

PhD THESIS

*A dissertation submitted in partial fulfilment of the requirements
to obtain the title of*

Doctor of Université Paris Descartes

UFR Mathématiques et Informatique

Analysis and Retrieval of Historical Document Images

By

Khurram KHURSHID

2009

Defended on 15th of December 2009 in front of the following jury:

Dr	Georges	STAMON	Université Paris Descartes	Chair
Dr	Salvatore	TABBONE	Université Nancy 2	Reviewer
Dr	Michel	CRUCIANU	CNAM, Paris	Reviewer
Dr	J. Marc	OGIER	Université de la Rochelle	Examiner
Dr	Claudie	FAURE	CNRS, Télécom Paris Tech	Supervisor
Dr	Nicole	VINCENT	Université Paris Descartes	Supervisor

Acknowledgements

This thesis is the result of three years of devoted work which would not have been possible without the support of many. Here, I would like to express my thanks to people who have been very helpful to me during the course of this work.

First of all, I would like to take this opportunity to gratefully acknowledge the wholehearted supervision of my very learned advisors, **Dr. Nicole Vincent** and **Dr. Claudie Faure**, who complemented each other wonderfully well. I count myself very lucky to have these two, who are ranked amongst the best in the domain, as my supervisors. I was always led by their skillful guidance and helpful suggestions, which made it possible for me to go this far. Dr. Nicole Vincent, being my internal supervisor, was always there for help and long discussions whenever I got stuck in something. The patience, dedication, and constant encouragement of my supervisors made it possible for me to deliver a dissertation of appreciable quality and standard.

I would also like to thank **Dr. Salvatore Tabbone** and **Dr. Michel Crucianu** for accepting to review my dissertation and providing me with valuable comments and suggestions which assisted a great deal in improving the quality of this thesis considerably. Also many thanks to **Dr. J. Marc Ogier** for accepting to be a part of jury to evaluate my work.

Special thanks are due to the co-director of research group SIP, **Dr. Georges Stamon** for providing me with valuable guidance and support at various stages of my work. His continual encouragement provided the much needed motivation, at every step, during the three years of my stay in the lab.

My cordial appreciation extends to **Dr. Nicolas Lomenie** and **Dr. Florence Cloppet** for providing a good research environment and extending worthwhile support and constructive suggestions. My stay at SIP would not have been such a pleasurable one without the presence of friendly **colleagues**, and here I have to specially mention **Imran Siddiqi** and **Hassan Chouaib**, who helped me out in one way or another and exchanged fruitful views from time to time.

I owe my thanks to all my friends, specially, **Shoab, Hussain, all Kashif's, Shehzad, Rauf sb, Sarmad and Aamir** for helping me get through difficult times, and for all the support, camaraderie and care they provided.

I would also like to express my deepest gratitude to the **Higher Education Commission** of Pakistan for financially supporting the study and to my employer **Institute of Space Technology Pakistan**, and here I would like to mention the names of **Mr Raza Hussain** (Chairman SUPARCO), **Mr Imran Rehman** (VC IST), **Mr Ayyaz Aziz** and **Mr Arbab Mehmood**, for providing me their full support and backing as well as time, thus making it possible for me to pursue my research in a prestigious institution of esteemed repute.

And finally, I am forever indebted to my extremely loving **parents** for their patience, understanding and immense love, and my brother **Khawar** and my loving sister **Kiran**, alleviating my family responsibilities and encouraging me to concentrate on my study.

Thank you all !

Abstract

Libraries across the world hold huge numbers of historical printed documents. By digitizing these documents, their content can be preserved and made available to a large community via the internet or other electronic media. Such data can nowadays be shared relatively easily, but they are often large, unstructured, and only available in image formats, which makes them difficult to access. In particular, finding specific locations of interest in a historical document image collection is generally very tedious without some sort of index or other direct access tool.

The current/obvious solution for this problem is to manually annotate a historical collection, which is very costly in terms of time and money. In this work we explore some automatic techniques that would allow the retrieval of document images with both ASCII as well word-image queries, and would help in automatic indexing/annotation of the document images. For that matter, Word spotting: an approach that finds all similar word images based on the query given to the system, has been thoroughly examined along with different character's string matching techniques.

The main contributions of this work are a detailed examination of retrieval approaches for historical documents, the development of a document image retrieval framework that allows text and image queries for information searching, and also an automatic figure/caption pair indexing model by employing a fusion of symbolic and spatial information in the document image. Building such a system involves challenges on numerous levels: the noisy historical printed documents require adequate image pre-processing, binarization, segmentation and representation techniques, as well as a robust and scalable retrieval framework. We describe the construction of a prototype system, which demonstrates the feasibility of the proposed techniques for a large collection of document images. The system extends the field of document analysis and retrieval and enhances the importance of digital libraries manifolds in the field of research

The work has been done in collaboration with BIUM (Bibliothèque Interuniversitaire de Médecine, Paris) as they provided us with their historical Medic@ document base that gave us a chance to work on documents of different periods of our choice and interest.

Table of Contents

Acknowledgements	iii
Abstract	v
Table of Contents	vii
List of Figures	xi
List of Tables	xvii
List of Abbreviations and Terminologies	xix
Chapter 1 Introduction	1
1.1 Motivation behind the research.....	1
1.2 Digital Libraries and Historical Documents.....	2
1.2.1 Document Preservation and Retrieval projects.....	6
1.2.2 Online access to Digital Documents.....	7
1.3 Context of our Work.....	8
1.3.1 Data Sets.....	8
1.3.2 Global Overview of our subject.....	11
1.3.3 Major Contributions.....	13
1.3.4 Organization of the thesis.....	13
Chapter 2 State of the art	15
2.1 Document Image Segmentation.....	16
2.1.1 Top-down Techniques.....	16
2.1.2 Bottom-up Techniques.....	17
2.1.3 Hybrid Techniques.....	18
2.2 Information retrieval – word spotting.....	19
2.2.1 Holistic Analysis techniques.....	20
2.2.2 Analytical Recognition techniques.....	27
2.2.3 Comparison of the Retrieval methods - Discussion.....	31
Chapter 3 Document Image Indexing	35
3.1 Document Image Binarization.....	35
3.1.1 Well known sliding-window-based methods in literature.....	37
3.1.2 Proposed Method – NICK.....	40
3.1.3 Comparison of the different methods.....	42
3.2 Text/ Graphic segmentation and Extraction of Words.....	49
3.2.1 Run Length Smoothing Algorithm.....	50
3.2.2 Component Height-Area Analysis.....	52
3.3 Character segmentation.....	56
3.3.1 Post-processing for character extraction.....	57

3.3.2 Evaluation of Character Segmentation method.....	61
3.4 Feature Extraction.....	64
3.4.1 Features.....	65
3.5 Image Indexing.....	72
3.6 Conclusion.....	73
Chapter 4 Information Retrieval - Word Spotting.....	75
4.1 Introduction.....	75
4.1.1 Overview – basic idea.....	76
4.2 Query Formation.....	77
4.2.1 Query image - Crisp selection using GUI.....	77
4.2.2 ASCII Query.....	78
4.3 Length-ratio Filter.....	79
4.4 Information Retrieval - Word Spotting.....	81
4.5 S-character Matching.....	82
4.5.1 Dynamic Time Warping (DTW).....	85
4.5.2 Feature distance Normalization.....	89
4.6 Word Matching.....	94
4.6.1 Relative Position Correspondence (RPC).....	95
4.6.2 Edit Distance.....	100
4.6.3 Merge-Split Edit Distance.....	103
4.6.4 Linear Displacement Matching.....	110
4.6.5 Computational comparison: Linear Matching and Merge-Split Edit distance.....	114
4.7 Experimental Results.....	115
4.7.1 Performance Measures.....	115
4.7.2 Performance evaluation of the different feature sequences.....	117
4.7.3 Comparison of different Word matching methods.....	119
4.7.4 Experimental analysis using highly degraded 16 th century documents.....	128
4.8 Conclusion.....	134
Chapter 5 Multi-context Applications.....	135
5.1 Figure/Captions Retrieval.....	135
5.1.1 Selection of figure caption candidates.....	136
5.1.2 Spatio-symbolic information fusion.....	137
5.1.3 Caption Retrieval results.....	139
5.1.4 Conclusion.....	140
5.2 Application on contemporary documents.....	141
5.3 Application on Cursive oriental scripts – Arabic/ Urdu.....	143
Chapter 6 Conclusion and Perspectives.....	147

6.1.1 Summary of the method	147
6.1.2 Future work – perspectives.....	148
Appendix A Sample Images.....	151
A.1 BIUM Sample images of the 12 books of 18 th and early 19 th century.....	151
A.2 BIUM Sample images of the older books of the 16 th century	153
A.3 Contemporary document images	154
Appendix B Graphical Interface of our system	155
Appendix C Summary in French	158
Bibliography.....	171
Author’s Publications.....	181



List of Figures

Figure 1.1 - Digitization process of a document image: from a) crude scan image to b) the displayed gray image formed using commercial tools	3
Figure 1.2 - Marco Polo's "Le Livre des Merveilles", Latin edition of 14 th century	3
Figure 1.3 - Picture of the Gutenberg Bible owned by the US Library of Congress.....	4
Figure 1.4 - A degree issued by the University of Paris, 1739.....	5
Figure 1.5 - a) The oldest known document printed in Australia, a theatre playbill from the former British colony dating back to 1796. b) A document image from a 19 th century book in medic@ digital library [BIUM].....	5
Figure 1.6 - The Chinese Diamond Sutra, the oldest known dated printed book in the world.....	6
Figure 1.7 - Screen shot of the Google book search service where a book is opened and its text can be searched using the text search option.....	7
Figure 1.8 – An image from a 16 th century book by Bassaei available on BIUM web.....	10
Figure 1.9 - A BIUM document image of 19 th century: a) in web resolution images the digitization effects are more visible as compared to the b) original images.....	10
Figure 1.10 – Global overview of the document image retrieval system.....	11
Figure 1.11 - Four stages of document image processing for indexing.....	12
Figure 1.12 – Examples of characters with segmentation inconsistencies; some T-characters are divided into two S-characters (first four examples) or two T-characters are merged into one S-character (last example)	12
Figure 2.1 - Different artifacts occurring in historical document images (<i>Rath, 2005</i>)	15
Figure 2.2 - Extraction of characters using a combination of vertical and horizontal projection profiles (<i>Antonacopoulos and Karatzas, 2005</i>)	16
Figure 2.3 - Segmentation of figures and figure caption candidates by connected component analysis (<i>Faure and Vincent, 2009</i>).....	17
Figure 2.4 - Curled text lines segmentation using Active contours (<i>Bukhari et al. , 2008</i>)	18
Figure 2.5 - Segmentation in 3 and 4 classes by textural approach (<i>Journet et al. , 2006</i>).....	18
Figure 2.6 - a) Shape map using connected component analysis b) Background map of the image c) Combination of the two maps (<i>Ramel and Leriche, 2005</i>).....	19
Figure 2.7 – a) Corners detected with the Harris corner detector on two gray level images, b) Recovered correspondences in two words (<i>Rothfeder et al. , 2003</i>)	21
Figure 2.8 - Synthetically generated query word from prototype character images (<i>Konidaris et al. , 2007</i>).....	22

Figure 2.9 - a) Normalized query word in a pre-defined bounding box b) Upper and lower profiles c) extracted features, darker squares show higher pixel density d) Pixel density based features in the defined zones (<i>Konidaris et al. , 2007</i>)	23
Figure 2.10 - DTW matching on the MCC-DCT representations of the two contours (<i>Adamek et al. , 2007</i>)	24
Figure 2.11 - a) Original word image: slant/skew normalized and cleaned. The four features: b) Normalized vertical projection profile, c) Normalized lower profile, d) Normalized upper profile, e) Normalized number of ink/background transitions (<i>Rath and Manmatha, 2003</i>)	26
Figure 2.12 - a) Normalized character image, b) Features based on zones; darker squares indicate higher density of character pixels, c) Area formed by upper and lower profiles and the features extracted using them; darker areas indicate higher pixel density, d) Area formed by left and right profiles and features extracted using them; darker areas indicate higher pixel density (<i>Vamvakas et al. , 2008</i>)	28
Figure 2.13 - Use of a sliding window for extracting small fragments for which the features are defined (<i>Terasawa et al. , 2009</i>)	29
Figure 2.14 - LGH features calculated for small overlapping windows of a word (Rodriguez-Serrano and Perronnin, 2009)	30
Figure 2.15 - Guides and ZOIs for a template word image (<i>Leydier et al. , 2005</i>)	31
Figure 2.16 - A possible displacement of ZOIs (<i>Leydier et al. , 2005</i>)	31
Figure 3.1 - Different stages of Document Image Indexing in our implementation	36
Figure 3.2 - Binarization using NICK with $k = -0.1$ and $k = -0.2$ at window size 19	41
Figure 3.3 - T and TNB thresholds for different average gray levels m	42
Figure 3.4 - Comparison of binarization using Niblack and NICK methods for first three pages of a book	43
Figure 3.5 – Creation of a Scan Line Image: a) selection of a pixel line b) extending the pixel line in vertical direction c) ground truth of the synthetic image binarization	45
Figure 3.6 - Local thresholds obtained by different methods for the scan line image above	46
Figure 3.7 – Portions of the four sample document images among the 25 used in the OCR test ..	47
Figure 3.8 - F-scores of 43 different methods in the ICDAR 2009 competition – NICK (7c) stands at 16 th position	49
Figure 3.9 – Distance between two words must be greater than RLSA threshold	51
Figure 3.10 – Applying H-RLSA on portions of an image with $C = 9$	51
Figure 3.11 - H-RLSA on complete document images	52
Figure 3.12 - Areas and heights of all the bounding boxes of the 425 components in the document above	53

Figure 3.13 - Larger components are classified as graphics (red) and smaller ones as words (blue)	54
Figure 3.14 - Stages of word extraction a) Original image b) Binary image c) H-RLSA image d) Connected components (words)	55
Figure 3.15 – Title word “CHAPITRE” is not extracted correctly	55
Figure 3.16 - Concatenating the bounding boxes of two S-characters on top of each other	57
Figure 3.17 – Example of pass-1, a) Raw S-characters b) S-characters after Pass1	58
Figure 3.18 - Concatenating two S-characters into one S-character	59
Figure 3.19 - a) S-characters after pass1 b) S-characters after Pass 2	59
Figure 3.20 - Examples of bounding boxes overlapping in <i>italic</i> font	60
Figure 3.21 - Examples of T-characters broken into two S-characters	60
Figure 3.22 - Examples of S-characters comprising two T-characters	60
Figure 3.23 - Removal of punctuation marks	61
Figure 3.24 - Recall percentages for T-characters in each document image	62
Figure 3.25 – a) A document image and b) its binarized image, where character segmentation is not so good	63
Figure 3.26 - Normalized vertical projection profile of character image ‘p’	65
Figure 3.27 - Normalized Upper profile of S-character image ‘p’	66
Figure 3.28 - Normalized lower profile of S-character image ‘p’	66
Figure 3.29 - Ink Non-ink transitions for each column of an image	67
Figure 3.30 – Normalized Vertical histogram for the S-character image ‘p’	67
Figure 3.31 - Mean Row using three central rows of the image	68
Figure 3.32 - Mid row ink to non-ink transitional sequence for two S-character images of widths 20 and 23	68
Figure 3.33 - Automatic clustering of S-characters in an image by matching the feature sequences using DTW - Red blocks represent the end of a cluster	70
Figure 3.34 - Top and base lines drawn on the text lines	71
Figure 3.35 – S-characters of category 1 in red, 2 in green, 3 in blue and 4 in yellow	71
Figure 3.36 - Indexing time for different document images	73
Figure 4.1 - Different stages of word retrieval system	76
Figure 4.2 - Selection of a query word image by clicking on the document image	77
Figure 4.3 – Variations of a character in different books	79
Figure 4.4 - Percentage of words filtered out by the length-ratio filter	81
Figure 4.5 – The overall system - highlighted S-characters (in gray) of two words are matched using DTW (matrix D) while two words are compared using different techniques	82

Figure 4.6 - Comparison of the features of S-character 'p' written two in different styles	84
Figure 4.7 – Vertical projection profiles of ‘p’ and ‘p’, aligned using linear scaling and Dynamic Time Warping. DTW ensures that only corresponding locations will be compared	85
Figure 4.8 - Local continuity constraint, showing valid neighborhood relationships in a warping path.....	86
Figure 4.9 - The entry D(4,5) shows the distance between two sequences X and Y	87
Figure 4.10 - Individual distances between S-characters using only a) Vertical projection b) Upper profile c) Lower profile d) Ink/non-ink e) Histogram d) Mid row transitional sequence	90
Figure 4.11 - Distance histogram using all features for 85,718 comparisons.....	91
Figure 4.12 - Individual normalized distances between S-characters using only a) Vertical projection b) Upper profile c) Lower profile d) Ink/non-ink e) Histogram d) Mid row transitional sequence	93
Figure 4.13 - Normalized distance histogram using all features.....	94
Figure 4.14 - RPC between S-characters of two words of length 12 with $X = 1$	95
Figure 4.15 – S-character comparisons for a) query word of length 3 b) query word of length 15. S-character 1 & 15 (corner S-characters) are compared with three S-characters, S-character 2 & 14 are compared with four S-characters, while others are compared with five S-characters in the test word.....	96
Figure 4.16 - Example of a false positive for a small word with a non-zero ‘X’.....	97
Figure 4.17 -Example of an unwanted match for a query word 'nette'	98
Figure 4.18 - Multistage system - features of two S-characters X and Y are matched using elastic DTW (D) while the two words are matched using Edit matrix (W).....	99
Figure 4.19 - Each entry of matrix W is dynamically calculated using a DTW matrix (D).....	99
Figure 4.20 - The entry $W(4, 4)$ is the cost of aligning the two words of length 4.....	102
Figure 4.21 - Words not matched due to segmentation problems.....	103
Figure 4.22 - Character segmentation errors (Split as in first five examples and merge as in last example) causing problems in matching stage.....	104
Figure 4.23 - One S-character of query word matched against two S-characters of test word....	105
Figure 4.24 - Two S-characters of query word are matched against one test S-character	106
Figure 4.25 - Calculating Edit Matrix W for two similar words of lengths 4 and 3	108
Figure 4.26 - Calculating Edit Matrix W for two similar words of lengths 6 and 7	109
Figure 4.27 - Area of interest for the matching of two words in Edit distance.....	110
Figure 4.28 - Different operations for linear displacement matching.....	111

Figure 4.29 - First iteration of the algorithm for matching the S-characters of two words. We can see that Merge-Q, the operation of matching S-characters 'F', 'I' of the query word with S-character 'FI' of the test word, yields the minimum cost.....	113
Figure 4.30 - Time taken per 100 words searched, by linear displacement matching and Merge-Split Edit distance, for query words of different lengths.....	114
Figure 4.31 – Example of relevant word and false positive word for given queries.....	117
Figure 4.32 - Precision vs Recall at different thresholds for the six features.....	118
Figure 4.33 - F-Scores for different features at different thresholds.....	118
Figure 4.34 - Recall rates for different methods on our data set.....	122
Figure 4.35 - OCR result for a page where green zones highlight the text areas while red zones highlight the graphics. OCR problems : word 'FIG' has been misread as 'Fis', while some words are taken as part of graphics and not as text.....	123
Figure 4.36 - Label text completely misread by the OCR.....	124
Figure 4.37 - Precision percentages for different methods.....	125
Figure 4.38 - F-scores of different methods.....	125
Figure 4.39 – Merge-Split Edit distance - Precision vs Recall at 8 different thresholds between 0.4 and 1.1.....	126
Figure 4.40 – Merge-Split Edit distance - F-score at different thresholds.....	126
Figure 4.41 - R-scores showing the relevant word matches against total extra words detected ..	127
Figure 4.42 - Portion of a 16th century document image - difficult to read even from the naked eye due to document quality and also because of the old font styles.....	128
Figure 4.43 - Extraction of text areas using ABBYY. Some text areas are taken as graphics (green parts are segmented as text areas while red parts highlight the graphics).....	129
Figure 4.44 - Another example of text extraction and recognition by ABBYY software.....	129
Figure 4.45 - Text extraction using the proposed method. All text areas are detected.....	130
Figure 4.46 - Segmentation of words (in red bounding boxes) and S-characters (in blue bounding boxes).....	130
Figure 4.47 - Word segmentation problems: Some words are merged together as the spacing between them is very low.....	131
Figure 5.1 - Manual Figure/caption indexing on BIUM web base.....	135
Figure 5.2 – a, b) The nearest neighbors of each black square belong to the perceived a) row or b) column. c) Filled in black, the NNH bounding boxes in horizontal text lines, d) and the NNV in vertical text lines.....	136
Figure 5.3 - Close views of a page: a) After grouping NNV and NNH. b) Final detection.....	137
Figure 5.4 - Different Caption Labels used in Medic@ books.....	138

Figure 5.5 - a) Figure and caption candidates, true caption lines are filled in black. b) Three line candidates, after word spotting of “Fig” candidates 1 and 2 are eliminated. c - d) Candidates do not include the first caption line retrieved by word spotting	139
Figure 5.6 - Character segmentation issues in contemporary document images	141
Figure 5.7 - Examples of words broken down in two lines	143
Figure 5.8 - Sample page of the Quran with the text lines slanted	144
Figure 5.9 - Examples of S-characters for different words	144
Figure 5.10 - Sequence of S-characters in a word	145

List of Tables

Table 3.1 - : Portion of output images obtained by using different methods at windows size 19, a) Niblack b) Sauvola c) Wolf d) Feng e) NICK.....	44
Table 3.2 - Recognition rates achieved by the ABBYY OCR	48
Table 3.3 - Result summary of character segmentation process	64
Table 4.1 – Length-ratio filter for different word lengths	81
Table 4.2 - Pseudo code for the DTW algorithm	87
Table 4.3 - Overall distance values using individual features as well as all the features for 85,718 different comparisons	92
Table 4.4 - RPC pseudocode	98
Table 4.5 - Pseudo code for the Edit distance algorithm.....	101
Table 4.6 - Pseudo code for the Merge-Split Edit distance algorithm.....	106
Table 4.7 - Pseudo code for linear displacement matching algorithm	112
Table 4.8 - Different instances of query words from 12 different books	119
Table 4.9 – Experimental analysis of the four proposed string comparison methods.....	120
Table 4.10 - Results using word feature based method of <i>Rath and Manmatha</i> , and ABBYY OCR software	121
Table 4.11 - Performance evaluation of the different methods and the professional OCR software based on their recognition rates.....	132
Table 5.1 - Results for figure caption retrieval.....	140
Table 5.2 - Summary of word retrieval results for different query words.....	142
Table 5.3 - Results of word spotting on Arabic text – Top 6 matches alongwith their distances (the distance values of false positives and extra words are highlighted).....	145



List of Abbreviations and Terminologies

A list of commonly used abbreviations and the different terminologies that frequently are used throughout the thesis:

BCC: Basic Connected Component

CC: Connected Component

DTW: Dynamic Time Warping

OCR: Optical Character Recognition

RLSA: Run Length Smoothing Algorithm

ROI: Region of Interest

RPC: Relative Position Correspondence

Document image: A page of a book

S-character: Segmented character

T-character: True/Actual character

Chapter 1

Introduction

The importance of digital libraries for information retrieval cannot be denied. The ancient historical books contain invaluable knowledge but it is very time consuming for the researchers to search the required information in these books. Our work in this domain aims *to facilitate this information search by spotting different instances of a given query word in the text*. With this ability to search in ancient historical documents, digital libraries will further enhance their importance. Research in word spotting on the Latin alphabet has received considerable attention over the last few years. A wide variety of techniques have been proposed in literature. The field however, remains inviting and challenging, especially if the document base comprises low quality images, which in fact will be the subject of our research.

1.1 Motivation behind the research

Libraries and museums all across the world contain extensive collections of ancient historical documents printed or hand written in their native languages. Typically, only a small group of people are allowed access to such collections, because the preservation of the material is of great concern. In recent years, libraries have begun to digitize historical document corpora that are of interest to a wide range of people, with the goal of preserving the content and making the documents available via electronic media.

Historical collections are of interest to a number of people, like historians, students and scholars who need to study the historical originals. Unfortunately, digitization alone is not enough to render historical document collections useful for such research purposes. Having the information available in an electronic image format makes it possible to share it with many people across large distances via the Internet, Digital Versatile Discs (DVDs) or other digital media. However, the size of a collection is often substantial and the content is generally unstructured, which makes it hard to quickly find particular documents or passages of interest.

Various solutions for this problem that rely entirely on human labor could be possible. A simple way of structuring a collection of historical documents is by ordering them chronologically and manually annotating each of the pages to construct an index from this annotation. A very high level of detail in content annotation may be achieved with transcription. It allows full-text search using a traditional text search engine. But the cost of transcription increases manifolds with the

size of the database. Using an automatic recognition approach seems like an obvious solution. There are professional OCR software designed from different languages especially Latin alphabets which give excellent results on newly scanned images of contemporary documents in good quality. But when used with ancient documents suffering from degradation due to faded ink, stained paper, dust and other factors, the recognition results drop appreciably.

