Katsouros, and G. Carayannis. Robust text-line and word segmentation for handwritten documents images. *2008 IEEE International Conference on Acoustics, Speech and Signal Processing*, pages 3393–3396, March 2008.

[91] M. Stamp. A revealing introduction to hidden Markov models. Technical report, Department of Computer Science San Jose State University, 2004.

[92] T. Su, T. Zhang, and D. Guan. Corpus-based HIT-MW database for offline recognition of general-purpose Chinese handwritten text. *9th International Conference on Document Analysis and Recognition (ICDAR '07)*, 10(1):27–38, March 2007.

[93] H. M. Sun. Page segmentation for Manhattan and non-Manhattan layout documents via selective CRLA. *8th International Conference on Document Analysis and Recognition (ICDAR '05)*, pages 116–120, 2005.

[94] C. Sutton and A. McCallum. An introduction to conditional random fields for relational learning. In Lise Getoor and Ben Taskar, editors, *Introduction to Statistical Relational Learning*, volume 7 of *Adaptive Computation and Machine Learning*, chapter 4, page 93. The MIT Press, 2006.

[95] M. S. Taylor, F. S. Brundick, and A. E. Brodeen. A statistical approach to the generation of a database for evaluating OCR software. *International Journal on Document Analysis and Recognition*, 4:170–176, 2002.

[96] K. Tombre, S. Tabbone, and L. Pélissier. Text/graphics separation revisited. *DAS '02 Proceedings of the 5th International Workshop on Document Analysis Systems V*, pages 200–211, August 2002.

[97] A Viterbi. Error bounds for convolutional codes and an asymptotically optimum decoding algorithm. *IEEE Transactions on Information Theory*, 13(2):260–269, 1967.

[98] F. M. Wahl, K. Y. Wong, and R. G. Casey. Block segmentation and text extraction in mixed text/image documents. *Computer Graphics and Image Processing*, 20(4):375–390, December 1982.

[99] B. Waked. *Page segmentation and identification for document image analysis*. PhD thesis, Concordia University, Montreal, Canada, 2001.

[100] Y. Weiss. Correctness of local probability propagation in graphical models with loops. *Neural Computation*, 12(1):1–41, 2000.

[101] K. Y. Wong, R. G. Casey, and F. M. Wahl. Document analysis system. *IBM Journal of Research and Development*, 26(6):647–656, 1982.

[102] Y. Xiao and H. Yan. Text region extraction in a document image based on the Delaunay tessellation. *Pattern Recognition*, 36(3):799–809, March 2003.

[103] F. Yin and C. Liu. Handwritten Chinese text line segmentation by clustering with distance metric learning. *Pattern Recognition*, 42(12):3146–3157, 2009.

[104] L. A. Zadeh. A simple view of the Dempster-Shafer theory of evidence and its implication for the rule of combination. *The AI Magazine*, 7(2):85–90, July 1986.

[105] A. Zahour, B. Taconet, L. Likforman-Sulem, and W. Boussellaa. Overlapping and multi-touching text-line segmentation by block covering analysis. *Pattern Analysis and Applications*, 12(4):335–351, July 2009.

Index