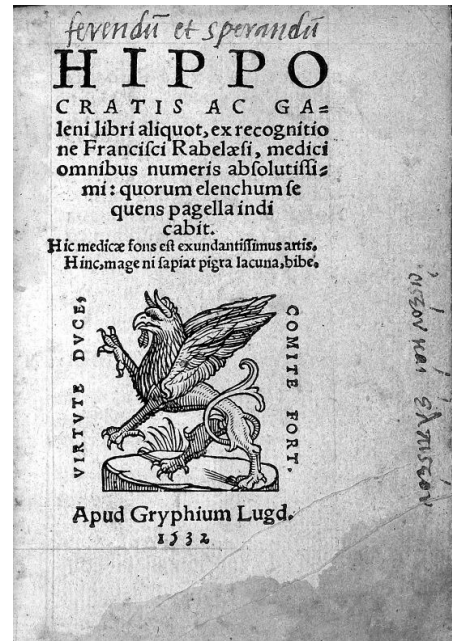
Word spotting is a relatively new alternative for information retrieval in ancient document images. Word spotting means matching a query word image with all the words in the documents and retrieve all the document images or passages containing similar words to the given query. Research has been going on in this field for some time now and already different methods have been proposed for efficient word spotting (which are discussed in detail in the next chapter) but there is always some room for improvement. Moreover, mostly the word spotting methods require a word image as query which is rather difficult for the users as giving an ASCII query is always more feasible and easy. Main motivation and aim for us behind this research *was to propose an efficient information retrieval system that could work efficiently with good precision rates for huge volumes of printed document images with word image or ASCII text queries to search information in text areas which we call as regions of interest (ROI).*

1.2 Digital Libraries and Historical Documents

There are libraries all across the world that are working on the preservation and digitization of ancient historical books and document images. The promises and challenges of digital libraries have been discussed in detail in [Jameson2004]. The digitization process of the document images, which are made available for research purposes by these digital libraries, is not simple scanning. It is because the crude scanned document images cannot be used directly for information retrieval. Some post processing (like cropping, rotation, etc.) is done for each document to get them in shape for use in information retrieval. An example of this processing is shown in Figure 1.1. This sort of trivial post processing is usually done by the digital libraries before making the digitized document images available for public reading and research purposes. In our work, we will be using digitized document images of the digital medic@ library of Paris [BIUM].



(a)



(b)

Figure 1.1 - Digitization process of a document image: from a) crude scan image to b) the displayed gray image formed using commercial tools

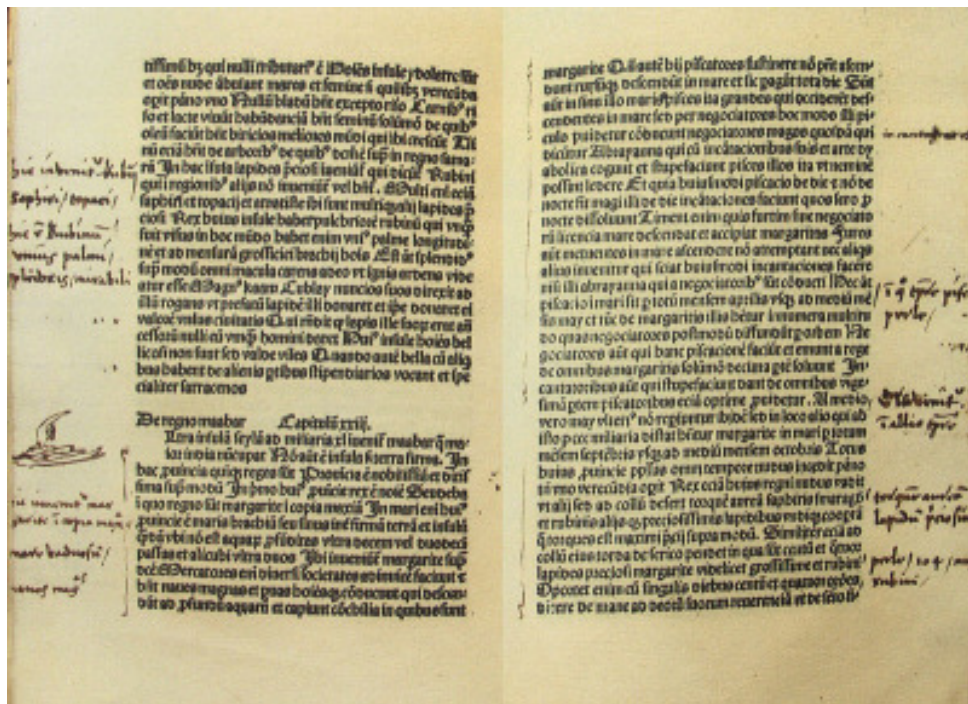


Figure 1.2 - Marco Polo's "Le Livre des Merveilles", Latin edition of 14th century¹

¹ http://en.wikipedia.org/wiki/Marco_Polo

If we look at the history of printed documents, we learn that in the beginning, the fonts and the layouts of the pages were very close to the handwritten books [Journet *et al.* 2005, Journet *et al.* 2006]. These printed documents were handcrafted and the technical constraints of the past introduced irregularities in book production like variations in spacing, margins, random alignments, etc. (see Figure 1.2). These documents contain many defects due to the manufacturing process, the conditions in which these books were conserved and also because of the absence of printing rules which came into practice much later on.

Johannes Gensfleisch Gutenberg was a German goldsmith and printer who is credited with being the first European to use movable type printing, in around 1439, and the global inventor of the mechanical printing press. His major work, the Gutenberg Bible (Figure 1.3) has been acclaimed for its high aesthetic and technical quality.



Figure 1.3 - Picture of the Gutenberg Bible owned by the US Library of Congress²

With time, more regularity came into printing as the printing rules were penned down. **Joseph Moxon**, an English printer of 17th century and hydrographer of the King Charles II, is credited globally for producing the first ever written manual on printing. His book *Mechanick Exercises* in 1683 provides detailed descriptions of contemporary printing methods that have proved to be useful for bibliographers³. Another name which is equally important in this respect is the frenchman **Martin Dominique Fertel**. His Book *La Science pratique de l'imprimerie* is acknowledged by scholars to have been as important to French printers as its predecessor was to English printers [Pankow2005].

² http://en.wikipedia.org/wiki/Johannes_Gutenberg

³ http://en.wikipedia.org/wiki/Joseph_Moxon

Some examples of 18th and 19th century historical printed documents of different countries are shown in Figure 1.4 and Figure 1.5. We can see the variations in print styles, in different images. These variations in printing styles along with quality issues of ancient documents pose major problems for document analysis and information retrieval.

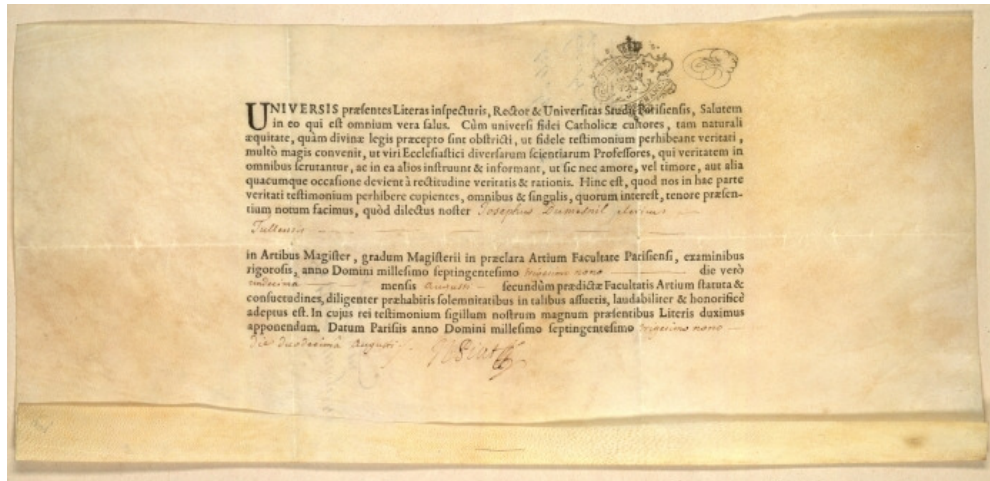
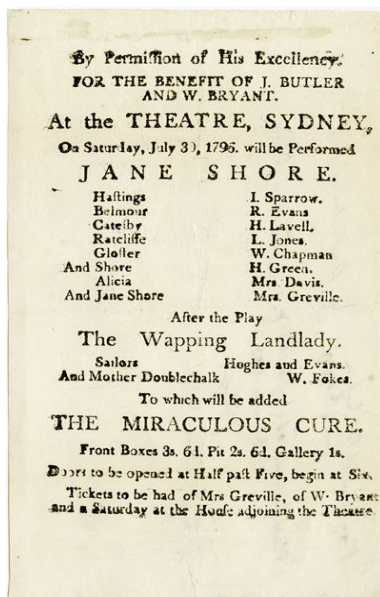
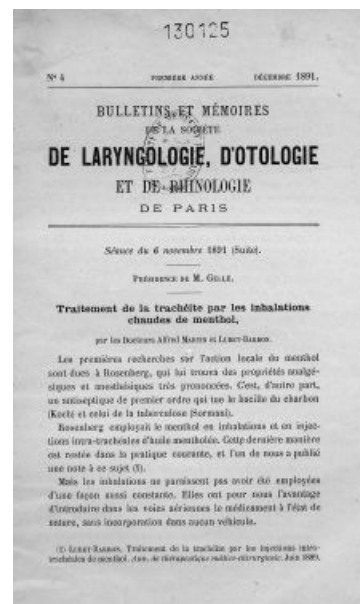


Figure 1.4 - A degree issued by the University of Paris, 1739⁴.



(a)



(b)

Figure 1.5 - a) The oldest known document printed in Australia, a theatre playbill from the former British colony dating back to 1796. b) A document image from a 19th century book in medic@digital library [BIUM]

⁴ <http://pergametai.mch.mii.lt/DokPranc/indexen.en.htm>

Most of our work focuses on the 18th and 19th century document images in which we analyze different problems these document images pose to information retrieval methods. We will see it briefly in section 1.3.

1.2.1 Document Preservation and Retrieval projects

Many libraries across the globe are working to preserve the historical documents and books. Different international collaborations also exist for this purpose. One such collaboration is *International Dunhuang project* (IDP) [IDP]. IDP is a ground-breaking international collaboration to make information and images of all manuscripts, paintings and textiles from Dunhuang and archaeological sites of the Eastern Silk Road freely available on the Internet and to encourage their use through educational and research programs. IDP partner institutions which provide data and act as hosts to the multilingual web site and database include Bibliothèque Nationale de France and the British Library in London, along with five other partners from different parts of the world. British library has been at the fore front in preservation of the historical books. In fact it is believed that the oldest complete survival book *Diamond Sutra* (see example in Figure 1.6) printed in Buddhist text in 868 AD⁵, is also preserved there.

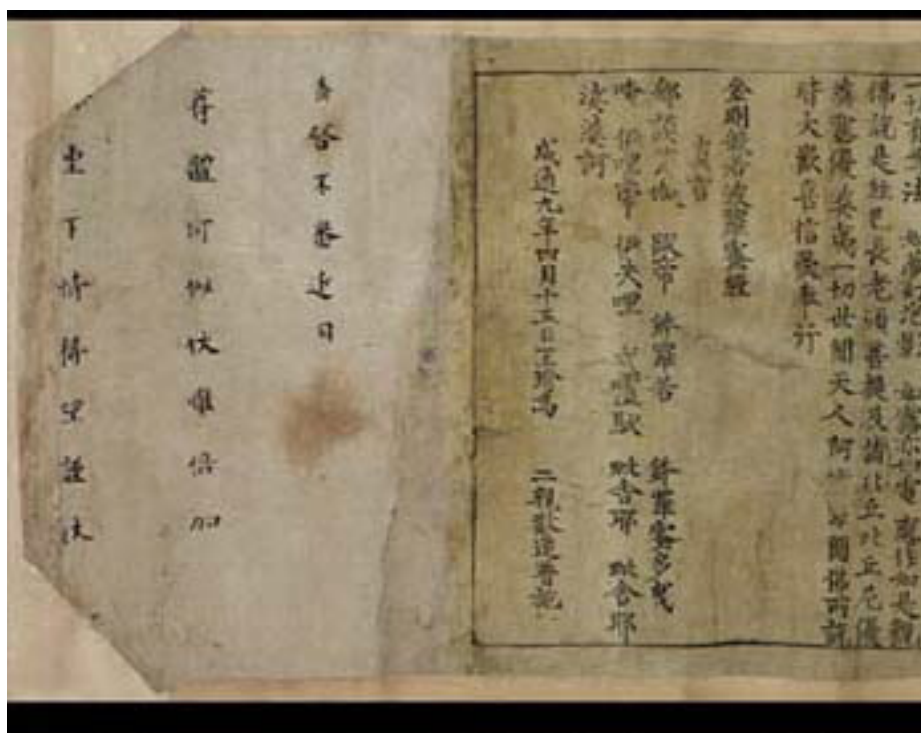


Figure 1.6 - The Chinese Diamond Sutra, the oldest known dated printed book in the world

⁵ http://en.wikipedia.org/wiki/Diamond_Sutra

Apart from IDP, there are other different national projects as well that focus on the preservation and indexing of document images. These include *Madonne* [Ogier and Tombre2006] which is already finished and another bigger project *NAVIDOMASS* (NAVigation in Document MASSes) [NAVIDOMASS] which is currently underway in colloration with six major national laboratories in this field. The general purpose of NAVIDOMASS is the construction of a framework to digitally preserve the heritage document collections in libraries, museums and public archives and provide universal access to them.

1.2.2 Online access to Digital Documents

Presently, online information access services in a large number of digitized books are provided by many international organizations, companies and libraries across the world. The biggest such project is undertaken by Google, called **Google Book Search**⁶. It is a service that searches the full text of books that Google scans, converts to text using OCR, and stores in its digital database⁷. Figure 1.7 shows the screen shot of this online access service.

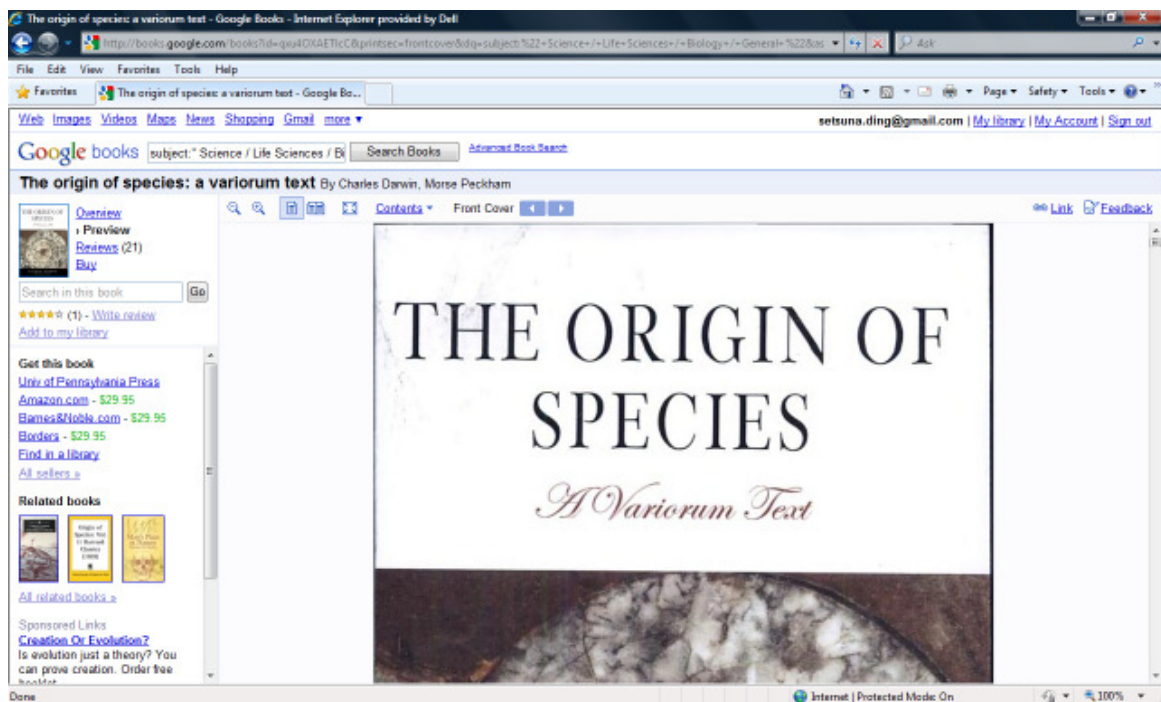


Figure 1.7 - Screen shot of the Google book search service where a book is opened and its text can be searched using the text search option

⁶ <http://books.google.com>

⁷ http://en.wikipedia.org/wiki/Google_Book_Search

Other online document resource access facilities include the **Universal Digital Library**⁸ which is provided by Carnegie Mellon University in USA and **Europeana**⁹ which is co-funded by the European Union. . Different national libraries are also working hard to provide online access of their document resources. These include **Gallica**¹⁰ which is the digital library of the bibliothèque nationale de France, **BVH**¹¹ (Bibliothèques Virtuelles Humanistes) of the Centre d'Études Supérieures de la Renaissance Tours and **Digital library Medic@** of the bibliothèque interuniversitaire de médecine Paris [BIUM]. Our present work is partly in collaboration with BIUM which provides online access to a large number of historical medical books of different era but the text search service is not provided for these documents.

1.3 Context of our Work

A *document retrieval system* holds great promise for providing access to historical printed documents containing immense knowledge for a large set of audience. Given a word query, the document retrieval system would find all document images containing relevant “answers” to the query, which saves the user the tedious work of browsing or reading through an entire collection of documents while looking for a particular document image or ROI. This work provides a thorough examination of several retrieval techniques for historical document images that will allow queries in the form of a word image or ASCII text. It is particularly appealing that the queries can be textual, a fact that makes this system very practical.

1.3.1 Data Sets

A data set is very important for proper validation of some method. Unfortunately, there is no standard benchmark data set available for historical printed documents, on which different researchers could validate their algorithms. Every work has been done in a particular context and so a particular data set has been used each time. For us, we have used document resources of the Digital library Medic@, Paris [BIUM] for testing and validation of different stages. We built a data set with historical document images gathered from the digital medical library of Paris “Bibliothèque Interuniversitaire de Médecine, Paris”. The total experimental data set consists of 12 books of 19th century. For experimental validation of different individual stages as well as for complete system testing and its applications (Chapter 5), we made three subsets from this document base. We have named the data sets as data set A, data set B and data set C. Apart from

⁸ <http://www.ulib.org/index.html>

⁹ <http://www.europeana.eu/portal/>

¹⁰ <http://gallica.bnf.fr>

¹¹ <http://www.bvh.univ-tours.fr/>

that, the approach has later on been applied to ancient documents of 15th and 16th century to analyze different challenges that are posed to the system by these ancient documents. The sample images of the data sets are shown in Annex A.

a. Data Set A

In data set A, we have document images taken from 5 different books. The image sizes vary from 1256 x 1939 to 1708 x 2721 for different books. From each book, four different pages are selected making the size of the data set to 20 document images. A total of 6,632 words exist in these pages. For query, five different words are selected from the images of each book thus giving a total of 25 different query words having 175 instances in total. Instances of a query word are the different occurrences of the same word in similar or different sizes and styles. For example, the instances of a word “query” could be “query”, “*query*”, “**query**”, “Query”, etc. (The word “QUERY” is not counted as the instance of word “query”).

This data set is used to test individual stages of the system, for computational comparisons and also for some performance measures.

b. Data Set B

In data set B, we selected four images from each of the 12 books thus giving a total of 48 document images. The image sizes vary from 1256 x 1939 to 1708 x 2721 for different books. Total number of words in these pages is 17,010. For query, five different words of different lengths are selected from the pages of each book thus giving a total of 60 ($5 \times 12 = 60$) different words having 435 instances in total. The query words have been selected on the basis of their relevance to the book (i.e. key words).

This data set is used to test word and character segmentation stages as well as for the detailed comparisons of different word matching methods proposed later on.

c. Data Set C

In data set C, we have three books having a total of more than 500 document pages. This data set is used to test the figure/caption retrieval application discussed in next chapter.

The same documents but in *web-efficient* resolution (images with relatively lower resolution than the original ones), are available on the web and are open to access to other researchers. A couple of examples are shown in Figure 1.8 and Figure 1.9. We can see that the document resolution is not good and when zoomed in, the digitization problems in the words and the characters become clearer and strong, which make it extremely hard to segment the characters properly. These

digitization effects are not as strong in the original document images though, so a possible character extraction may be thought of in the higher resolution images. Example images are shown in Annex A.

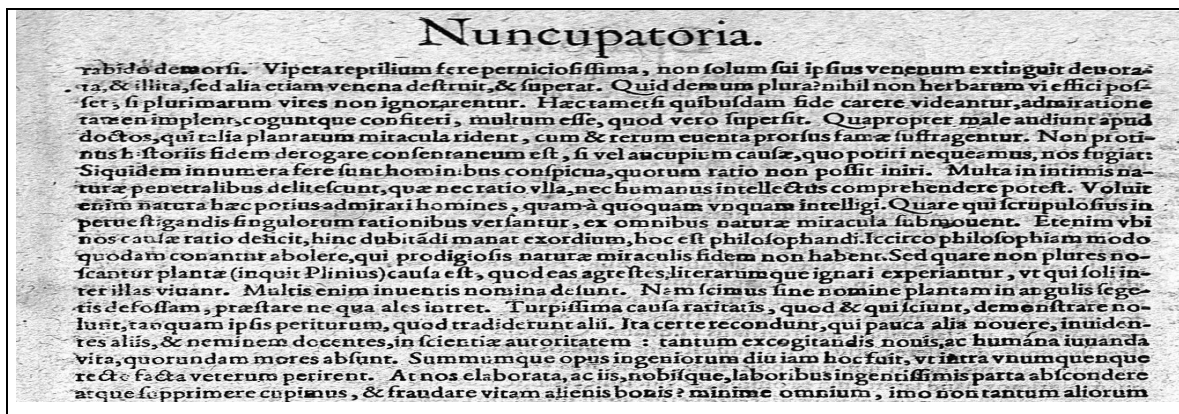


Figure 1.8 – An image from a 16th century book by Bassaei available on BIUM web¹²

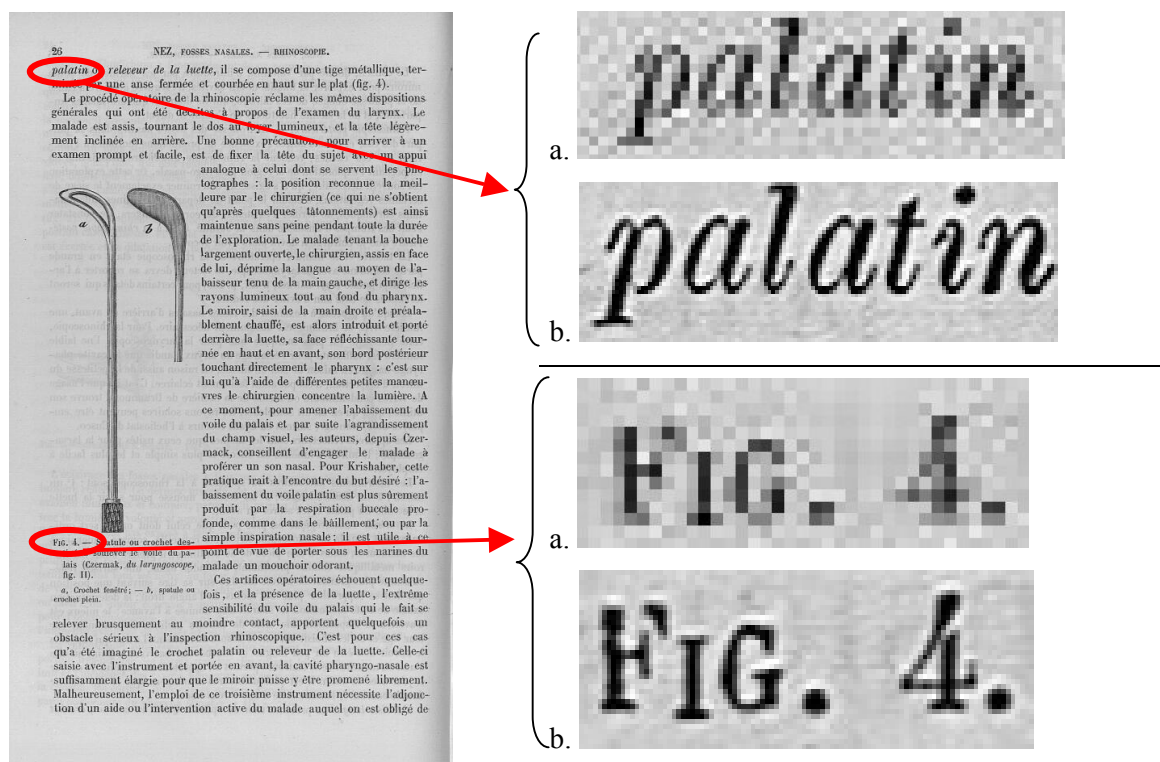


Figure 1.9 - A BIUM document image of 19th century: a) in web resolution images the digitization effects are more visible as compared to the b) original images

¹² <http://web2.bium.univ-paris5.fr/livanc/index.las?cote=00825&do=chapitre>

1.3.2 Global Overview of our subject

Our main objective is to propose a robust document image retrieval system based on word spotting. The task of retrieval is to retrieve document images, that is, the pages of books in the collection at hand according to their relevance to the given query. All document images containing the given query instance are retrieved and are presented to the user in a chronological order. Retrieval is not limited to document images; it can often be easily extended to other retrieval units, such as paragraphs or lines. For example, lines are taken as unit to allow figure captions to be retrieved from the document images to create figure's index. (It is explained in detail in Chapter 5). As the title of this work indicates, we are retrieving images of text, so the user will always be presented with an image as the response to a query, be it of a line, a ROI or a page.

In our implementation, we have divided the overall retrieval system into two distinct parts: the document indexing part and the matching part for retrieval of relevant document images.

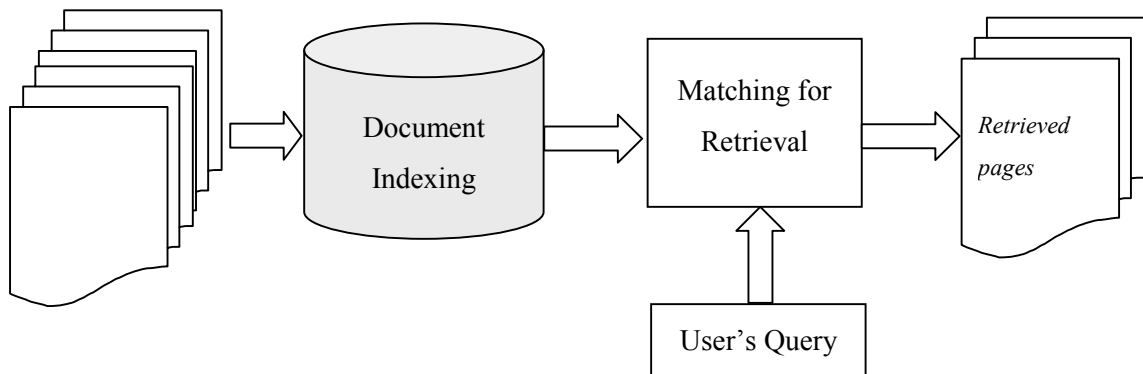


Figure 1.10 – Global overview of the document image retrieval system

For indexing, we have divided the whole indexing process into four sub-steps as depicted in Figure 1.11.

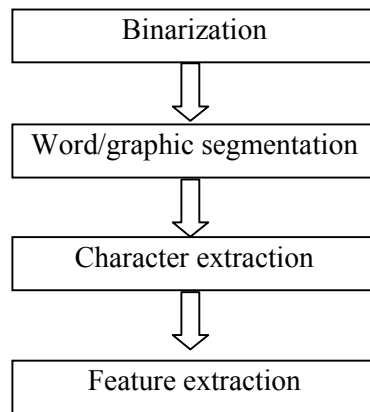


Figure 1.11 - Four stages of document image processing for indexing

The first step is the preprocessing stage involving an optimized document *binarization* algorithm to crisply distinguish the foreground from the background. Second step involves the *word/graphic segmentation*. As we are working on printed documents, where a high level of granularity is possible, it prompted us to work at character image level thus allowing more flexibility in the matching phase. For that, the third indexing step revolves around the connected component analysis of words for the extraction of characters. We call these extracted connected components as *segmented characters* or ‘S-characters’ while the actual (perfectly segmented alphabetical) characters are termed as *true characters* or ‘T-characters’. The S-characters may or may not be same as T-characters. To get better S-characters (which relate closely to T-characters), S-characters of a word are post-processed using a 3-step process to get better S-characters, which in most cases are the true characters. But still some character segmentation errors remain (see Figure 1.12) which are handled in the matching part. In the last step, *multidimensional features* are defined for each S-character of a word. These character features are used for matching purposes. The feature values along with some other document image relevant information are stored in an index file. For each document image, an index file is created.



Figure 1.12 – Examples of characters with segmentation inconsistencies; some T-characters are divided into two S-characters (first four examples) or two T-characters are merged into one S-character (last example)

For word spotting part of the system, we have proposed a two level *character-feature* based retrieval system whose main aim is to retrieve relevant document images while catering for the character segmentation errors that are left in the indexing process (see Figure 1.12). It makes sure that only a 100% accurate character segmentation is **not** necessary to achieve good results. In our two-level matching system, two S-characters are matched using elastic Dynamic Time Warping (DTW) while the two words are matched using different string comparisons algorithms. We also introduce an original Merge-Split Edit distance at this level that takes into account the character segmentation inconsistencies in document images for correctly matching the two words.

1.3.3 Major Contributions

Major contribution of our work lies in the whole idea of representing the words based on S-character features. It enables us to have multi-level matching, at word level and at character level, which opens up different options for new and efficient implementations of word matching algorithms, aiding to overcome the existing problems in the field. Our major contributions lie in both the two phases of our system i.e. indexing and retrieval. In indexing phase, our contributions lie in binarization algorithm (NICK algorithm), character segmentation and feature extraction. Though we have used some features which are frequently used in literature [Rath and Manmatha2007], [Kolcz *et al.* 2000], [Konidaris *et al.* 2008], we have introduced some of our own as well (like mid row transitional vector, etc.) to improve character representation. In retrieval phase, we have introduced a concept of multistage retrieval system based on dynamic matching at two levels – word and character. To take into account the bad quality of the document images that causes character segmentation problems, a new Merge-Split Edit distance has also been proposed to match two words irrespective of the fact that their characters have been segmented correctly or not.

Apart from that, we have discussed different applications of our system such as an automatic figure caption retrieval system by fusion of spatial and perceptual information of the document image.

An easy to use graphical user interface (GUI) has also been developed which facilitates us in easily searching the required information by giving ASCII or word image queries and can be used with any kind and number of document base for the purpose of document image indexing and retrieval.

1.3.4 Organization of the thesis

The thesis is organized in a sequential way starting with the description of state of the art methods in document retrieval in *chapter 2*. Different methods are categorized and discussed in detail to give an overall idea of what is going on in the field right now.

In *chapter 3*, we discuss in detail the document indexing part. It starts with document image binarization using the proposed NICK algorithm. It is followed by the description of text/graphic segmentation using run length smoothing algorithm. Once the words from the text are extracted, we discuss in detail the process of extraction of S-characters for each word using connected component analysis followed by a 3 stage post processing. The description of the features selected to represent the S-character images is given afterwards. Lastly we discuss the indexing of the features in data files and the issues pertaining to that. These indexed files are later on used for matching purposes.

In *chapter 4*, we discuss the document retrieval process. It starts with the explanation of the different ways to input a query word to the system are explained. The multi-stage matching process is then discussed in detail. It includes length-ratio filter, character matching and word matching stages. Elastic Dynamic Time Warping is used for matching the features of the two characters. For matching two words, different methods are discussed along with their pros and cons. Experimental results are then elaborated in the last section showing the results of word retrieval methods and also performance evaluation of different feature sequences representing the S-character image.

In *chapter 5*, different multi-context applications of the retrieval process are shown. First the automatic figure/caption retrieval system is discussed in detail. Another application pertaining to contemporary document analysis and retrieval is shown afterwards.

In *chapter 6* some perspectives of the work are discussed along with conclusion of the thesis.

A couple of annexes are given at the end. *Annex A* shows some sample images used in our experimentation. In *Annex B* some screen shots of the graphical user interface of the document retrieval system developed are shown.

Bibliography is given after the Annexes. A list of author's publications is the last part of the thesis.

Chapter 2

State of the art

Lot of work has been done and plenty is in progress in the domain of information retrieval in historical documents. There are a lot of issues and problems related to ancient printed documents which are discussed in detail in [Baird2004], [Antonacopoulos *et al.* 2004] and [Zhang and Tan2001]. These include the physical issues such as quality of the documents, the marks and strains of liquids, inks and dust, etc.; and the semantic issues such as foreground entity labeling. Also when document images are scanned from thick bound volumes of historical books, there always occurs perspective distortion which causes shadow at the book spine area and also the warping of the words in the shadow [Zhang and Tan2001]. Similar sort of spurious components are observed in old manuscripts as discussed by [Rath2005] which are depicted in Figure 2.1. All these problems provide a big challenge for researchers working in the domain to achieve better results in information retrieval.

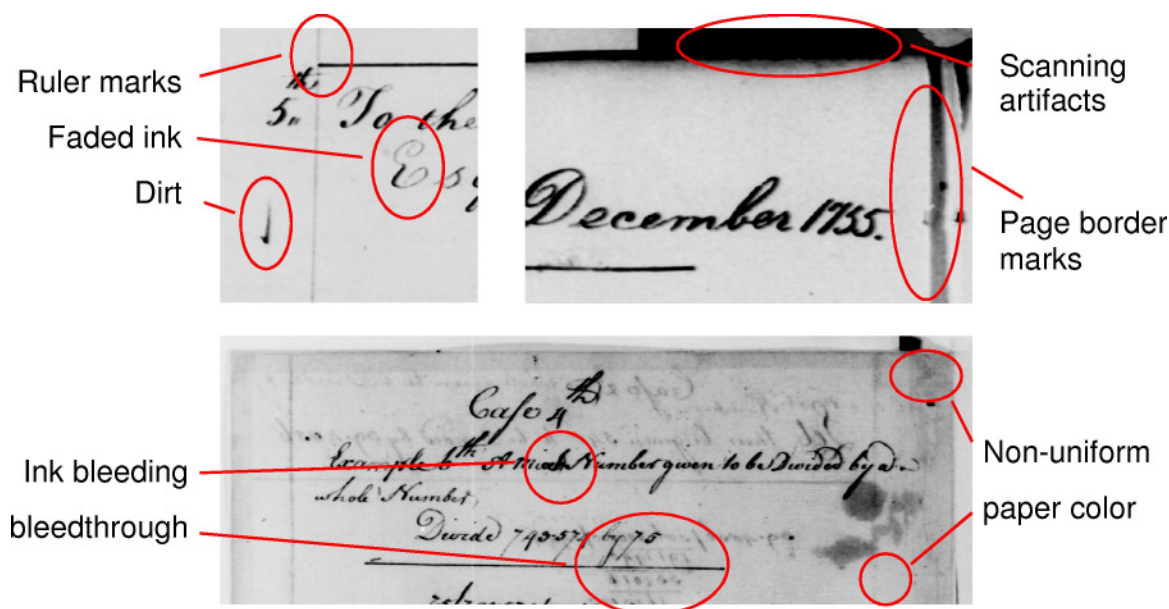


Figure 2.1 - Different artifacts occurring in historical document images (Rath, 2005)

Over the years, various methods have been proposed to deal with historical document images for their segmentation (including image binarization, text/graphic segmentation, text words

extraction, etc.) as well as for information retrieval. In this chapter, we will briefly discuss the state of the art methods for document image segmentation and information spotting in historical document images. Section 2.1 analyzes the different categories of document segmentation methods and section 2.2 focuses on the different information retrieval methods in historical documents.

2.1 Document Image Segmentation

Text/graphic segmentation and a fine extraction of words and characters in ancient document images are very useful for applications such as information retrieval using word spotting or optical character recognition. The degraded quality of these ancient documents poses different problems such as characters broken into multiple components, text of the verso appearing on the recto, etc., thus making the crisp extraction of the words and characters very difficult [Antonacopoulos *et al.* 2004], [Baird2004]. Lot of work has been done earlier to cater for these problems and provide a better way for document image segmentation for different applications. Traditionally, page segmentation methods are divided in three main groups: top-down, bottom-up and hybrid approaches [Okun *et al.* 1999], [Duong *et al.* 2001], [Journet *et al.* 2005], [Shi and Govindaraju2005], [Journet *et al.* 2006]. We discuss them briefly here.

2.1.1 Top-down Techniques

In top-down techniques, document images are recursively divided from entire image to smaller regions. These techniques are often fast, but the efficiency depends on a prior knowledge about the class of documents to be processed. Developments have been produced in early times. The most well known methods are projection methods [Antonacopoulos and Karatzas2005] (with many variations like rectangulation, white streams method [Pavladis and Zhou1991], etc.), histogram analysis, form definition languages [Tang *et al.* 1993], rule based systems [Lee *et al.* 2000], or space transforms [Jain1989] (Fourier transform, Hough transform, etc.). Figure 2.2 shows the segmentation of characters using the projection method of [Antonacopoulos and Karatzas2005]

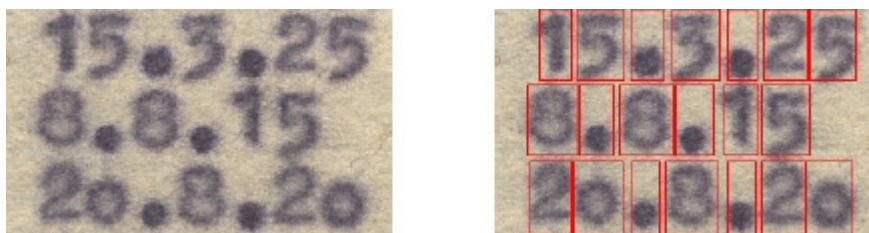


Figure 2.2 - Extraction of characters using a combination of vertical and horizontal projection profiles (Antonacopoulos and Karatzas, 2005)

Though these top down methods generally perform well, they have a major drawback which is the need to have a prior knowledge about the document class and type (number of columns, width of margins, etc.) for them to be effective.

2.1.2 Bottom-up Techniques

Bottom-up methods start with the thinnest elements (pixels), merging them recursively in connected components or regions, and then in larger structures. They are more flexible but may suffer from accumulation of errors. They make use of methods like connected component analysis [Mitchell and Yan2001], [Faure and Vincent2009], region-growing methods, run-length smoothing [Wong *et al.* 1982], neural networks [Tan and Zhang2001] and active contours [Bukhari *et al.* 2008]. Figure 2.3 shows the result of extraction of figures and the caption line candidates for the extracted figures using the rule-based connected component position and area analysis approach in [Faure and Vincent2009].

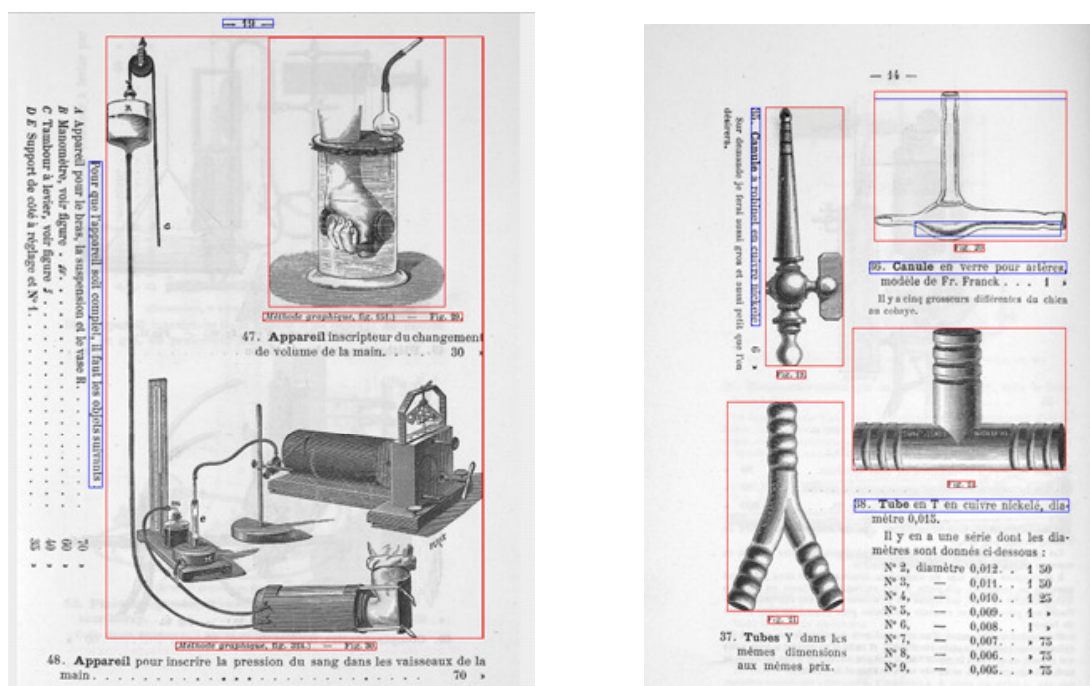


Figure 2.3 - Segmentation of figures and figure caption candidates by connected component analysis (Faure and Vincent, 2009)

Curled and curved text lines are segmented by [Bukhari *et al.* 2008] using an active contour method. Figure 2.4 shows an example of the segmentation of curled text lines in [Bukhari *et al.* 2008].

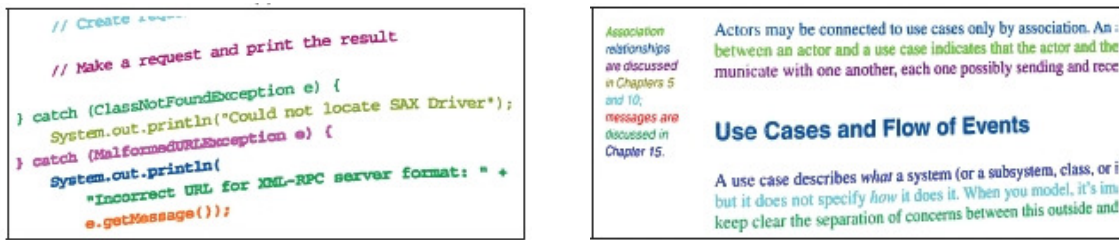


Figure 2.4 - Curled text lines segmentation using Active contours (Bukhari et al. , 2008)

The advantage of bottom-up methods is that they are very flexible [Shi and Govindaraju2005]. On the other hand, although these methods are efficient for modern and contemporary books, they are not so efficient for ancient medieval books because of the specificity of these ancient documents such as non-constant spacing between characters, words and images, etc. Another thing is that these methods make use of a lot of parameters that need to be adjusted precisely for good results.

2.1.3 Hybrid Techniques

Many other methods that do not fit into either of these categories are therefore called hybrid methods. Among these are the texture-based methods [Randen and Husøy1994], [Journet *et al.* 2006] (Figure 2.5). They include Gabor filter method [Ar and Karsligil2007], wavelet and fractal analysis, auto correlation function [Journet *et al.* 2005], etc.

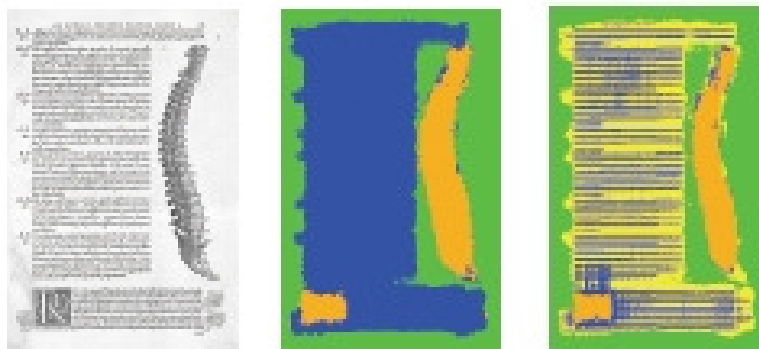


Figure 2.5 - Segmentation in 3 and 4 classes by textural approach (Journet et al. , 2006)

There are also some hybrid methods that combine and make use of both bottom-up and top-down approaches. For example, connected component analysis for shape information and block separation for background block map have been used in [Ramel and Leriche2005] in a hybrid segmentation approach (Figure 2.6). Classification of these blocks is achieved according to the scenarios defined by the user.

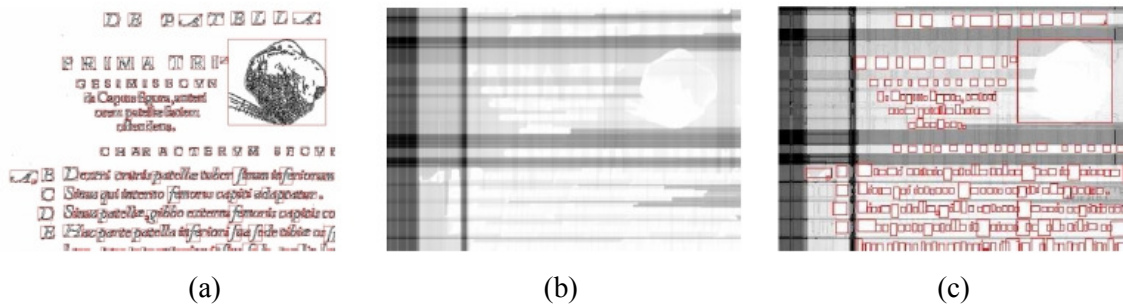


Figure 2.6 - a) Shape map using connected component analysis b) Background map of the image c) Combination of the two maps (Ramel and Leriche, 2005)

Hybrid methods do not need prior knowledge about the page structure and work very well for major text/graphic segmentation in historical and contemporary pages (news papers, journals, etc.), but not for a very fine level segmentation in historical books (for example to segment words and their individual characters, etc.).

Overall, we can see that every type of methods has some pros and cons and the final choice of a method depends solely on the type of application for which the segmentation is required. If the requirement is limited to text/graphic segmentation on the images for which we don't have any prior knowledge, then hybrid methods are more efficient for that. If, however, we have the prior knowledge about the class of the documents that we are working on, then top-down methods are more efficient and precise while if we want to segment on a smaller/finer level (e.g. characters, etc.), then bottom-up methods seem to do the job better. In the present work, a multi-step bottom-up method has been used by taking use of the classic Run Length Smoothing Algorithm (RLSA) in horizontal direction followed by an analysis of the connected components in the image. The main advantage here is that no prior knowledge of the page structure or character size is necessary for segmentation. We will explore that in detail in chapter 3.

2.2 Information retrieval – word spotting

Most of the work in the field of word spotting has been done on handwritten manuscripts. The reason for that mainly being the irregular writing styles that prevent commercial OCRs from achieving higher recognition rates. Printed document images are usually considered 'OCR friendly' as OCR software achieves relatively better on printed documents as compared to handwritten ones. But if the printed text is from old historical ancient documents, then OCR results on these document images degrade significantly. In that case, word spotting comes as a

lucrative alternate of OCR as B. Gatos in [Gatos and Pratikakis2009] remarked “OCR is a very difficult problem to solve, especially for historical [printed] documents”.

Historically, word spotting methods have been divided into different categories in multiple ways by different researchers. For example, [Rothfeder *et al.* 2003] divided all word spotting methodologies into two main categories based on matching techniques. These two main matching techniques are *image based* matching techniques and *feature based* matching techniques. Image based matching includes the methods that compute word distances directly on image pixels (such as template matching using correlation [Lewis1995]) while the feature-based matching methods first compute certain features for word images and then those features are matched. But as most of the recent contributions use feature-based implements, so this division does not serve the purpose for us. Another more popular categorization technique has been to divide the methods into either segmentation-based methods or segmentation-free methods as in [Gatos and Pratikakis2009], [Adamek *et al.* 2007], [Konidaris *et al.* 2007], [Rath *et al.* 2002], [Madhvanath and Govindaraju2001], [Steinherz *et al.* 1999]. We have followed this categorization theme as well and have divided the different approaches for word spotting into two broad categories:

- a. Holistic analysis techniques
- b. Analytical recognition techniques

Holistic word recognition techniques are basically *segmentation-free* techniques that view a word image as a unit that will not be further segmented. Analytical techniques, on the other hand, are *segmentation-based* techniques in which a document image or a word image is segmented into smaller units which can be recognized independently or when grouped. As word spotting has mostly been applied on hand-written documents where segmentation of characters is a big issue, so most of the techniques in literature are based on the holistic analysis category but lately work is also being done in analytical techniques as well with the objective of achieving higher recognition rates. Here we will analyze both matching categories by discussing different existing methods belonging to each of them.

2.2.1 Holistic Analysis techniques

Holistic word recognition techniques [Gatos and Pratikakis2009], [Rath and Manmatha2007], [Adamek *et al.* 2007] view word images as a unit that will not be further segmented. They are often motivated by the word superiority effect [Reicher1969] which tells that humans can recognize characters faster if they appear in valid words than in isolation. In the domain of ancient documents, other factors make a holistic approach attractive, such as their ability to deal with high level of noise and the font variations in the text, which can complicate the character segmentation.

Using a holistic approach helps to avoid character segmentation and whole words can be matched directly with acceptable recognition rates. Here we will discuss some of the recent holistic analysis techniques proposed in literature.

[Li *et al.* 2009] proposed a retrieval algorithm in which a simple local feature sequence is extracted for each document image. The feature sequence contains the pixel length of each word in the document image. A query which is a portion (a text line or a paragraph) of a document image is given and the document images containing that patch are retrieved by matching the feature sequences using suffix tree and dynamic programming. The method is tested on different contemporary document images with a retrieval precision of 99.5%. Doubts remain though over the performance of the method for ancient document images.

In [Rothfeder *et al.* 2003], certain points of interest are found out for each word image using Harris Corner detector. An example of the corner points is shown in Figure 2.7a.

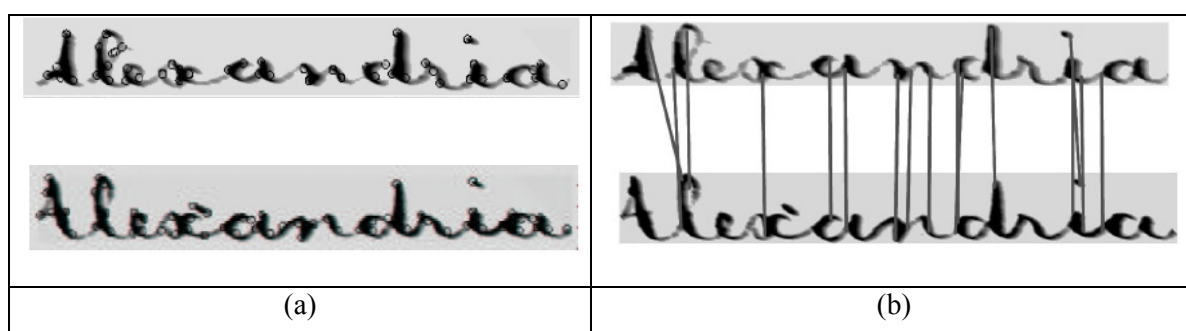


Figure 2.7 – a) Corners detected with the Harris corner detector on two gray level images, b) Recovered correspondences in two words (Rothfeder *et al.* , 2003)

Relative corner correspondences are found using Sum of Squared distances (SSD) error measure for comparing the gray level intensity windows centred around the detected corner point. Two words need to be of same size to retrieve the correspondences, so all candidate words are first resized to the size of the query word. An example of retrieved correspondences between the two words is shown in Figure 2.7b. For comparing two words A and B, the two sets of corner points are matched using Euclidean distance of correspondences as:

$$D(A, B) = \frac{\sum_i \sqrt{(x_{bi} - x_{ai})^2 + (y_{bi} - y_{ai})^2}}{\#\text{correspondences}}$$

where A and B are the query and test word images respectively and (x_{bi}, x_{ai}) and (y_{bi}, y_{ai}) are the co-ordinates of a pair of corresponding feature points, in the query image and the test image

respectively. Using this approach, the authors were able to achieve about 63% recognition rate on good quality images and 16% on poor quality images. Similar work has been done in [Rusinol and Lladós2008] where key point features are determined for words and symbols and comparison is done using cross-correlation matching. [Andreev and Kirov2009] have discussed a customized Hausdorff distance for matching two words in image space.

[Marinai *et al.* 2006] described a general system for performing word image retrieval using Self organizing maps (SOM) based on word image clustering combined with Principal component analysis (PCA). The words are extracted using RLSA and are divided into six index partitions based on their aspect ratio. Vectorial representation of the words is obtained by average gray value of the grid cells. Clustering is performed on each subset of the partition. SOM which is a kind of artificial neural network is used for clustering of the word images. The restriction of SOM clustering though is the need to compute a new set of clusters when dealing with different documents. Query words are generated using Times font and are searched in the clusters to get the top three clusters which are then analyzed using PCA in projected space to get the final top 20 ranked words. The testing of the method has been done on 2 books of 19th century. This work has been extended in [Marinai *et al.* 2007] to build a framework for document retrieval in digital libraries.

In [Zagoris *et al.* 2006], query word is synthetically generated using character images written in Arial font. A fixed length feature vector is defined for each word using nine features comprising six scalar characteristics and initial coefficients of the discrete cosine transform of three profile features). Two words are matched using Euclidean distance on the two feature vectors. For a total of 30 searches on a document set created automatically from different printed documents, the system achieves a mean Precision of 53.43% and a mean recall of 94.78%. Similar work has been done in [Konidaris *et al.* 2007] where authors use synthetically generated query word images from the manually selected prototype characters in Greek historical documents. The spacing between characters has been set to 10% of the average character height in the images for query generation as shown in Figure 2.8.

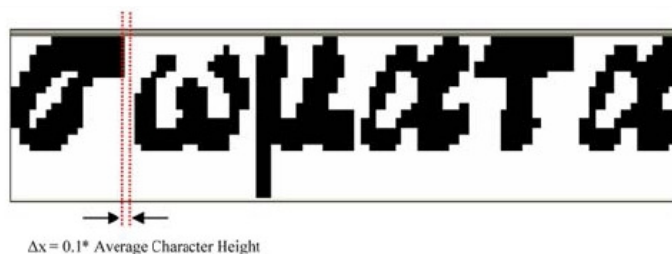


Figure 2.8 - Synthetically generated query word from prototype character images (Konidaris *et al.* , 2007)

These query words are normalized to fit in a pre-defined bounding box (Figure 2.9a). Two types of features are defined for a word image. In the first type, the area formed by the upper and lower profiles of the word is calculated in 30 small zones each (see Figure 2.9b, c). In the second type, the image is divided into a set of 90 zones and density of character pixels is calculated in each zone (see Figure 2.9d). So it makes a total of 150 features for each word

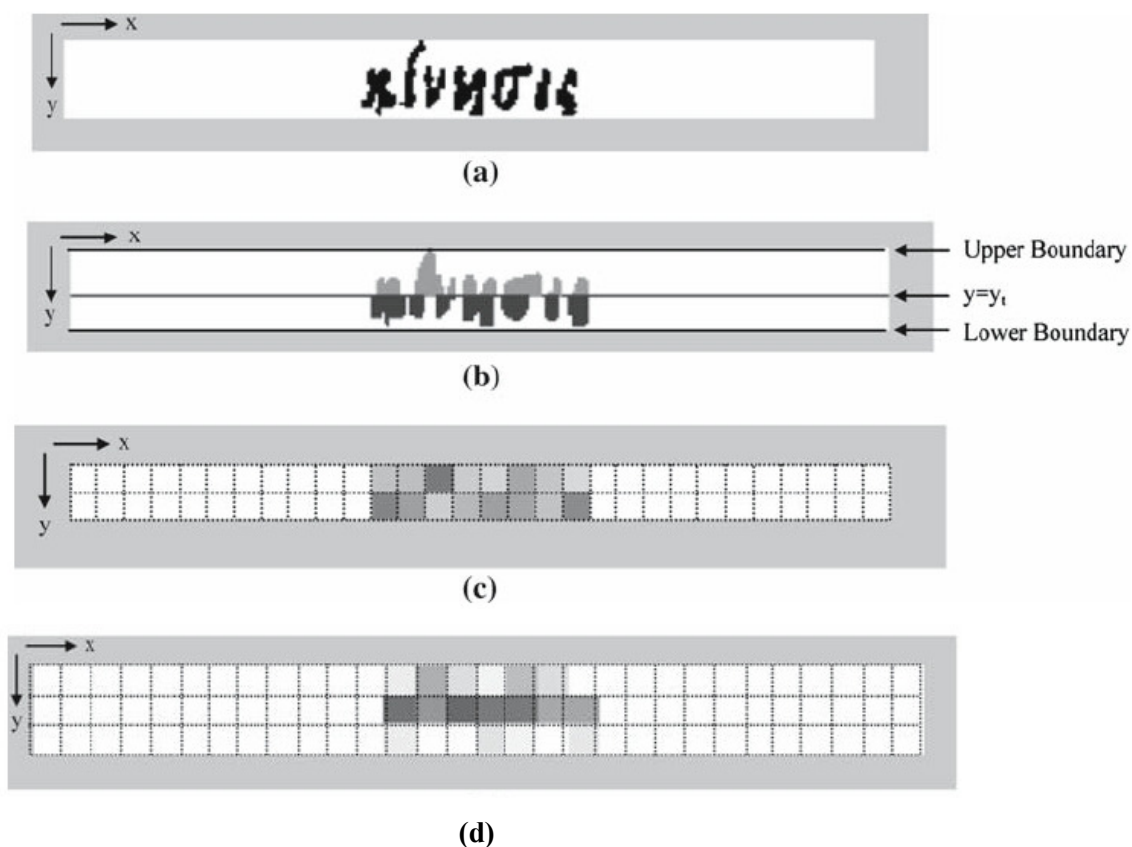


Figure 2.9 - a) Normalized query word in a pre-defined bounding box b) Upper and lower profiles c) extracted features, darker squares show higher pixel density d) Pixel density based features in the defined zones (Konidakis *et al.*, 2007)

For word matching, a simple distance measure is used between the features of the two words. For experiments, 25 query words are searched in 100 Greek printed document images. A recall rate of 65% with about 60% precision is achieved with user feed back. The problem with this approach is that there are usually lot of ‘unused’ zones and thus ‘non-features’ and they are compared in any case in the distance which is not very efficient (though it will not have any impact on the results).

Word shape coding technique has been used in [Adamek *et al.* 2007], [Bai *et al.* 2009] and [Bertolami *et al.* 2008] for word image matching. [Adamek *et al.* 2007] proposed a closed contour matching technique in which the contours of the two word images are matched using DTW. The bounding boxes of the words in the document images are already known. Separated parts in the

words are connected using position of words base line and height. Connected components are labelled and smaller components are removed from each word. The problem in this approach is that all the diacritic marks are removed as well which may not be useful for us as we are working on French language documents which have plenty of accents and diacritic marks. Successive components in the word are connected by manually adding a synthetic ink line in the binary image. This is followed by a contour tracing procedure which extracts a single order contour for each word. Multi-scale convexity concavity (MCC) representation is used to describe the word contours. This representation gives information about the amount of convexity and concavity at different scale levels for each contour point. 1D discrete cosine transform (DCT) is applied to each multi-scale contour point to get an MCC-DCT representation. A DTW technique is used to find the alignment along the two contours upon which the dissimilarity measure is defined. Figure 2.10 shows an example of DTW matching on the MCC-DCT representations of the two contours of word “Orders” taken from the paper itself. Word spotting results were computed on 20 document images and performance was measured in terms of average word error rate (WER). Excluding out of vocabulary errors, the system obtained an average WER of 17.4% and if these errors are included, the WER increases to 30.6%.

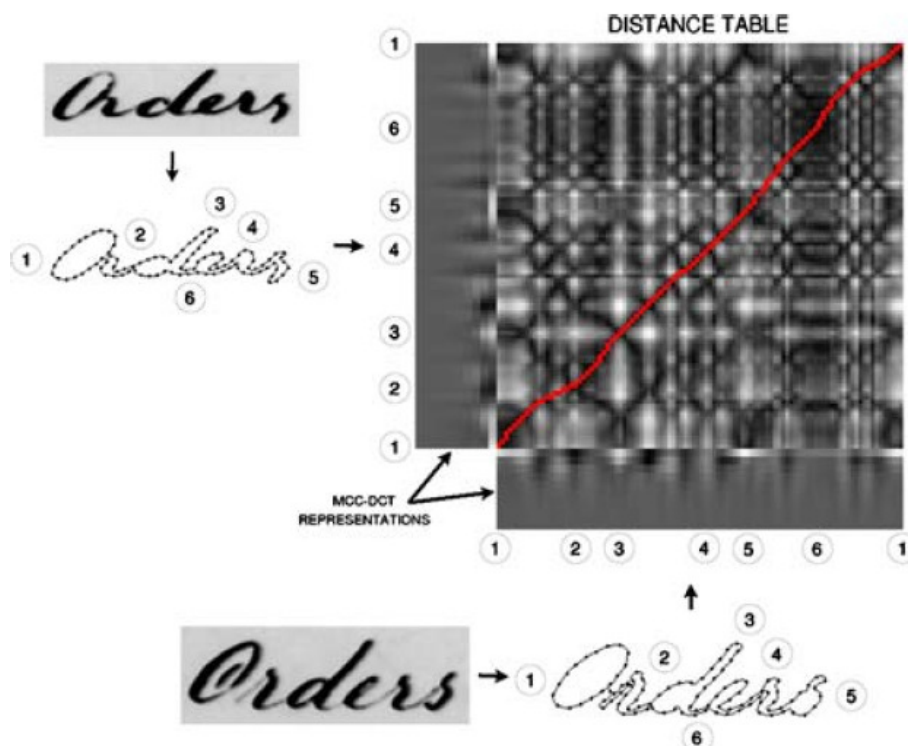


Figure 2.10 - DTW matching on the MCC-DCT representations of the two contours
(Adamek et al. , 2007)

Gatos et al. in [Gatos and Pratikakis2009] have presented a segmentation free approach where the query word image is searched in the regions of interest in the test document image. For each query image, 15 different query instances are obtained by applying different rotation and scaling variations. Five different set of feature vectors are found for the query word. These features are based on all 5x5 non-overlapping window pixel density calculations and applying word image translation at (-2,2), (2,2), (2,-2) and (-2,-2). For test image, regions of interest (which are the text lines in the image) are found by using RLSA in horizontal direction. At the word matching step, query word is compared with rectangular text areas in the test by moving a rectangular window over the region of interest in the test image. The corresponding feature vectors of the query word and test rectangular area are compared using a distance measure. In the end all matching results are combined. If there are several intersecting rectangular areas that correspond to successful matching results then authors select only that rectangular area which corresponds to the lower distance value. The evaluation of the method was performed on a late 18th century book. Five query words were manually selected and were searched in 100 pages of the book. Overall, the system achieved a recall rate of 93.2% with 75.1% precision.

In [Kluzner *et al.* 2009], the authors propose an adaptive OCR system based on clustering together all the similar words in a book and simultaneously handling an entire class. The image is segmented into words using the FineReader engine. The system then finds the word clusters with each cluster containing images presumed to show different instances of the same word. This allows recognition to be performed simultaneously for a cluster. The main problem that the authors address is the comparison of word images determining which words are the same. A two-step methodology has been used that comprises: *Image Distortion Compensation* and *Difference Detection*. Image distortion compensation first uses a coarse registration based on cross-correlation and then performs fine registration using a modified optical flow method. The difference between the two binary images is computed using a non-linear difference measure. On a data set of 101 scanned pages from a book printed in 18th century, the authors demonstrate that the commercial OCR results improve by more than 4% by adding the proposed adaptive OCR system.

The main motivation of our work came from the work of Rath and Manmatha [Rath and Manmatha2007]. In fact, Manmatha first used the notion of word spotting for indexing the document images and information retrieval in his early work [Manmatha *et al.* 1996a], [Manmatha *et al.* 1996b]. Later on, Rath continued the research in word spotting in historical document images for document image retrieval as well as indexing in which word images are grouped together into clusters of similar words [Rath and Manmatha2007]. Already processed word images (cleaned and with no skew/slant) are given as input to the system. Different features

were already evaluated by authors in [Rath and Manmatha2003] and only the best four are selected here for representing a word image. These include vertical projection, upper and lower profiles, and background/ink transitions which are depicted in Figure 2.11.

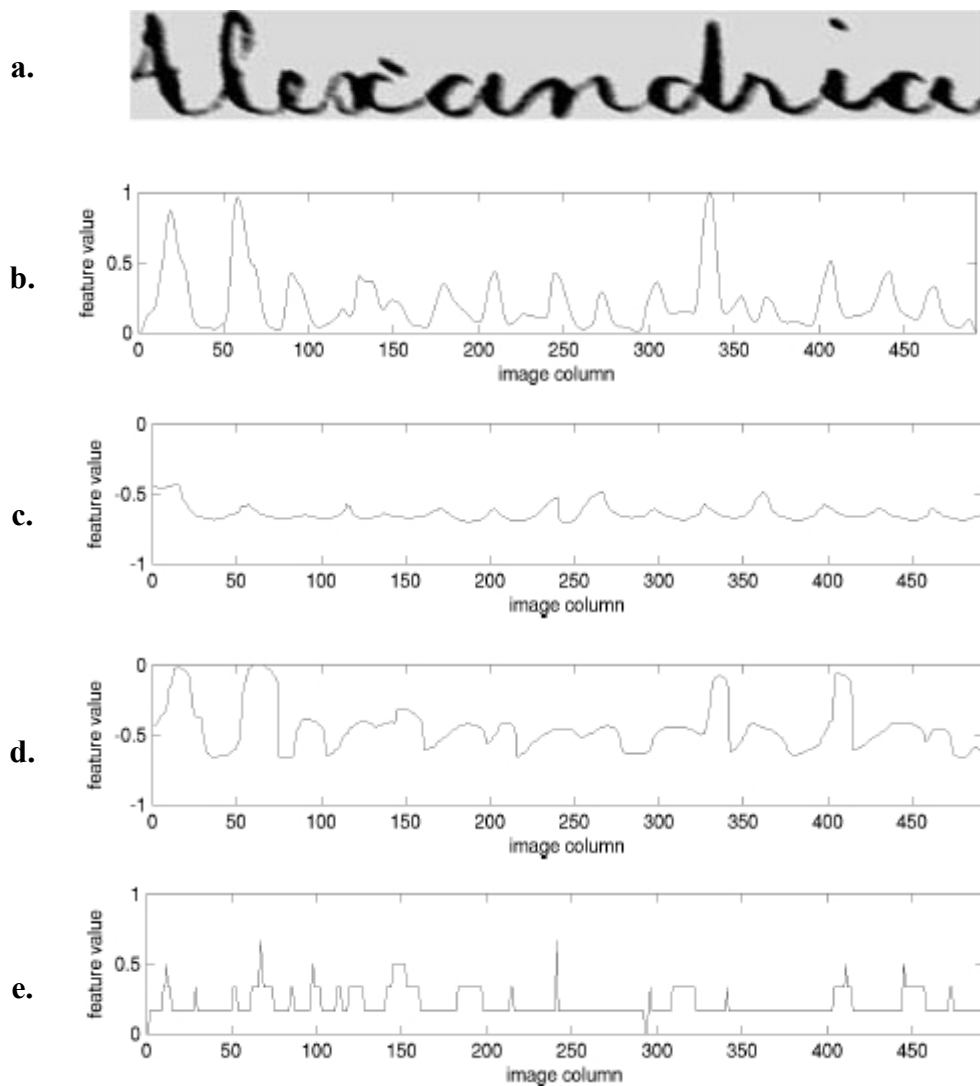


Figure 2.11 - a) Original word image: slant/skew normalized and cleaned. The four features: b) Normalized vertical projection profile, c) Normalized lower profile, d) Normalized upper profile, e) Normalized number of ink/background transitions (Rath and Manmatha, 2003)

For matching, DTW with Euclidean distance is used to match the features of the two words. Experiments were carried out using two data sets of words taken from 10 pages each. First data set comprising good quality document images while other one not so good. For each data set, two tests were performed, test-one with small number of query words (15 and 32 for the two data sets), and test-two with all words in the data set taken as query. For data set 1, the overall

precision rates reported in the paper are 73.71% and 65.34% for the two tests. Same percentages for data set 2 come out to be 58.81% and 51.81% for test-one and test-two respectively.

Similar methods have been proposed in literature, which we will mention briefly here. [Kolcz *et al.* 2000] have matched word images from hand written document images using DTW algorithm. Three profile features are found out for each word. For experiments, 19 model images for four most common words are selected. Word image comparison algorithm is based on matching the provided templates to segmented manuscript lines. The model words are searched in a data base consisting of 13 hand written documents with a conclusion that the system works better for longer words as compared to shorter words. In [Konidaris *et al.* 2008] query words are generated synthetically from the manually selected character prototypes. Four profile feature set of [Rath and Manmatha2007] are used here as well which are matched using DTW. The method uses word portions located at the beginning and end of each segmented word to estimate the position of the first and last characters in order to reduce the list of candidate words. A recall rate of 50% with 84% precision is achieved on 100 Greek printed documents for 25 query words.

2.2.2 Analytical Recognition techniques

Analytical recognition techniques segment a whole document image or word images into smaller units that can be recognized in isolation or when grouped [Vamvakas *et al.* 2008], [Marti and Bunke2001], [Terasawa *et al.* 2009]. Characters are a natural unit in alphabetical languages, so a word should be segmented into characters. However, accurately determining the segmentation points cannot be done without first recognizing the characters. This constraint has led researchers to consider multiple segmentation hypotheses by over segmenting images into smaller units, such as strokes, image columns or connected components. In these approaches, the correct segmentation into characters typically arises implicitly from the recognition process, which attributes segments to recognized characters. Other approaches use explicit word segmentation to break a word into smaller units that are believed to be characters, which are then recognized [Lu and Shridhar1996].

Though segmentation into characters is difficult, the analytical approaches tend to give better recognition results due to their ability to focus on the local intrinsic characteristics of the words and lower level matching. Another major advantage of all segmentation-based methods is their flexibility with respect to the size and nature of the lexicon which is due to the fact that these methods are being letter-oriented [Steinherz *et al.* 1999]. Here we discuss some of the recent segmentation based recognition approaches in literature.

[Vamvakas *et al.* 2008] have proposed an OCR methodology to generate ASCII files for a new document image based on the training set. Document image is segmented into words, and for each

word its characters are extracted by a bottom up approach using connected component analysis and skeleton features. Each character image is, first, normalized to fit in a pre-defined window size and then it is represented by a fixed length feature vector based on the character's zone and area properties as described in detail in Figure 2.12.

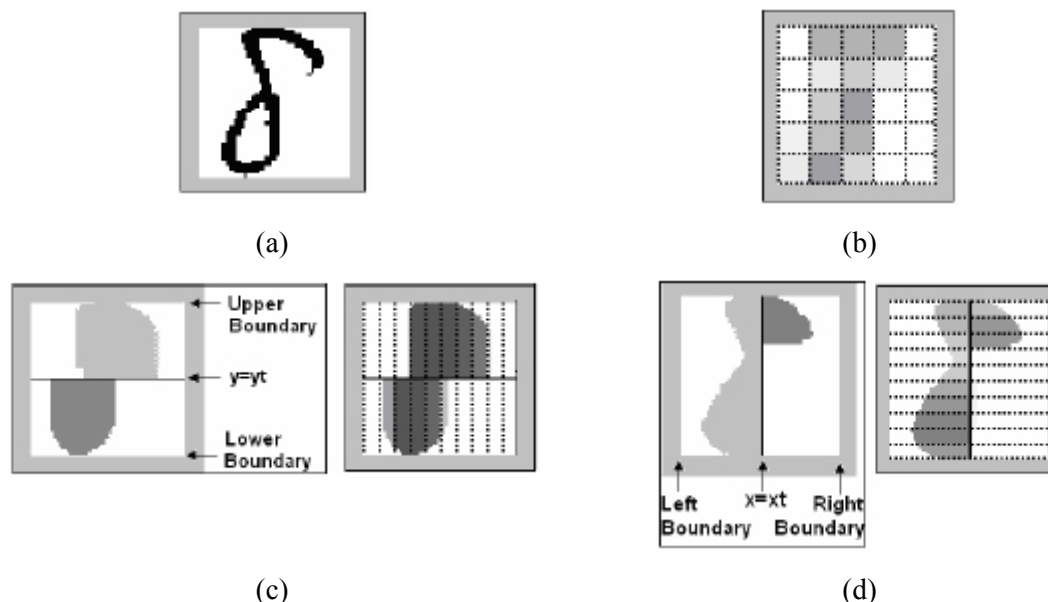


Figure 2.12 - a) Normalized character image, b) Features based on zones; darker squares indicate higher density of character pixels, c) Area formed by upper and lower profiles and the features extracted using them; darker areas indicate higher pixel density, d) Area formed by left and right profiles and features extracted using them; darker areas indicate higher pixel density (Vamvakas et al. , 2008)

A total of 25 zone feature values and 40 area feature values are defined for each character image thus making the size of the feature vector equal to 65. A k-means based semi-automated clustering algorithm is applied to cluster similar characters together. Number of clusters is hypothetically set to 65. The feature vectors of two characters are compared using support vector algorithm (SVM). Experimental validation is done using 10 printed document images. 4/5 of the total characters in these 10 documents are used in training while the remaining 1/5 are used for testing. Character recognition rate of 95.44% is achieved by the system. The recognition rate however drops to 83.66% for 5 other unknown printed document images which are not used in training.

[Moghaddam and Cheriet2009] presented a connected component based method for word spotting on cursive Arabic scripts. All the connected components in the document image are found and a basic connected component (BCC) library is generated which contains the basic

connected components found in the text. This library is extended by matching new CCs to the existing ones in the BCC. Matching is done using dynamic time warping by matching the normalized horizontal and vertical histograms of the two CCs. In order to increase the accuracy of comparison and also to reduce the computational cost, a clustering of BCCs in a set of meta-classes is done. The clustering is performed using six feature characteristics of CCs like aspect ratio, horizontal frequency, scaled vertical center of mass, number of branch points, height ratio to line height, and presence of holes. SOM are used for clustering. Each CC is compared to all clusters and then based on the number of target clusters, the nearest BCCs are selected as matching BCCs of that CC. The evaluation of the method has been performed on 85 images and an F-measure of 74% is achieved if the dots in the text are ignored. With the dots included, the same value drops to 57%.

In [Terasawa *et al.* 2009], authors have introduced a slit style HOG (Histogram of oriented gradient) feature based method. A narrow rectangular sliding window is applied on each text line (see Figure 2.13) which slides along the writing direction. For each sub-image clipped by the window, a HOG feature vector is calculated. HOG computes a histogram of gradient orientations in a certain local region, with the orientation bins evenly spaced over 0 – 360 degrees. The most effective number of bins is reported to be either 12 or 16.

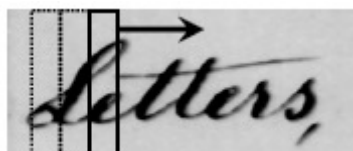


Figure 2.13 - Use of a sliding window for extracting small fragments for which the features are defined (Terasawa *et al.* , 2009)

A block optimization is applied to each slit window. It means that each slit window is further divided into small blocks which overlap in vertical direction. The dimensions of the slit window and inner blocks depend upon the text. For Latin text, slit window dimension is set to 16 x 80 pixels while that for each block in the slit window is 4x20. Gradient features are calculated at block level. For matching, query word is searched using the look up table technique by matching the feature vectors of the slit windows using DTW. Though there is a lot of redundancy in using block features, it is reported that these redundancies improve the word recognition rates. For Latin text, an average precision of 79.14% is reported for an average recognition rate of about 74%.

A similar gradient-feature based method has been proposed in [Rodriguez-Serrano and Perronnin2009] where the authors use modern fonts to create synthesized queries for matching with handwritten text. Local gradient histogram (LGH) based features are used to encode word shape robustly. A sliding window is used to divide word image into T overlapping windows. Each

window is further divided into a 4x4 grid, and gradient histogram is calculated for each of these 16 cells. Thus for a window, a total of 16x8 features are defined (see Figure 2.14).

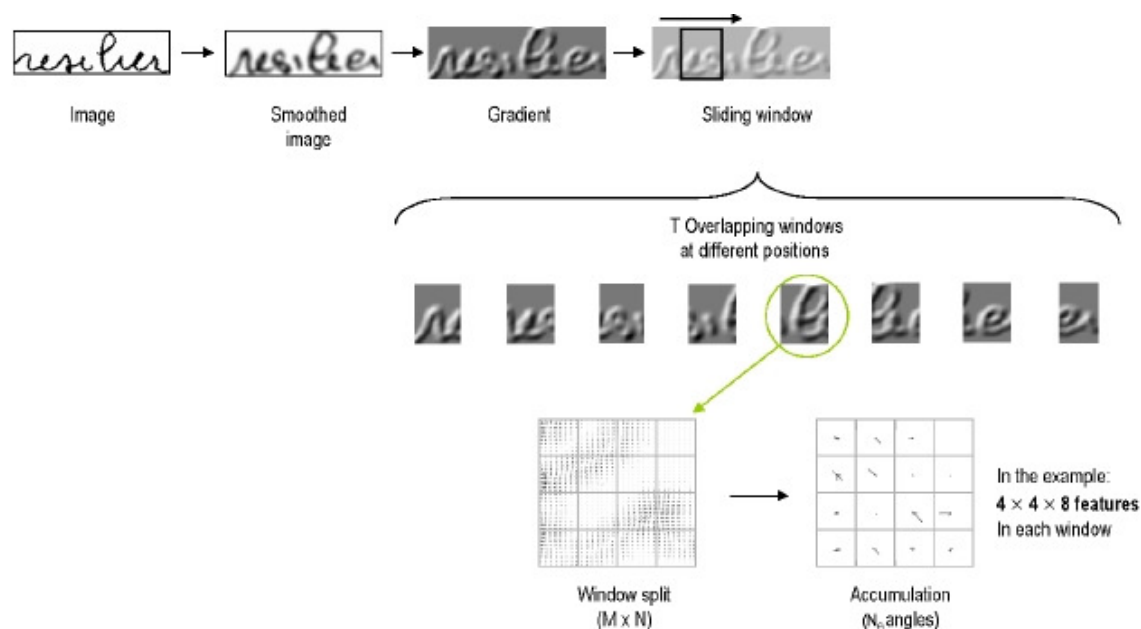


Figure 2.14 - LGH features calculated for small overlapping windows of a word (Rodriguez-Serrano and Perronnin, 2009)

For word modeling, a semi-continuous Hidden Markov Model (SC-HMM) is used in which the feature space is clustered using a Gaussian mixture model (GMM). GMM is trained offline with a large set of LGH feature extracted from many windows in actual handwritten images. For the synthetic query words, an LGH feature sequence is extracted for each word and SC-HMM is trained using these sequences. 25 different fonts are used for generating 10 keywords which are searched in a document base of 101 pages. Results obtained using some of the fonts (which are handwritten-like) giving better recognition results than others.

In [Leydier *et al.* 2005], authors present two alternative methods for textual indexation of old documents, the *computer assisted transcription* and the *word spotting*. Computer assisted transcription is based on character pattern redundancy in document images. Characters are first segmented and compared to one another to create a pattern dictionary. All characters within a class have a same label which removes the requirement to recognize all the characters of a class. Manual transcription of 50% of the pattern dictionary, the authors achieved a correct transcription of 80% of an entire book (200 pages and 2000 characters per page) in 3 hours.

For word spotting, the authors tested a number of differential features and selected the gradient angle on the basis of the best P-R curve on a validation set. The authors suggest selecting the relevant Zones of Interest (ZOI) to increase efficiency and improve the accuracy. Morphological

operations are performed on the word image to get the guides and the enlarged bounding boxes of these guides are identified as the zones of interest (see Figure 2.15).

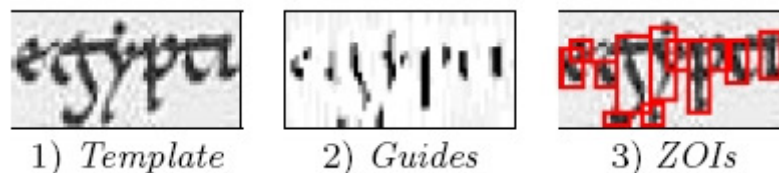


Figure 2.15 - Guides and ZOIs for a template word image (Leydier *et al.* , 2005)

The template is thus split into pieces using the ZOIs and the distance and orientation between the centers of these zones are stored. The first ZOI of the template is matched sequentially with each ZOI of the test image while the other ZOIs of the template are matched with small displacement possibility as shown in Figure 2.16.

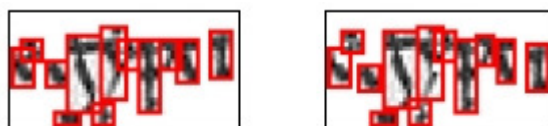


Figure 2.16 - A possible displacement of ZOIs (Leydier *et al.* , 2005)

The authors report results of two experiments. The first one is carried out on 185 images of two column pages with 20 lines per column where the search of the word ‘fyon’ found 28 good hits out of 28 and the time of execution was 260mins. On the same document images, the authors then searched the word ‘egypt’ and found 15 occurrences out of 15 with the first bad hit at rank 13 and the last good hit at rank 68 and this search took 400mins. Overall, we can say that the results achieved by this approach are satisfactory but the execution time is too long which makes this approach less lucrative.

Having presented an overview of some of the recent and notable studies in the area, we will now sum up with a comparative analysis of these methods in the following section.

2.2.3 Comparison of the Retrieval methods - Discussion

There have been no standard criteria for the evaluation of different information retrieval / word spotting algorithms. [Terasawa *et al.* 2009] have recently proposed an automatic evaluation framework for word spotting algorithms which provides certain guidelines and protocols for obtaining a uniform standard in the evaluation process. If followed globally, it can prove to be useful in future

for a better and fair comparison of different methods. Traditionally, retrieval techniques have generally been evaluated by running a set of queries and analyzing the list of retrieved words. Two most common measures for judging the quality of the retrieved-words list are recall and precision. But as all these methods have been evaluated in different conditions on different proprietary databases sets with different number of test and query images, comparing the listed precision and recall rates portrays no true picture of the performance and efficiency of a particular method with respect to others, thus rendering a quantitative analysis irrelevant. A way though could be to program all these algorithms and then run the tests for the different methods in exact same computational conditions on one common data set. But due to many real life constraints, this is not feasible for us. So we will limit ourselves to a brief qualitative analysis of the different methods that we have discussed.

We have discussed two main categories of the document image retrieval methods which are holistic or segmentation-free techniques and analytical or segmentation-based techniques. Both the techniques have their pros and cons. Holistic analysis methods compare a sequence of observations derived from a word image with similar features for the words in the database. There are many factors that make this approach very attractive and natural. These factors mainly concern the poor quality and printing of historical documents that result in high level of noise, irregularity in printing and different font variations in ancient printed texts. All these factors can complicate the character segmentation process which itself is not an easy task. By using a holistic approach, we can avoid all the character segmentation issues and can match whole words directly with good recognition rates using the different methods discussed earlier in the chapter.

On the other hand, analytical methods look for the best match between consecutive sequences of primitive segments or characters of a word. Though segmentation of characters is a very difficult problem, especially for ancient documents [Casey and Lecolinet1996], the analytical approaches usually tend to give better recognition results as compared to their holistic counterparts. It is due to their ability to focus on the local intrinsic characteristics of words which give more in depth details of a word, thus differentiating between two different words becomes easier. Another important point is the low level matching, which takes more intrinsic details into the matching process, making the systems robust. Thus similar feature set and matching distance could give better results when these features are defined for characters as compared to same features defined for words as evaluated in [Khurshid *et al.* 2008a]. Another major advantage of all segmentation-based methods is their flexibility with respect to the size and nature of the lexicon, which is a result of these methods being letter-oriented.

From the above discussion, we can conclude that though plenty of work has been done in the past for information retrieval using word spotting, there is always plenty of room for improvement, especially when dealing with large volumes of old historical books. The challenges these old documents pose are enormous, and researches around the globe continue to thrive on these challenges, bringing forward their propositions for more robust and more efficient systems. In the next couple of chapters, we will discuss our approach of information retrieval using an analytical word spotting methodology. We will also demonstrate the efficiency, flexibility and robustness of our approach when we compare our system with other state of the art methods as well as commercial OCR results. We start the discussion from the document image indexing in next chapter.

Chapter 3

Document Image Indexing

Document image retrieval using word spotting has been a very hot research topic in document analysis domain after the advent of the digital library concept. In the last chapter, we discussed different approaches for word spotting that have been proposed over the years to facilitate information searching. In our work, we have employed a granular approach for word spotting by introducing a two level dynamic matching approach – *at word level and character level*. For that, the first step we perform is the indexing of document images that includes the segmentation of words and characters, defining features for the characters and storing all this information in individual data files. It is a time consuming process, that's why document image indexing is done beforehand /offline to facilitate users in rapid information search and document image retrieval.

In this chapter, we focus on the indexing of document images. The whole indexing process has been divided into four sub-steps:

- The first step is the preprocessing stage involving an optimized document binarization algorithm to crisply distinguish the foreground from the background.
- Second step involves the segmentation of words and graphics.
- As we are working on character level, the third step revolves around the connected component analysis of words for the extraction of characters. The connected components of a word are post-processed using a 3-step process to get S-characters which are the best possible approximation of the T-characters.
- In the last step, we calculate certain features for each S-character in a word and store them along with some other information in an index file. Thus for each document image, we create one index file. Figure 3.1 shows the block diagram of these stages.

Now we talk in detail about each of these stages.

3.1 Document Image Binarization

In an OCR or text extraction application, one of the pre-processing stages usually is binarization of document images, i.e. separation of foreground from background. Binarization of a text image should give us, in an ideal case, the foreground text in black and background in white. Though different thresholding methods already exist in literature, they don't give perfect results for all types of documents [Leedham *et al.* 2003]. Some algorithms might work better for one type

of documents where there are marks of strain while they might give poor results for other types where there are extremely low intensity variations.

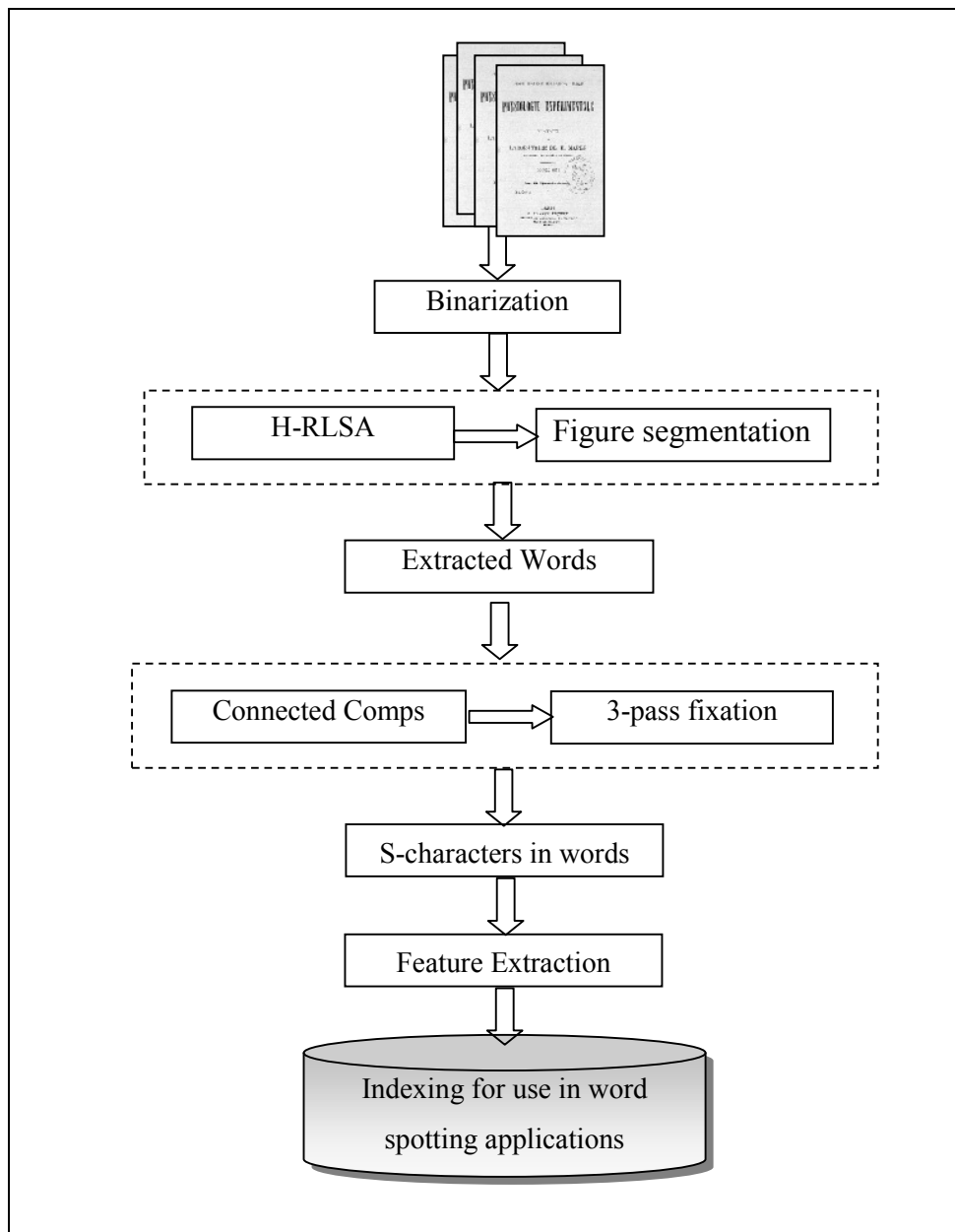


Figure 3.1 - Different stages of Document Image Indexing in our implementation

The problem of document binarization is as old as document image analysis itself. A large number of binarization techniques have been proposed over the last two decades. These techniques can generally be classified into two categories i.e. *global thresholding* and *local thresholding*. **Global thresholding** methods employ a single intensity threshold value i.e. one

fixed threshold is used for one image. The global method which is most common and is still used for the documents where a global threshold will suffice is by Otsu. In this method, the threshold value is calculated based on some heuristics of some global image attributes to classify image pixels into foreground (text) or background (non-text) pixels [Otsu1979]. The main drawback of global methods is that they can not adapt well to uneven illumination and noise, hence do not perform well on low quality document images.

Local thresholding methods, on the other hand, compute a threshold for each pixel (or group of pixels) in the image on the basis of the content in its neighborhood. As opposed to global thresholding, local methods generally perform better for low quality images, specially classifying pixels near text and object boundaries as either foreground or background. Different binarization methods have been evaluated in [Gatos *et al.* 2006] and [Leedham *et al.* 2003] for different types of documents. In [Gatos *et al.* 2006], authors have presented a multi-step method specific for document images in which, first, a low pass Wiener filter is used for preprocessing of the document image to obtain image I. Next, the authors use Sauvola's method [Sauvola and Pietikäinen2000] to extract the initial binary image S. It is followed by a background estimation of I to get image B. Final thresholding is done using image I and B. A pixel is a text pixel if the distance between the corresponding pixels in I and B exceeds a threshold d . In [Øivind D. Trier and Taxt1995], authors present an evaluation of eleven locally adaptive binarization methods for gray scale images with low contrast, variable background intensity and noise. In that evaluation, Niblack's method [Niblack1986] was found to be the best of them all. Different improvements have since been made to the original Niblack's method to improve the results. These include Sauvola's algorithm [Sauvola and Pietikäinen2000], Wolf's work [Wolf and Jolion2003] and Feng's method [Feng and Tan2004].

We tried fixed global thresholding on our document images but the results were not satisfactory. On the contrary, when we tested some local sliding-window based thresholding methods on our images, results were immediately better. We have developed a customized binarization algorithm 'NICK' [Khurshid *et al.* 2009c] by improving the original Niblack's formula. The results achieved using NICK are better than other Niblack inspired methods as later shown in the results. Here we give a brief account of the Niblack inspired sliding window methods that we tested and compare them with the proposed method NICK to analyze the improvements in results.

3.1.1 Well known sliding-window-based methods in literature

In this section, we give an account of some of the well known local sliding-window binarization methods which we tested on our data set to see how they fared on the same base. These include Niblack's method [Niblack1986], Sauvola's algorithm [Sauvola and Pietikäinen2000], Wolf's

method [Wolf and Jolion2003] and Feng’s method [Feng and Tan2004]. The results of the comparison are given later in this section.

3.1.1.1 Niblack’s Algorithm

Niblack’s algorithm [Niblack1986] calculates a pixel-wise threshold in a rectangular window over the gray level image. The computation of threshold is based on the local mean m and the standard deviation s of all the pixels gray levels in the window and is given by the formula 3.1 below:

$$\begin{aligned}
 T_{Niblack} &= m + k * s \\
 T_{Niblack} &= m + k \sqrt{\frac{1}{NP} \sum_{i=1}^{NP} (p_i - m)^2} \\
 &= m + k \sqrt{\frac{\sum_{i=1}^{NP} p_i^2}{NP} - m^2} = m + k \sqrt{B}
 \end{aligned} \tag{3.1}$$

where NP is the number of pixels in the window, k is a constant fixed to -0.2 by the authors. Advantage of Niblack’s method is that it always identifies the text regions correctly as foreground but on the other hand tends to produce a large amount of binarization noise in non-text regions and text boundaries.

3.1.1.2 Sauvola’s Algorithm

Sauvola’s algorithm [Sauvola and Pietikäinen2000] improves the Niblack’s method by computing the threshold using the following formula:

$$T_{Sauvola} = m * (1 - k * (1 - \frac{s}{R}))$$

where k is set to 0.5 and R to 128. This method outperforms Niblack’s algorithm in images where the text pixels have near 0 gray-values and the background pixels have near 255 gray-values. However, in images where the gray values of text and non-text pixels are close to each other, the results degrade significantly.

3.1.1.3 Wolf’s Algorithm

To address the issues in Sauvola’s algorithm, *Wolf et al.* [Wolf and Jolion2003] propose to normalize the contrast and the mean gray value of the image and compute the threshold as:

$$T_{Wolf} = m * (1 - k) + k * M + k * \frac{S}{R} (m - M)$$

where k is fixed to 0.5, M is the minimum gray value of the image and R is set to the maximum gray-value standard deviation obtained over all the local neighborhoods (window).

This method in most cases outperforms its predecessors. However, degradation is observed in its performance if there is a sharp change in background gray values across the image. This is due to the fact that the values of M and R are calculated from the whole image. So even a small noisy patch could significantly influence M and R values thus eventually calculating misleading binarization thresholds.

3.1.1.4 Feng's Algorithm

Instead of calculating dynamic range of gray-value standard deviation from the whole image like Wolf, *Feng et al.* [Feng and Tan2004] propose calculating it locally introducing the notion of two local windows, one contained within the other. The values of local mean m , the minimum gray-level M , and standard deviation s are calculated in the primary local window while the dynamic range standard deviation R_s is calculated in the larger window termed as 'secondary local window'. Binarization threshold is then computed as:

$$T_{Feng} = (1 - \alpha_1) * m + \alpha_2 * \left(\frac{s}{R_s} \right) * (m - M) + \alpha_3 * M$$

where $\alpha_2 = k_1 (s/R_s)^\gamma$ and $\alpha_3 = k_2 (s/R_s)^\gamma$. Based on the experimental experiences of authors, γ is set to 2 while the values of other parameters, α_1 , k_1 and k_2 are proposed to be in the ranges 0.1-0.2, 0.15-0.25 and 0.01-0.05 respectively. This method addresses well the R-problem in the Wolf's algorithm. However the introduction of three parameters, as well as the size of the second window, determined empirically, leaves the robustness of this method questionable.

As we saw above, each method tries to resolve the drawbacks in the predecessor from which it is being driven. Niblack's method identifies the text regions correctly as foreground but produces a large amount of noise in non-text regions. Sauvola's method tries to overcome the noise problem in Niblack's method but it fails in images where the gray values of text and non-text pixels are close to each other. Wolf's method overcomes this problem of Sauvola's method though degradation in binarization results is observed if there is a sharp change in background gray values across the image. While for Feng's method, the robustness of the method is left questionable due to plenty of parameters which are determined empirically.

3.1.2 Proposed Method – NICK

We now put forward our proposition of calculating the binarization threshold which works better for many (if not all) types of degraded and noisy ancient documents. Instead of following the chain of one algorithm proposing modifications in its predecessors, we derive our thresholding formula from the basic Niblack algorithm, the parent of all the methods discussed earlier.

In NICK, binarization threshold is found out for each pixel by taking into account its neighbouring pixels in a sliding window using the following formula:

$$\begin{aligned}
 T &= m + k \sqrt{\frac{\left(\sum_{i=1}^{NP} p_i^2 - m^2 \right)}{NP}} \\
 &= m + k \sqrt{A}
 \end{aligned} \tag{3.2}$$

where ,

k is the NICK factor having value between -0.2 and -0.1

p_i = pixel value of gray scale image

NP = number of pixels in the window

m = mean gray value of these NP pixels

One major advantage of NICK over Niblack that we observed during experiments was that it considerably improves binarization for "white" and light page images by shifting down the binarization threshold to make sure no non-text areas are taken mistakenly as text. We show it mathematically by finding a relation between NICK and Niblack's methods. Using equation (3.1) and (3.2), we have:

$$\begin{aligned}
 (A) - (B) &= \frac{\sum p_i^2}{NP} - \frac{m^2}{NP} - \frac{\sum p_i^2}{NP} + m^2 = \\
 m^2 - \frac{m^2}{NP} &= m^2 \left(1 - \frac{1}{NP} \right) = m^2 \left(\frac{NP - 1}{NP} \right)
 \end{aligned}$$

If the value of NP is high, we can approximate $(A - B)$ by m^2 .

$$\begin{aligned}
 A - B &\cong m^2 \\
 \text{thus } A &\cong B + m^2
 \end{aligned}$$

It shows that if the image is very dark, the value of m is low meaning that the difference between A and B is very small. But if the image is light, the value of m is high and thus the difference between A and B is greater which lowers the binarization threshold for NICK.

$$T \cong m + k\sqrt{B + m^2} \quad (3.3)$$

The value of NICK factor k can vary from -0.1 to -0.2 depending upon the application requirement. Value of k close to -0.2 makes sure that noise is all but eliminated but characters can break a little bit, while with values close to -0.1, some noise pixels can be left but the text will be extracted crisply and unbroken as shown in Figure 3.2. So, for an OCR application, the value of k must be set at -0.1 and in applications where we don't desire any noise, k should be -0.2.

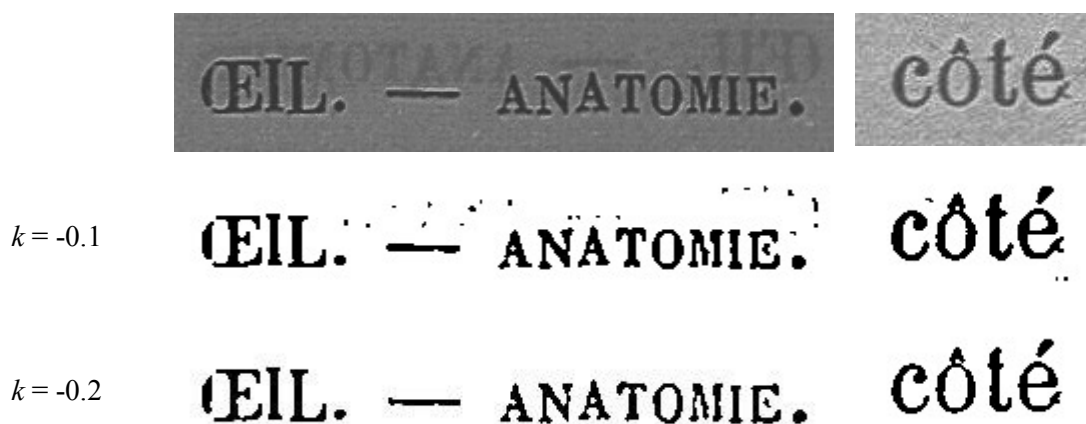


Figure 3.2 - Binarization using NICK with $k = -0.1$ and $k = -0.2$ at window size 19

Equation (3.3) shows that the difference between values of T and $T_{Niblack}$ increases with m . To highlight this difference, we applied NICK and Niblack's methods, globally (computing the binarization threshold using mean and standard deviation of complete image to get one threshold value for one image), to a set of 20 document images having different mean gray values. Figure 3.3 shows the variation of T and $T_{Niblack}$ for these images. For whiter images (m near 255), this difference becomes more significant. Now coefficient k being negative, T becomes smaller than $T_{Niblack}$, thus implying that fewer pixels are coded as black pixels. Figure 3.4 shows the result obtained with Niblack's method and NICK at $k = -0.2$ on the first three pages of a book.

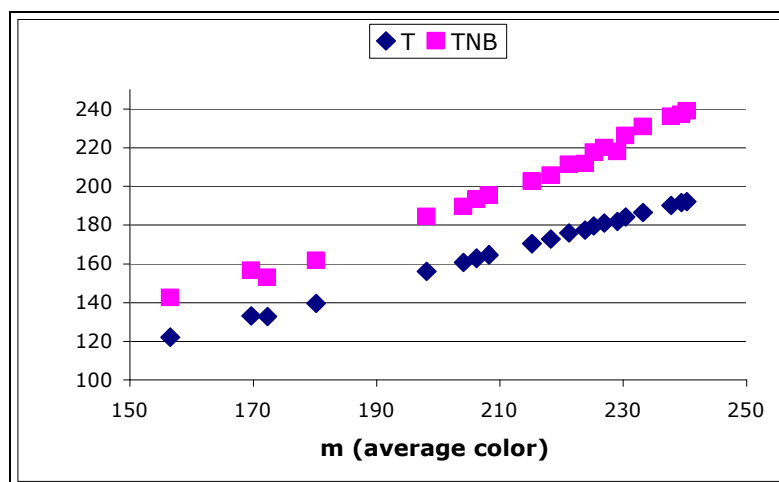


Figure 3.3 - T and TNB thresholds for different average gray levels m

To evaluate our method and compare the results with the local methods described above, we binarize the document images locally, computing the binarization threshold for each pixel of the input image using a sliding window. The window size (optimized empirically on the data set) has been set to 19×19 in our case. The results are given in the following section.

3.1.3 Comparison of the different methods

The binarization results formulated have been based on tests performed mainly on the images of the Bibliothèque Interuniversitaire de Médecine, Paris [BIUM]. The sample images are shown in Annex A. A total of 120 images, of size 1536×2549 pixels, were selected from the database and were binarized using Niblack's, Sauvola's, Wolf's and Feng's methods. The results of these four methods were compared with the results achieved by our method. All these methods have been evaluated for a window size of 19×19 , which was chosen considering the text size in the images, and with bigger window in Feng's method being kept at 33×33 . Some of the results obtained by these methods are shown in Table 3.1.

Based on visual criteria, the proposed algorithm seems to outperform the other methods with respect to image quality and preservation of meaningful textual information. After a thorough visual examination of the experimental results, important observations are summarized in the following:

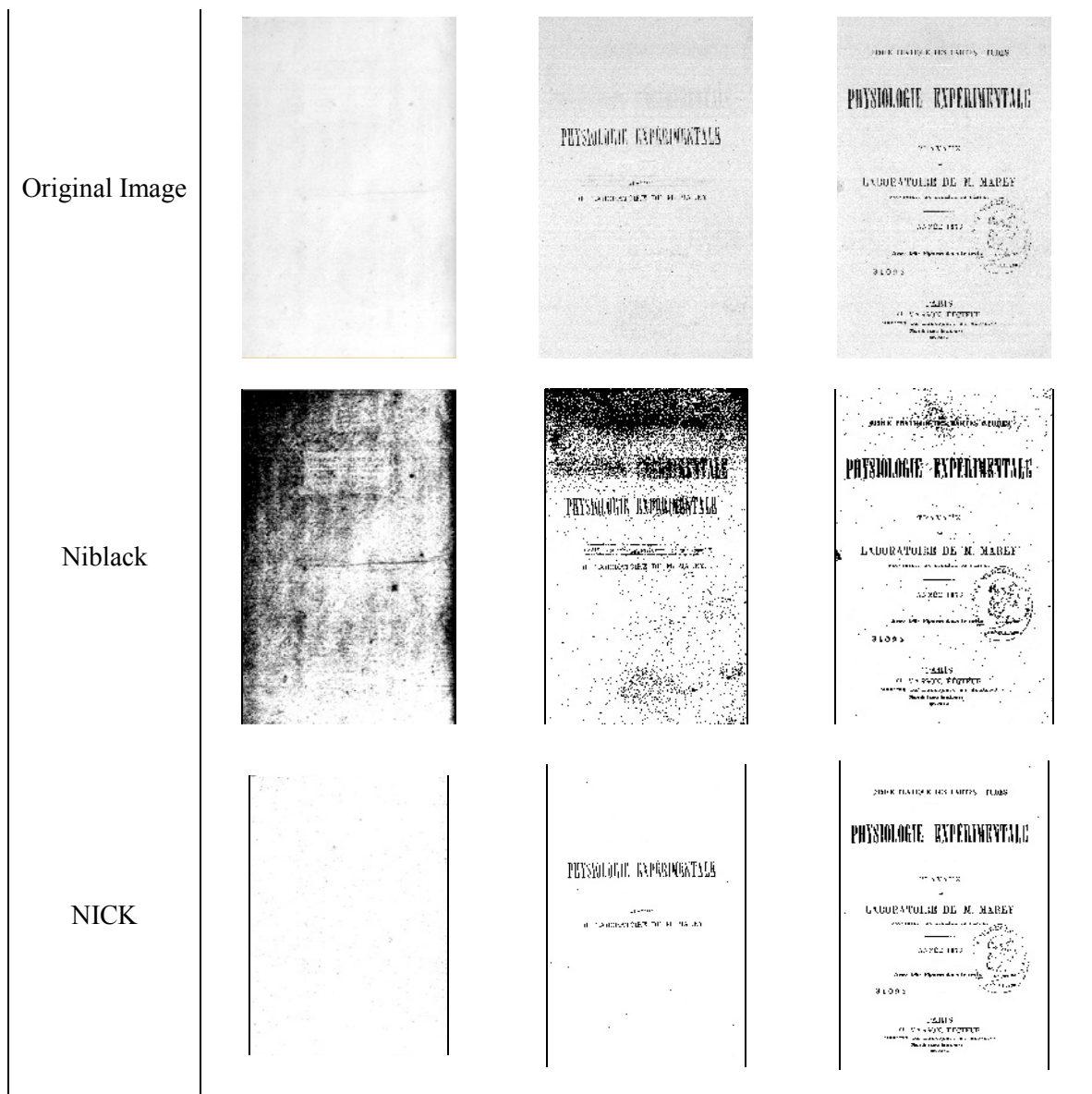


Figure 3.4 - Comparison of binarization using Niblack and NICK methods for first three pages of a book

- With Niblack’s approach, the resulting binary image generally suffers from a large amount of background noise, especially in areas without text.
- With Sauvola’s method, the background noise problem that appears in Niblack’s approach is solved but in many cases where there are less intensity variations, characters become extremely thinned and broken. In some cases, the characters disappear totally giving a white output image.

Table 3.1 - : Portion of output images obtained by using different methods at windows size 19, a) Niblack b) Sauvola c) Wolf d) Feng e) NICK

Portion of original Image	a	b
c	d	e

- In most of the cases Wolf’s algorithm outperforms the two predecessors; however there are occasions when the characters disappear or break if intensity variations are very small or there is some noisy patch with a very sharp intensity variation from the rest of the image.
- Feng’s method generally works very well but the main drawback remains its susceptibility to the empirically determined parameter values as discussed earlier. A slight change in parameter values could drastically affect the binarization results, as was observed in our experiments. One set of parameter values could give excellent results for one image but the same set would not work for another image with different intensity and illumination variations. Secondly, the introduction of a larger secondary window (around the primary window) also makes this method computationally inefficient as compared to the rest.
- NICK shows improved performance when compared to the other methods tested, and performs better especially when the images have extremely low intensity variations and for whiter images. Computationally, NICK is much more efficient as compared to Feng’s method as we have to find lesser parameters and that also within a single window. Table 3.1 shows portions of some output images.

As in [Feng and Tan2004], we did a threshold analysis to see the difference in local threshold values between these methods. For that, a synthetic image was created from a small portion of real text image from a historical book. We selected a pixel row 140 pixels wide from the text and extended it above to make it 19 pixels high. It is shown in Figure 3.5. Height of the synthetic image was set to 19, as the window size we used for binarization was set to 19. The ground truth image for the pixel row is also shown in Figure 3.5.

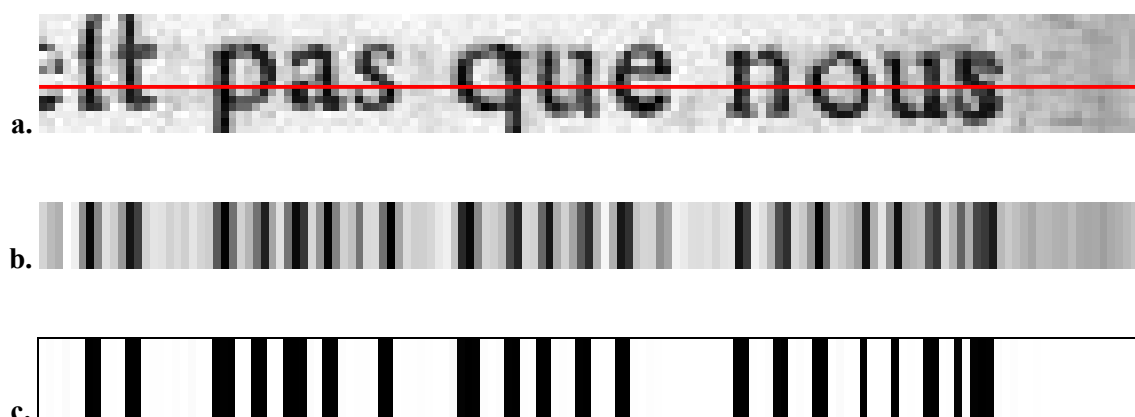


Figure 3.5 – Creation of a Scan Line Image: a) selection of a pixel line b) extending the pixel line in vertical direction c) ground truth of the synthetic image binarization

Different intensity variations were applied on the synthetic image to create its different variants and binarization thresholds were calculated for all of them using each of the five methods. It enabled us to see how each of the methods responds to the changing background conditions in an image. Figure 3.6 shows one example of the thresholds been determined along a scanline (central line) for one of the variant of our synthetic image obtained by changing the intensity levels in a photo editor. This analysis shows the effectiveness of our method in correctly discriminating the foreground and background regions. We can see from the graph that apart from Sauvola's method, rest of the methods gave a reasonable separation of text and background regions for the synthetic image. There were two occasions though where other methods wrongly identified noisy background areas as text whereas NICK was able to successfully identify those regions as background.

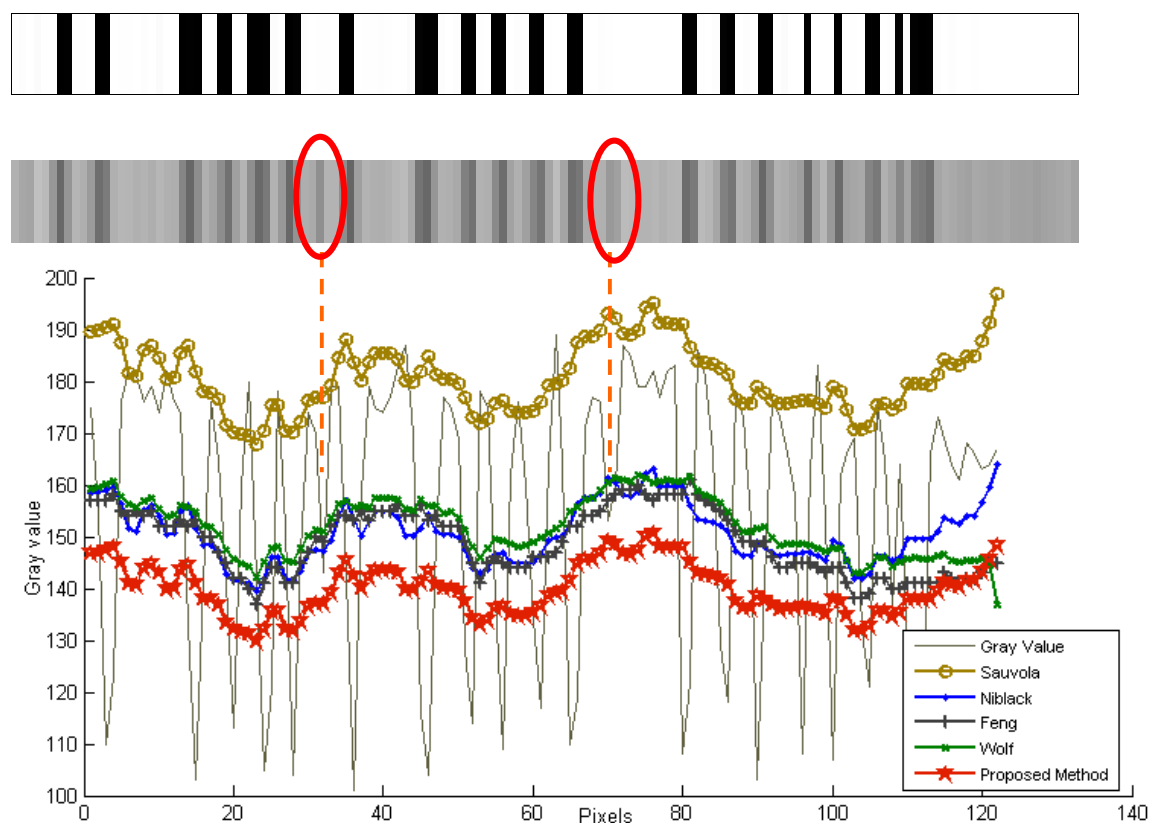


Figure 3.6 - Local thresholds obtained by different methods for the scan line image above

The evaluation of the different binarization methods is not an easy task. Apart from doing the visual qualitative analysis, we also chose to perform a quantitative analysis of these different methods as well. That's why, to quantify the efficiency of our proposed binarization method, an OCR experiment was performed on text samples having different intensity variations and quality.

These samples were selected from 26 representative BIUM images from different books. Four of the samples are shown in Figure 3.7. Character recognition was performed by the well-known OCR engine ABBYY Fine Reader 9 beta [ABBYY] on the binarization results of Niblack’s, Sauvola’s, Wolf’s & Feng’s methods as well as NICK on each of the 25 images. It is important to note that the actual OCR recognition rate is not important for us but the difference in recognition rates on the different binary images is important.

OCR results are quantified by character recognition rate, calculated as:

$$\text{Recognition rate} = \frac{\text{Number of Correctly Recognized Characters}}{\text{Total Characters in Ground Truth}} \times 100$$

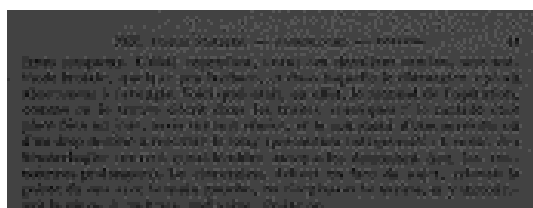


Image 1

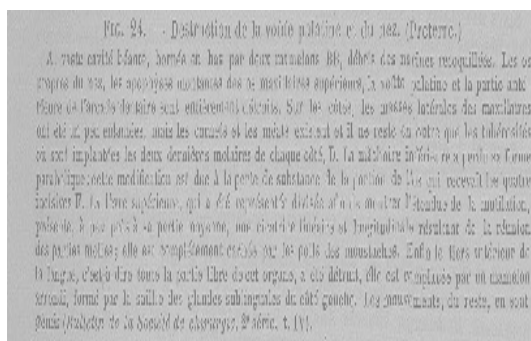


Image 2

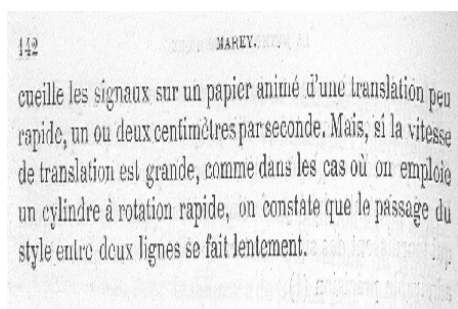


Image 3

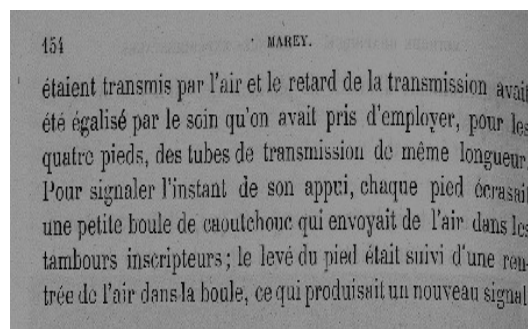


Image 4

Figure 3.7 – Portions of the four sample document images among the 25 used in the OCR test

The recognition rates for each of the images are given in Table 3.2.

Table 3.2 - Recognition rates achieved by the ABBYY OCR

Image	Total characters	Number of characters correctly recognized				
		NIBLACK	SAUVOLA	WOLF	FENG	NICK
Image 1	2012	0	1907	2003	1932	2002
Image 2	944	887	0	821	938	940
Image 3	241	239	233	233	230	238
Image 4	364	0	356	354	344	363
Other 22 Images	5896	2687	3850	4920	5460	5511
Total	9457	3813	6346	8331	8904	9054
Recg Rate		40.32	67.10	88.09	94.15	95.74

It can be observed from the analysis of first four images that ABBYY was not able to read characters from Image 1 & Image 4 binarized with Niblack's method, and Image 2 binarized with Sauvola's method, as it classified them as figures due to bad binarization. With Niblack's method, it was a case of extreme noise in the binary image that prevented ABBYY from detecting any text, while with Sauvola's method; it was due to very thin and broken characters, which as a matter of fact, were impossible to read even from the naked eye. With Wolf's method, binarization was better in most cases. In image 2 though, there were a lot of broken characters which the OCR could not read. With Feng's formula, OCR results were better than with Wolf's method, though in Feng's images, there were bit more noisy patches, which affected the OCR performance in some images. With NICK, ABBYY was always able to distinguish and read text areas, indicating the effectiveness and robustness of NICK for different illumination and intensity variations in document images. It can be noted that though our method does not always correspond to the best OCR performance for each image, it has no strong weaknesses. Thus in each case it is able to provide a binary image that the OCR reads as text even if the original image quality was very poor. Based on this test set, NICK achieves a recognition rate of 95.74% better than all the other methods tested here.

The algorithm was also submitted to the 2009 ICDAR document binarization competition. The performance of the methods was evaluated on ancient printed documents as well as historical handwritten document images. Figure 3.8 below shows the F-score graph for the different methods in the competition. Out of the 43 binarization algorithms, NICK achieved the 16th best F-score of all. It can be noticed that after the top three methods, the performance of a group of methods, apart from the last 9 odd methods, is very comparable. It is worth mentioning that other methods are complex multi-stage algorithms while NICK is a very simple thresholding formula which achieves comparable results to these multi-stage complex methods. Details of the competition can be consulted in [Gatos *et al.* 2009].

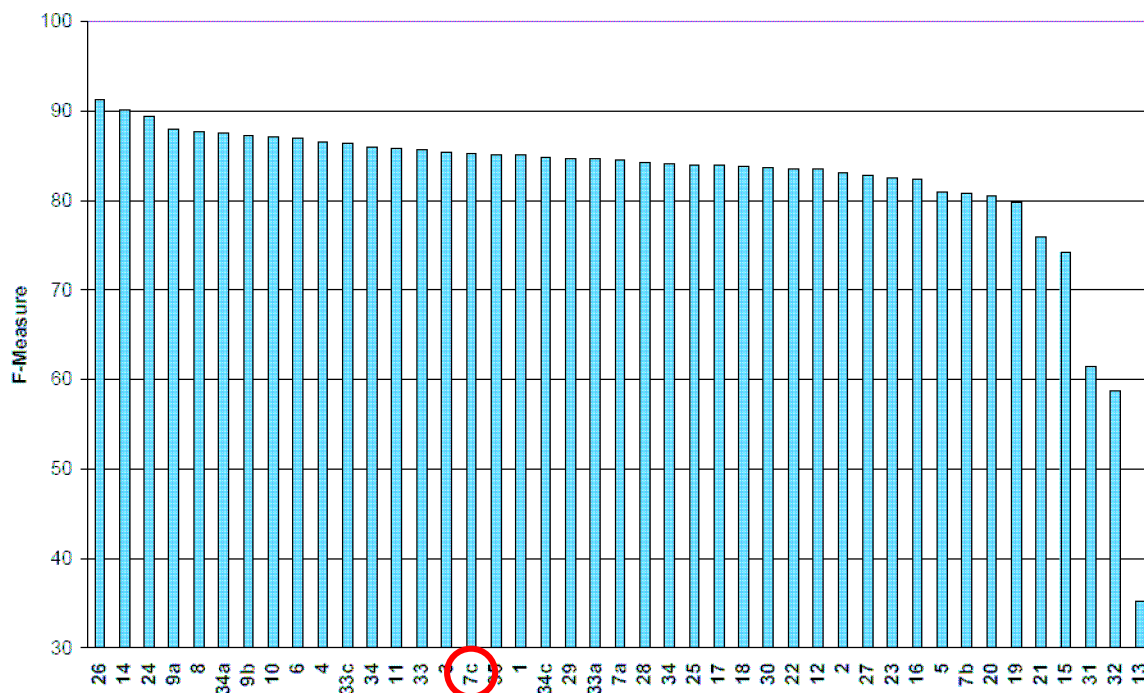


Figure 3.8 - F-scores of 43 different methods in the ICDAR 2009 competition – NICK (7c) stands at 16th position

Now that we have thoroughly discussed the binarization of document images, we can proceed to extraction of words and characters which is the second main stage of our indexing process.

3.2 Text/ Graphic segmentation and Extraction of Words

A fine segmentation of words and characters in ancient document images is very useful for applications such as information retrieval using word spotting or optical character recognition. The degraded quality of these ancient documents poses different problems such as characters broken into multiple components, text of the verso appearing on the recto, etc., thus making the crisp extraction of the words and characters very difficult [Antonacopoulos *et al.* 2004], [Baird2004]. Some earlier approaches to segmentation required knowledge of the character size in order to separate a document into text and non-text areas. But if the size of characters varies in the same page or in different pages of the same book, then the approach will not work well.

As already shown in chapter 2 (State of the art), traditionally page segmentation methods are divided in three groups: top-down, bottom-up and hybrid approaches [Okun *et al.* 1999], [Duong *et al.* 2001], [Shi and Govindaraju2005]. In top-down techniques, documents are recursively divided from entire images to smaller regions. These techniques are often fast, but the efficiency depends on a priori knowledge about the class of documents to be processed. Bottom-up methods

start with the thinnest elements (pixels), merging them recursively in connected components or regions, and then in larger structures. They are more flexible but may suffer from accumulation of errors. Many other methods that do not fit into either of these categories are therefore called hybrid methods.

To segment text and graphics and extract words from the text, we have formulated a multi-step bottom-up approach by using the classic Run Length Smoothing Algorithm (RLSA) in horizontal direction followed by a connected component area analysis in the RLSA image. The main advantage here is that we do not need to know *a priori* the character size to segment text and non-text areas. We now see it in detail.

3.2.1 Run Length Smoothing Algorithm

RLSA has been used previously in text/non-text segmentation. [Wong *et al.* 1982] used a combination of RLSA in horizontal and vertical direction to segment blocks of text and non-text. Afterwards, the text blocks are analyzed for the extraction of words. In our case, instead of segmenting the document image into blocks of text and non-text, we segment the image directly into words and graphics. For that we apply RLSA only in horizontal direction. The basic RLSA is applied to a binary sequence in which white pixels are represented by 0's and black pixels by 1's. The algorithm transforms binary 'x' into an output 'y' according to the following rules:

1. 0's in x are changed to 1's in y if the number of adjacent 0's between two 1's is less than or equal to a predefined limit C .
2. 1's in x are unchanged in y.

For example, with $C = 4$ the sequence x is mapped into y as follows:

```
x : 0 0 0 1 0 0 0 0 0 1 0 1 0 0 0 0 1 0 0 0 0 0 0 0 1 1 0 0 1
y : 0 0 0 1 0 0 0 0 0 1 1 1 1 1 1 1 0 0 0 0 0 0 0 1 1 1 1 1
```

When applied to sequence arrays of 0's and 1's, the RLSA has the effect of linking together the neighbouring black areas that are separated by less than or equal to C pixels. With an appropriate choice of C , the linked areas will be regions of a common data type. The degree of linkage depends on C , the distribution of white and black in the document, and the scanning resolution. For a set of document images, the value of the RLSA threshold C is set according to the average distance between the connected components in the binary document image (Figure 3.9). For most of the historical books that we used, the usual distance between the words varies between 6 and

14 pixels with most values concentrated around 10-12. An optimal value of C comes out to be 9 for most of the books which we use in our experiments. If the distance between two neighbouring components is less than 9, it means they belong to a same word and thus are merged by H-RLSA.

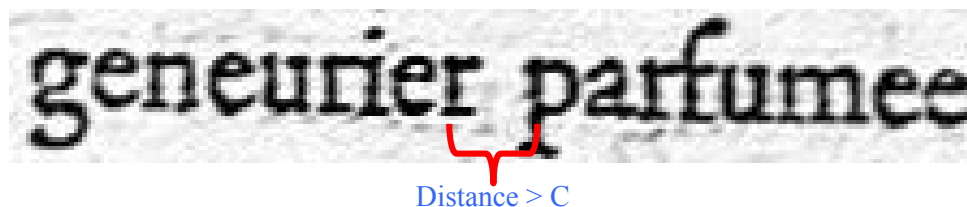


Figure 3.9 – Distance between two words must be greater than RLSA threshold

Figure 3.10 shows H-RLSA on small portions of a document image of 19th century. Figure 3.11 shows H-RLSA on two complete document images.

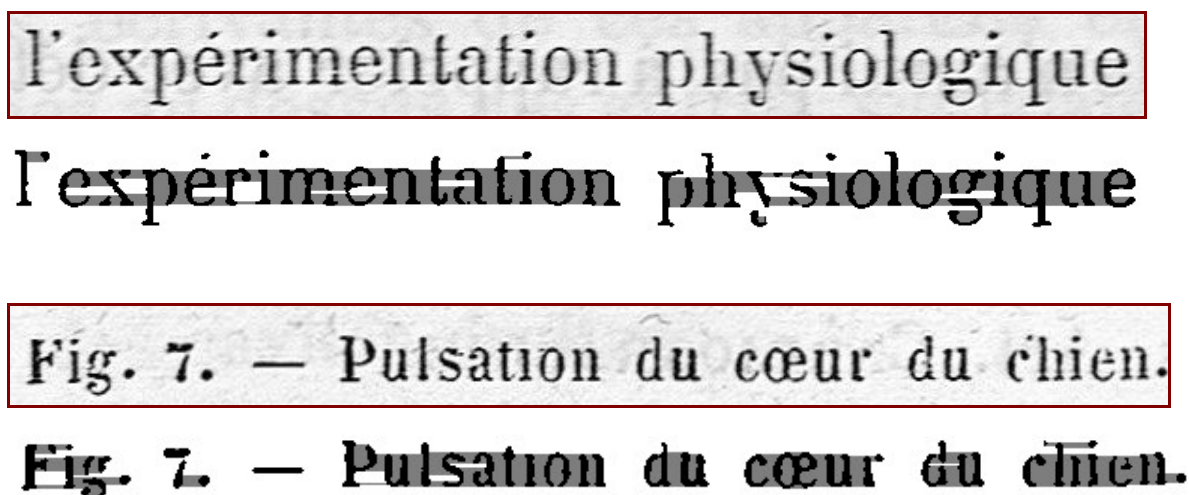


Figure 3.10 – Applying H-RLSA on portions of an image with $C = 9$

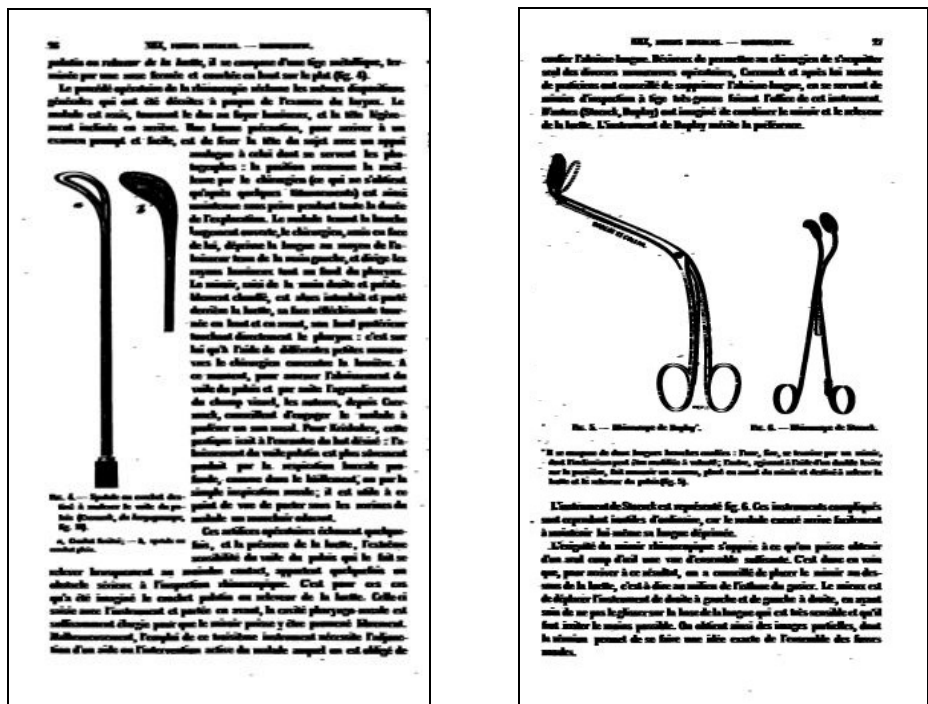


Figure 3.11 - H-RLSA on complete document images

H-RLSA has a similar effect as of dilation of black areas in horizontal direction. The characters in a word are dilated and get linked/connected to the other characters of the same word. The distance between two neighboring characters of two adjacent words is greater than the value of C , so that gap remains there meaning that each word becomes a connected component. Similarly, the graphics in the document image get dilated and they form a connected component as well. Area and height analysis of the bounding boxes of the connected components is applied to segment words and graphics. It is explained in the next sub-section.

3.2.2 Component Height-Area Analysis

Once we have the H-RLSA image, we find all the connected components present in that image. The components in the H-RLSA image also contain graphics as large components. So to segment graphics from words, we perform an analysis of the area and height of the bounding boxes of all the connected components the image. Figure 3.12 shows distribution of area and height of all the components' bounding boxes in the H-RLSA image.

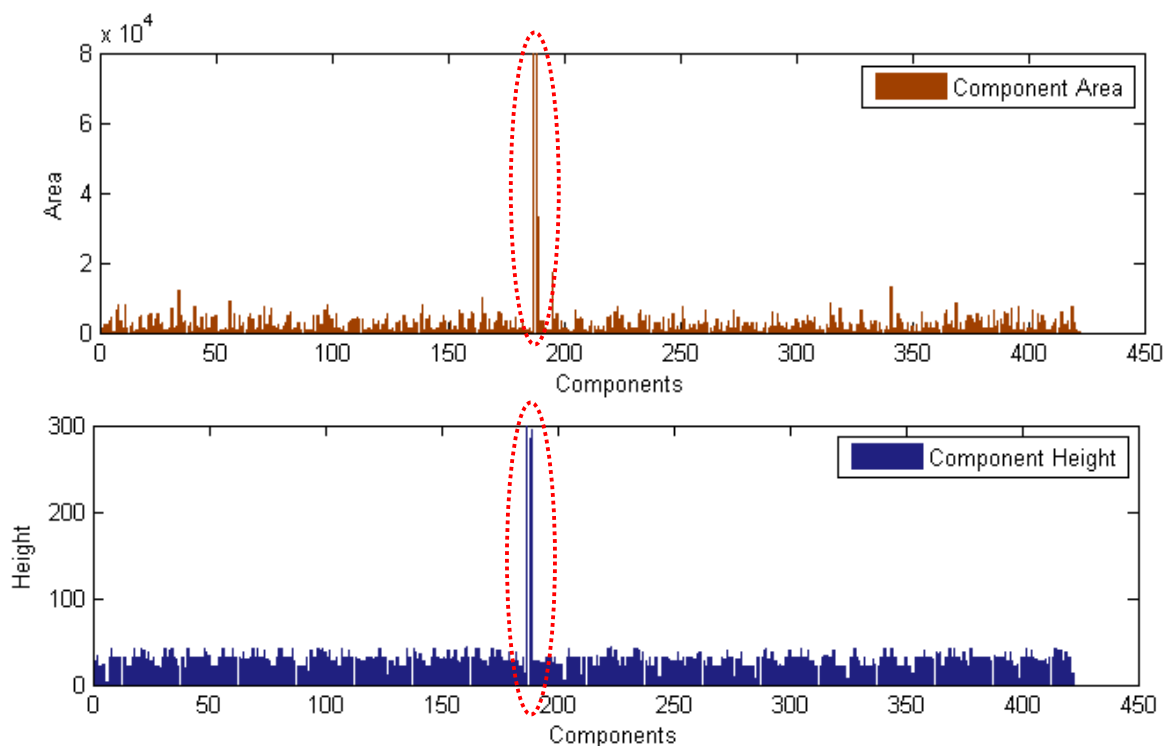
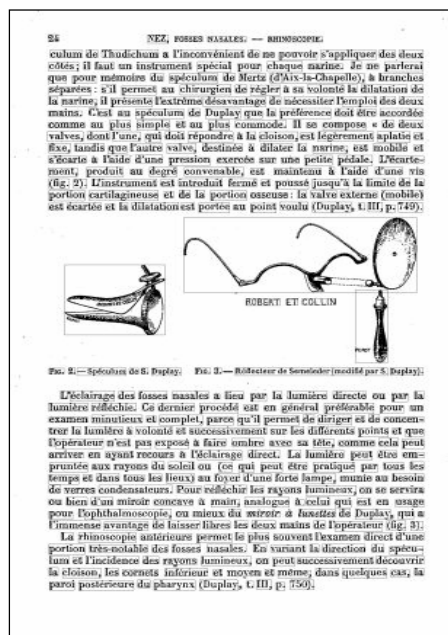


Figure 3.12 - Areas and heights of all the bounding boxes of the 425 components in the document above

From the graphs in Figure 3.12, we can see that there are three components for which the areas and heights of their bounding boxes are significantly greater than others. We have set the following criteria for these larger components to check if they are graphic components or not:

Graphic Components = [Component Area > (Mean comp. area x A)

AND

Component height > (Mean comp. height x B)]

where mean component area is the average of areas of the bounding boxes of all the components in that particular document image while mean component height is the average height of the bounding boxes of all the components in that document page. A and B are constants which are empirically found to be 5 and 4 respectively through thorough experimentation on different documents of BIUM. The results show that this method works for almost all types of documents for the separation of text from illustrations. Figure 3.13 shows the result of component area analysis for some images. The larger components represent figures (components in red) while smaller components represent words (blue components).

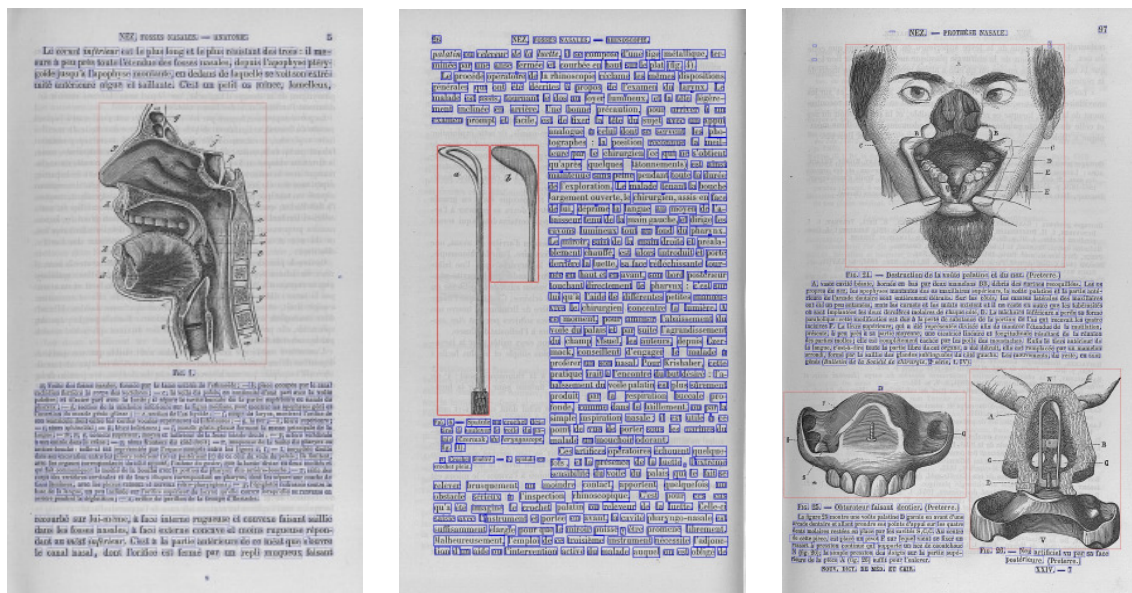


Figure 3.13 - Larger components are classified as graphics (red) and smaller ones as words (blue)

The very small components (having an area of less than 10 pixels) are marked as noise and are removed at this stage. Similarly, the small sub-components inside another component's bounding

box are removed as well, though this might not be efficient to do all the time, especially if we have documents where small text characters are written inside the figure's bounding boxes. Figure 3.14 illustrates the images at different stages of word extraction for a small gray image.

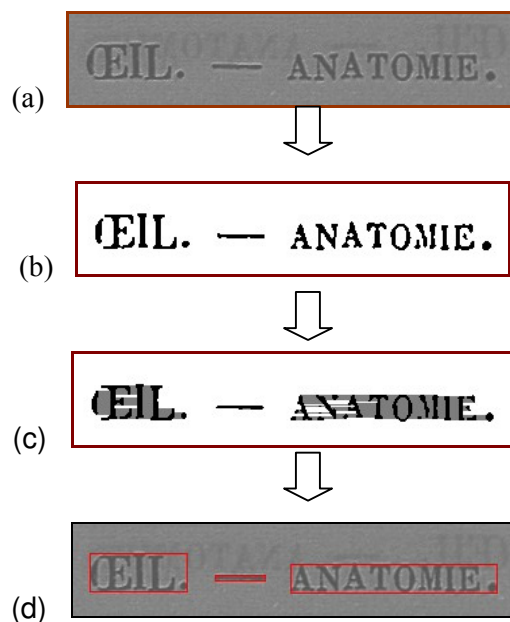


Figure 3.14 - Stages of word extraction a) Original image b) Binary image c) H-RLSA image d) Connected components (words)

The results of the word extraction are formulated on the data set B containing 48 document images from 12 different books. There are a total of 17,010 words in these pages and we are able to perfectly segment 16,970 words which is 99.76%. The words which aren't perfectly detected are mostly the title words written in very large fonts. For example, in Figure 3.15, the word "CHAPITRE" hasn't been extracted as one word but is broken into small sub words as the RLSA threshold is not enough to merge the characters in one component. These title words can be treated separately from the remainder of the text and words could be extracted using a larger RLSA threshold.

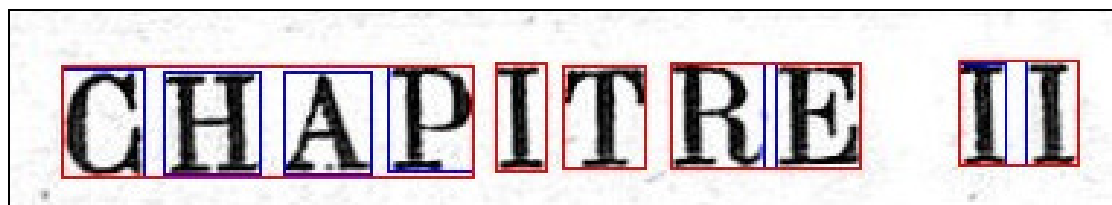


Figure 3.15 – Title word "CHAPITRE" is not extracted correctly

As our ultimate goal is a character-feature-based retrieval system, so once we get all the words in a document image, we move on to the next stage where we try to extract characters for each of the words. Following section describes the character extraction process.

3.3 Character segmentation

Over the years, different types of methods have been proposed for character segmentation in document images. These include the recognition based methods and the dissection based methods. In *recognition based* methods, word image is divided systematically into many overlapping pieces without regard to the content. These pieces are then built into characters during the recognition stage by matching with the already known characters using different methods [Jung *et al.* 1999], [Burges *et al.* 1992], [Casey and Nagy1982]. The problem with this type of segmentation is that we need to have a knowledge of characters in advance, which is sometimes not possible in ancient documents due to many variations of writing printing font styles. On the other hand, in *dissection based methods*, the word image is cut into meaningful components based on ‘character-like’ properties. These properties include height, width, separation from neighbouring components, disposition along a base line, etc. This category includes methods such as classic projection methods [Hoffman and McCullough1971] and connected component analysis methods [Cesar and Shinghal1990]. An experimental comparison of character segmentation by projection analysis vs. segmentation by connected components is given in [Wilkinson1992] showing that connected component based segmentation easily outperforms the projection based segmentation. A detailed survey of different methods and strategies in character segmentation in [Casey and Lecolinet1996] can be consulted for more information on the subject.

For character segmentation, we follow a connected component based approach. As we know that a word is composed of characters, and in an ideal case, the connected components in the word should be the characters so we carry out a connected component analysis on the extracted word images to get raw characters which we term as segmented characters or in short, S-characters. However, an S-character does not always correspond to an actual/true character (T-character). A T-character may be broken into multiple S-characters or multiple T-characters may form a single S-character. Also there are characters like i, j, é, etc. which are composed of more than one component. So once we get the S-characters, we need to have a post-processing stage in which the bounding boxes of S-characters are joined or split according to rules based on heights, widths and positions of their bounding boxes, so that the finally extracted S-characters could correspond to the T-characters. Below, we describe our 3 pass method for the fixation of the raw S-characters to get S-characters that correspond in a better way to the T-characters.

3.3.1 Post-processing for character extraction

A 3-pass post-processing method is applied to the bounding boxes of the S-characters of a word to get the actual characters.

3.3.1.1 Pass-1 : Multi-component characters

In the first pass, we analyze the T-characters which are composed of two or may be more S-characters on top of each other. These include T-characters like i, j é, etc., which are composed of two S-characters. We merge the bounding boxes of these S-characters into one bounding box enclosing the whole character. The coordinates of the bounding boxes are measured in accordance to the coordinate reference used in computer vision where top left corner of the screen corresponds to (0, 0) and bottom right corner corresponds to (Xmax, Ymax). The first post-processing pass is implemented as follows:

For bounding boxes of two neighboring S-characters A and B

If A.xmin is less than or equal to B.xmin **AND** A.xmax is greater than or equal to B.xmax **then**

A.ymin = **min** (A.ymin , B.ymin)

A.ymax = **max** (A.ymax, B.ymax)

Delete component B

The coordinates ymin and ymax of the larger S-character's bounding box A are adjusted. The height of A is increased so that it includes the S-character B within its bounding box, while the smaller S-character's bounding box (B) is deleted from the list of S-characters of the word. It is graphically shown in Figure 3.16:

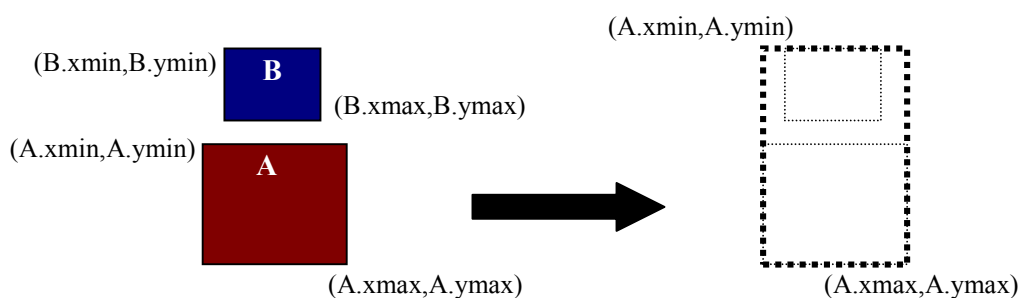


Figure 3.16 - Concatenating the bounding boxes of two S-characters on top of each other

Figure 3.17 shows images of words from relatively good quality and resolution document images in the data set (so that the post-processing steps could be clearer to understand). In these words, some of the T-characters are composed of two S-characters which are treated in this pass.

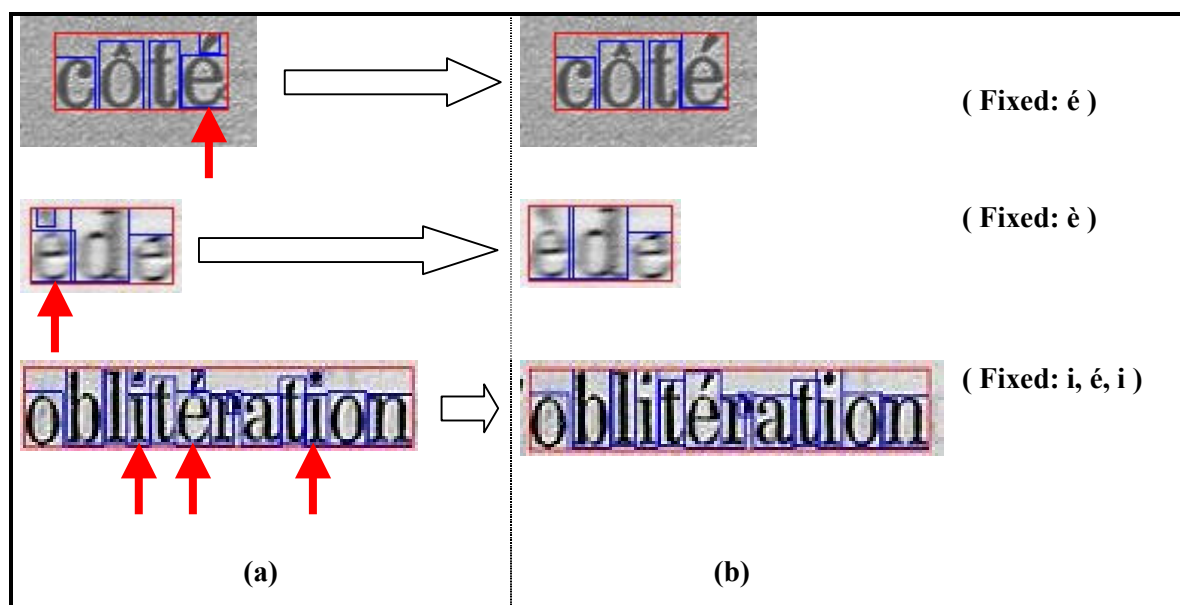


Figure 3.17 – Example of pass-1, a) Raw S-characters b) S-characters after Pass 1

3.3.1.2 Pass-2 : Broken characters

In the 2nd pass, we analyze the T-characters which are broken into multiple S-characters due to quality issues in the scanned document images. These include T-characters like r, g, m, etc. We merge the bounding boxes of the two S-characters into one bounding box enclosing the complete character as:

For bounding boxes of two neighboring S-characters A and B

If (B.xmin is less than A.xmax) **AND** (B.xmax - A.xmax) < T **then**

A.ymin = *min* (A.ymin , B.ymin)

A.ymax = *max* (A.ymax, B.ymax)

A.xmax = B.xmax

Delete component B

It works like this: we look for those S-characters in a word that overlap at some point. The coordinates ymin, ymax and xmax of the S-character A are adjusted so that it includes S-character

B within its bounding box, while the bounding box of the S-character B is deleted from the list of S-characters of the word. It is graphically shown in Figure 3.18.

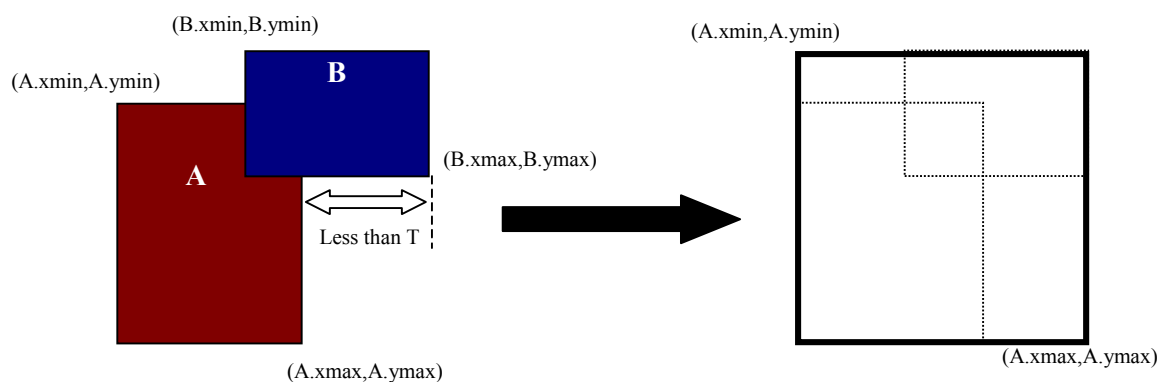


Figure 3.18 - Concatenating two S-characters into one S-character

Figure 3.19 shows a word after pass 1 containing character ‘r’ broken into two S-characters; it is fixed in this pass 2.

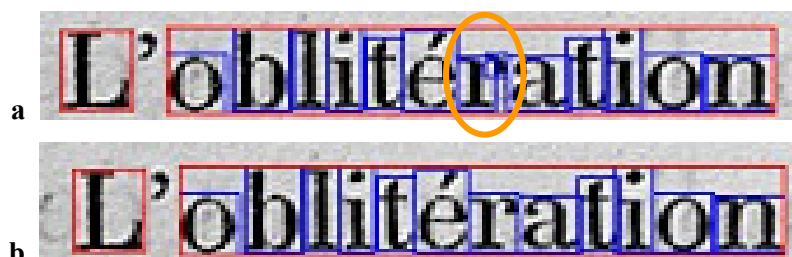


Figure 3.19 - a) S-characters after pass1 b) S-characters after Pass 2

We have set a bound on the difference of B.xmax and A.xmax to eliminate certain true positive cases getting the same “merge” treatment. It is usually the case if the text is written in *italic*, bounding boxes of different S-characters could overlap at certain points. Figure 3.20 shows a couple of word instances in italic with bounding boxes of some S-characters overlapping. If we do not have this “merge” bound, these S-character’s bounding boxes would be combined to one big bounding box.

The value of this limit is set by taking into account the average width of all the S-characters in the page and then dividing the average value into half.

$$T = \text{Average Width} / 2$$

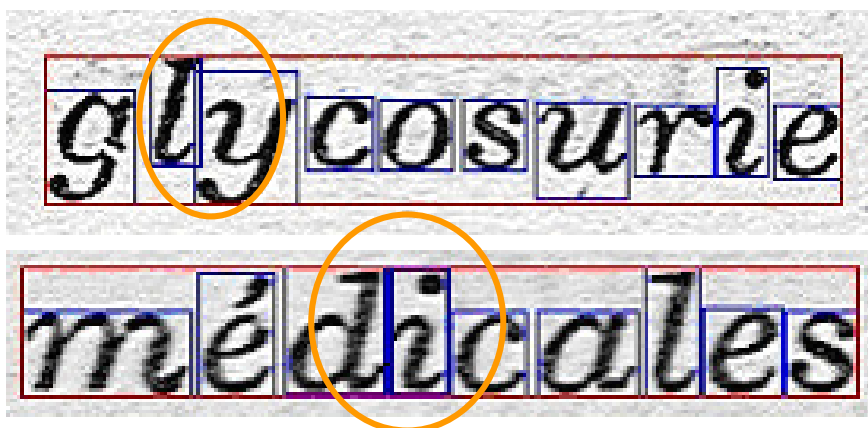


Figure 3.20 - Examples of bounding boxes overlapping in *italic* font

This limit leaves a few broken T-characters unfixed, mainly because the width of the S-character B is more than this limit criterion, but the number of these instances is very low. There are some other unfixed T-characters as well which are broken into two S-characters but the bounding boxes of these S-characters do not overlap at any point. So they cannot be merged in this way. Examples of the unfixed T-characters are shown in Figure 3.21:



Figure 3.21 - Examples of T-characters broken into two S-characters

Apart from these broken T-characters, there are some instances of merged T-characters as well. Figure 3.22 shows a couple of examples of these merged T-characters. These broken and merged T-characters remain non-treated here but we take them into account later on as we tackle the broken and merged T-characters in our Merge-Split Edit distance. It will be discussed in detail in the next chapter.

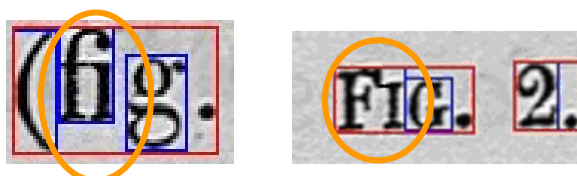


Figure 3.22 - Examples of S-characters comprising two T-characters

3.3.1.3 Pass-3 : Extra Component removal

S-characters that do not belong to a word but are segmented as a part of the word are handled in this pass. These S-characters mainly comprise punctuation marks (like ‘,’ or ‘.’) which are very close to the word they follow and thus are considered, incorrectly, as an S-character of the word. Figure 3.23 shows examples of words having these extra S-characters as their part.



Figure 3.23 - Removal of punctuation marks

To remove these S-characters from the word, we follow a simple component area approach where we find average S-character area for that word (using the areas of the bounding boxes of all S-characters in the word) and then exclude the S-character whose bounding box's area is less than a factor of the average S-character area. Mathematically,

$$\text{S-character area} < A \times \text{Average S-character area of the word}$$

Though it is very hard to estimate a perfect value of A , as for different word lengths the average area may vary drastically, still we found (empirically) that the value for which this approach works the best for all the images of our validation set comes out to be 0.4. This value makes sure that no actual T-character gets excluded from the word. Punctuation marks in the word component and also other small noisy S-characters are marked in this pass so that they no longer constitute a part of the word.

3.3.2 Evaluation of Character Segmentation method

After the three passes of S-character reconstruction/fixation, we get S-characters which are very close to T-characters. But how many of these S-characters are T-characters and how many do not represent a T-character? To evaluate this, whole character segmentation process is tested on data set B comprising 48 different document images taken from 12 varied 19th century books of the BIUM library. Total number of T-characters in these 48 images is 82,264. Evaluation of character segmentation process is done by using Recall and Precision percentages. In our context here, recall is defined as:

$$R = \frac{\text{Number of T-characters Extracted}}{\text{Total Number of T-characters}} \times 100$$

And precision is defined as:

$$P = \frac{\text{Number of T-characters Extracted}}{\text{Total Number of S-characters}} \times 100$$

Table 3.3 summarises the overall results for different stages of character processing. Final T-character recall percentages (after post-processing) for each document image are depicted graphically here in Figure 3.24. We can see that for a couple of document images, the percentages drop to 92 and 94%. This is because, in these document images, the intensity variations in text and background are very small. That is why we get more binarization noise, causing the T-characters to break into S-characters. An example document image is shown in Figure 3.25. How we cater for these broken T-characters will be discussed in the word matching part in next chapter.

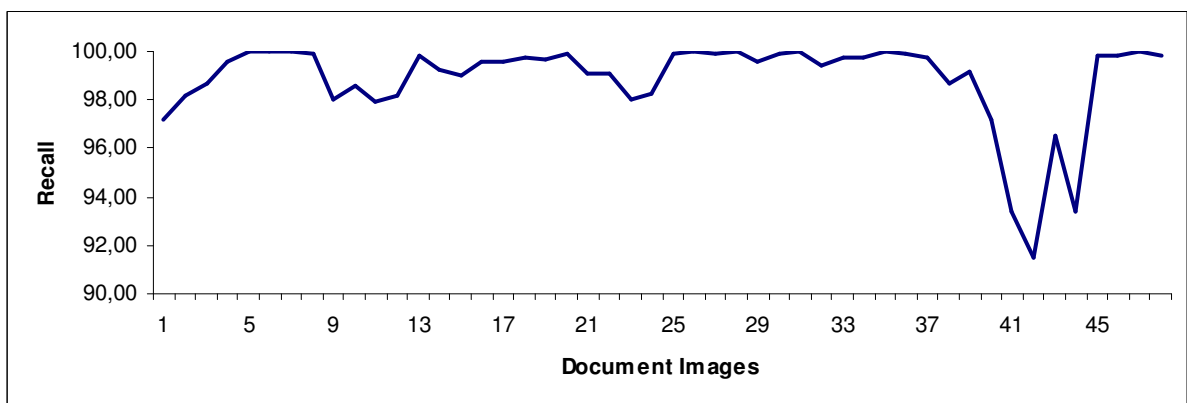


Figure 3.24 - Recall percentages for T-characters in each document image

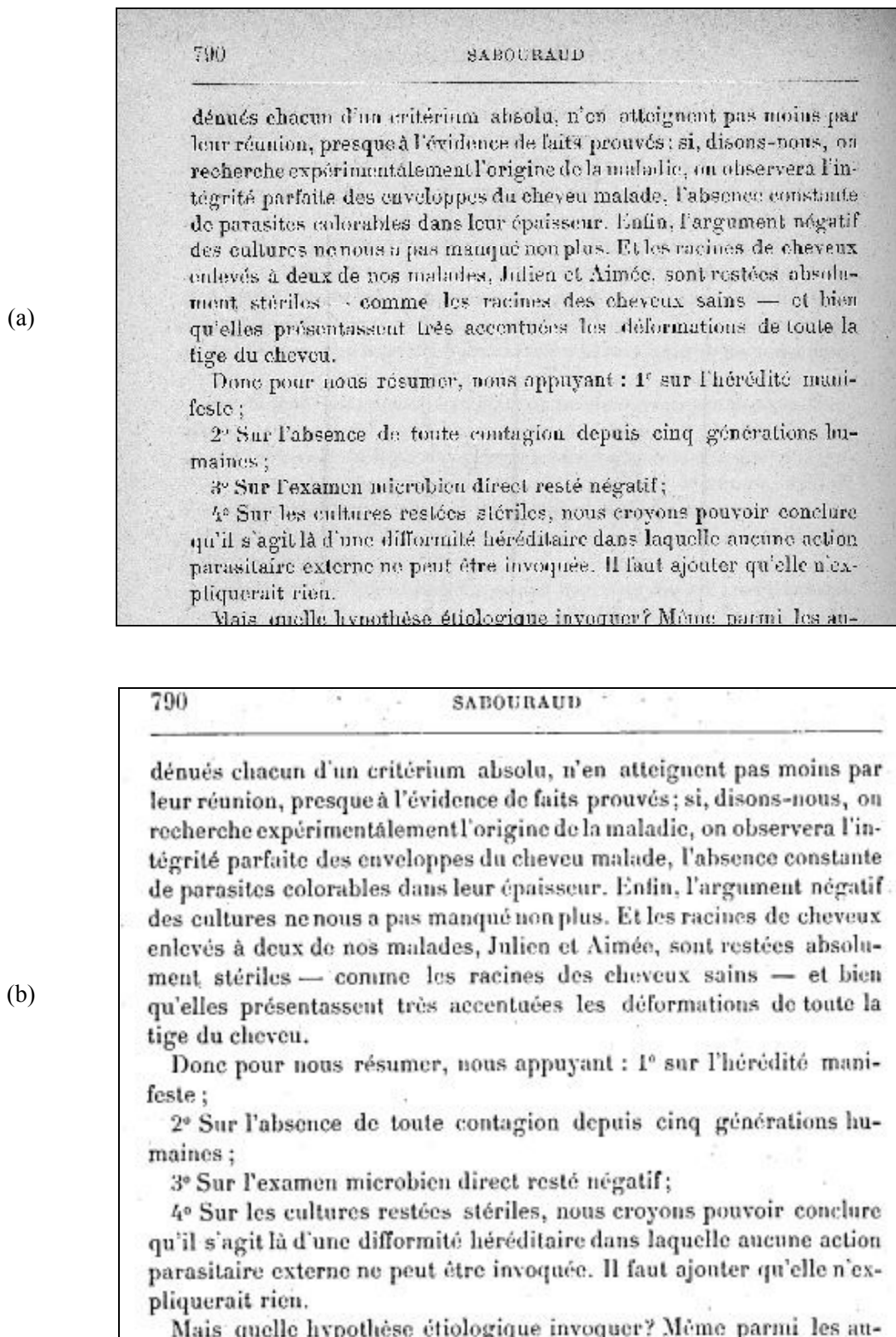


Figure 3.25 – a) A document image and b) its binarized image, where character segmentation is not so good

Table 3.3 - Result summary of character segmentation process

Before post processing passes	
# Total T-characters in the data set	82264
# of raw S-characters within words	115414
# of T-characters in these S-characters	60358
Recall %	73.4%
Precision %	52.3%
After pass 1 and 2	
# of S-characters treated (merged) during pass 1 and 2	20745
# of S-characters after pass 1 and pass 2	94669
# of T-characters in these S-characters	81103
Recall %	98.6%
Precision %	85.7%
After pass 3	
# of S-characters removed during pass 3	10244
# of S-characters after pass 3	84425
# of T-characters in these S-characters	81103
Recall %	98.6%
Precision %	96.1%

Without applying the 3 passes we got about 73% of the total T-characters in the data set. After applying the three passes, we were able to properly extract 98.6% of the T-characters. The remaining 1.4% T-characters which are not extracted correctly are either broken (as shown above in the examples) or there are few T-characters which are merged into a single S-character. We have not handled these merged T-characters here. These and the remaining broken T-characters will be handled by Merge-Split Edit distance method, explained in next chapter.

Now once we have the S-characters, we find certain features for each S-character for representation in feature domain. These features are used for matching two S-characters in the retrieval stage. Feature extraction is described in the following section.

3.4 Feature Extraction

To represent an S-character, we have defined a set of six feature sequences and five scalar characteristics. Four of these feature sequences (vertical projection, upper and lower profiles, ink-non-ink) are used in [Rath and Manmatha2007], for characterizing the word image as a whole.

Our approach of having these six feature sequences for character images gives a better representation of words in feature space as compared to word level features, as we will show later on in the results. Also having features at character level gives us more flexibility during the matching stage [Khurshid *et al.* 2009b], [Khurshid *et al.* 2008a]. The scalar features are used for coarse matching and decision making while the vector features are used in actual dynamic matching process. Features are described below.

3.4.1 Features

For each S-character, we find six feature sequences each having a length equal to the width of that particular S-character. It means that for different S-characters, the length of the sequences may be different depending on their widths. These feature sequences include vertical projection, upper character profile, lower character profile, ink-non-ink transitions, vertical histogram and mid row transitional vector. There might be some redundant information in these six features, but as we have seen, each one of them provides some unique information as well, as later depicted by individual feature results. Vertical projection has been calculated using the original gray image while the rest of the features have been found on the binarized image. Some of these features are known classically [Costa and Jr2001] and have been employed in a variety of applications. The profiles and histograms, for example, have shown good performance on character recognition [Heutte *et al.* 1998], writer identification [Siddiqi and Vincent2008] and font recognition [Zramdini and Ingold1993]. We describe the features in detail below.

3.4.1.1 Vertical Projection

It is the sum of intensity values in each pixel column of the gray scale S-character image. The values in the resulting sequence are normalized between 0 and 1 by dividing each of them by 255 times the height of the S-character's bounding box. Figure 3.26 shows the normalized vertical projection curve for the S-character image 'p' having width 23.

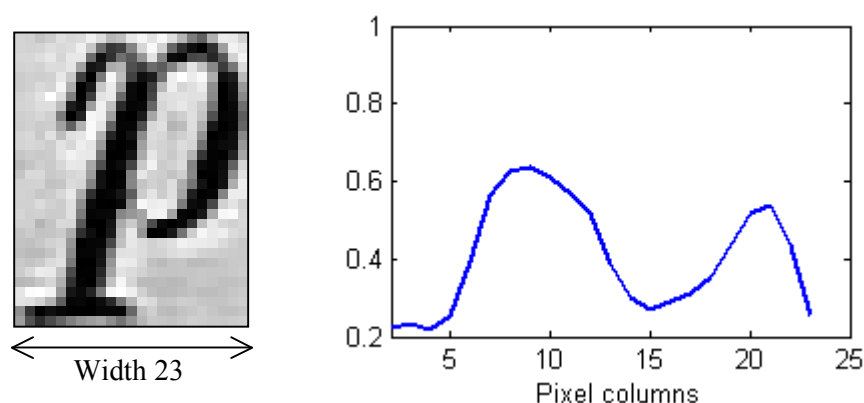


Figure 3.26 - Normalized vertical projection profile of character image 'p'

3.4.1.2 Upper Character Profile

For each column of the binarized S-character image, the distance of the first ink pixel from the top of bounding box is its upper profile. It captures the part of outlining shape of a character. Figure 3.27 shows upper profile of binary S-character image 'p'.

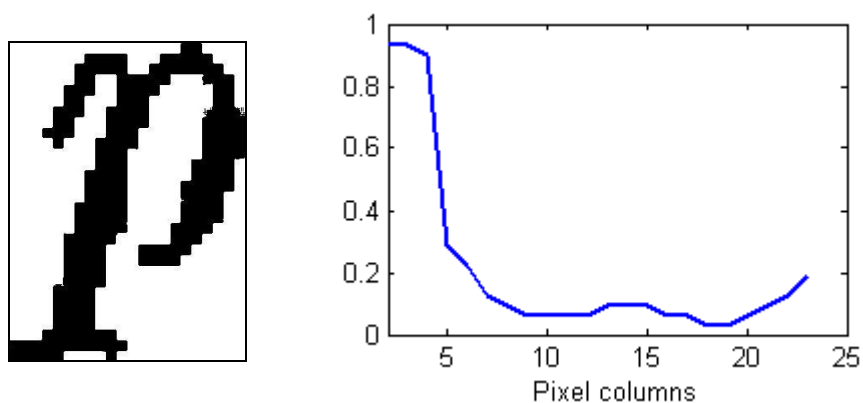


Figure 3.27 - Normalized Upper profile of S-character image 'p'

3.4.1.3 Lower Character Profile

For lower profile, we calculate the distance of the last ink pixel from the top of bounding box. Both upper and lower profiles are normalized between 0 and 1 by dividing each of them by the height of the S-character's bounding box. Figure 3.28 shows lower profile of S-character 'p'.

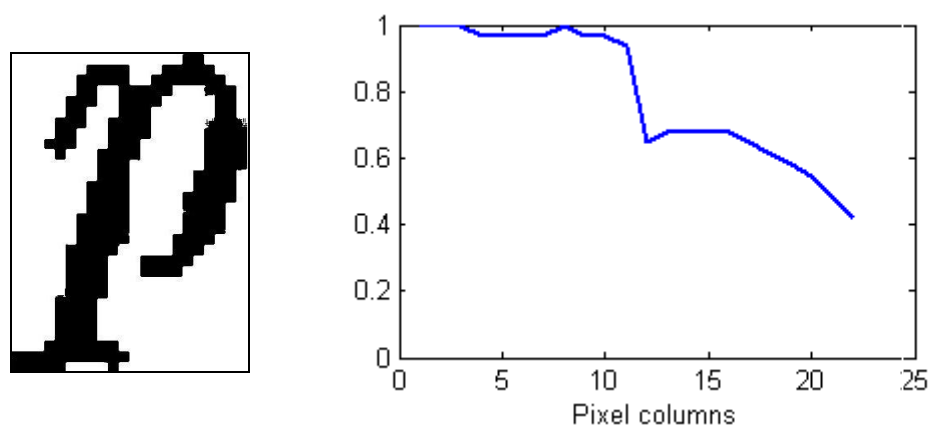


Figure 3.28 - Normalized lower profile of S-character image 'p'

3.4.1.4 Ink/ non-ink transitions

Both upper and lower profiles help to study the outer structure of an S-character. To capture the inner structure though, we consider an ink/non-ink transition feature. For each column of the binarized S-character image, we find the total number of ink to non-ink or non-ink to ink

transitions. The range of this feature is normalized using a (conservatively determined) constant that ensures a range of $[0 \dots 1]$. The value of that constant comes to be 6 as that is the maximum number of transitions we may have in any column of a Latin character image.

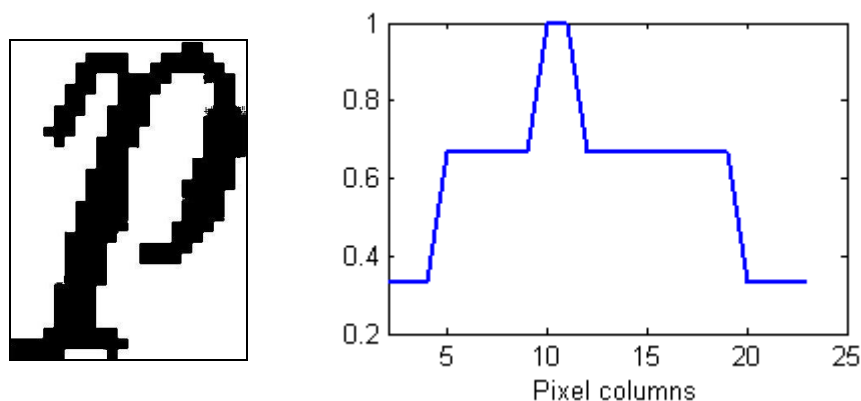


Figure 3.29 - Ink Non-ink transitions for each column of an image

3.4.1.5 Vertical Histogram

Number of ink pixels in each column of a binarized S-character image. Figure 3.30 shows the normalized vertical histogram of the binarized S-character image 'p'.

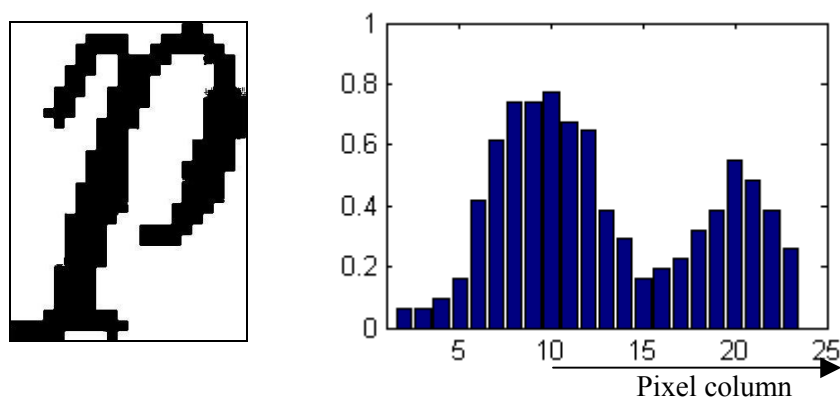


Figure 3.30 – Normalized Vertical histogram for the S-character image 'p'

3.4.1.6 Mid Row Transitional vector

For the central row of the S-character image, we find a transitional sequence accounting for all the ink/non-ink transitions. A '1' is placed for every transition from ink to non-ink or non-ink to ink, and a '0' for all the non-transitions in that row. This feature was also calculated in another way using three central rows instead of just one. The three central rows of the image were first merged to obtain a *mean row* using logical OR operator. It means that for a column, if there is an

ink pixel in any of the three central rows, we take it as an **ink pixel** for the same column in our mean row. Transitional vector is then calculated for this *mean row* using the same procedure.

We tested the feature both ways and found out that using only central row instead of the combination of three rows gives better results. This is because by merging three rows using OR operator, we might lose some transitions in between. Figure 3.31 shows it graphically. We can see that by using mean row, we would have only two transitions in our transitional sequence, while if we use only the central row, we would have 4 transitions which gives a better representation of the actual mid area in the S-character image. So we stuck to only the central row.

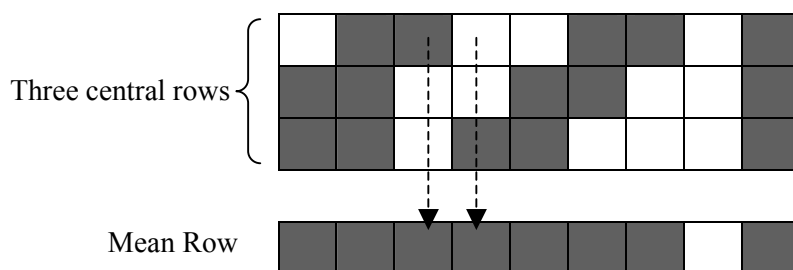


Figure 3.31 - Mean Row using three central rows of the image

Figure 3.32 shows the transitional sequences of two S-character images using only the central row.

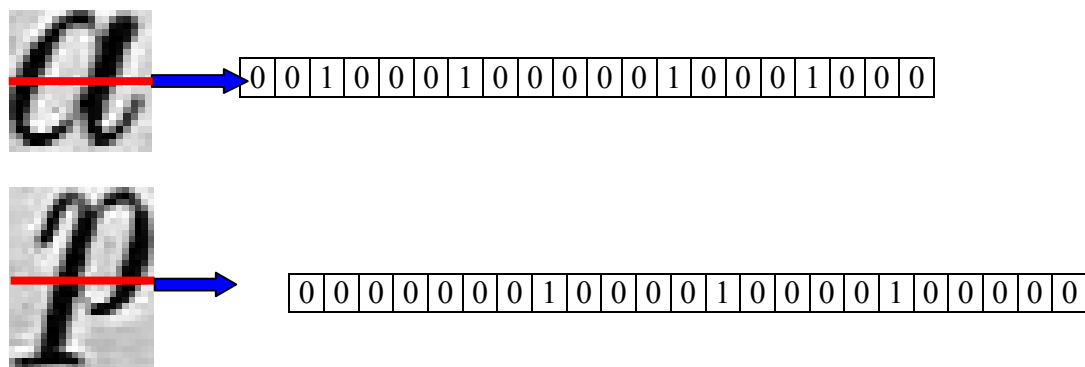


Figure 3.32 - Mid row ink to non-ink transitional sequence for two S-character images of widths 20 and 23

All these six features are used in word matching. Performance analysis of each of these features individually on our word matching algorithm is given in chapter 4 in the results section. To analyse the robustness and pertinence of the features and see if they are able to correctly distinguish the different S-characters, we performed automatic clustering of S-character images in a document image by matching these S-character features using DTW. Each cluster in ideal case

should contain all similar S-characters. Different clustering methods have been evaluated in [Marinai *et al.* 2008] for word images to perform word image indexing. The three methods are: the Self-Organizing Map (SOM), the Growing Hierarchical Self-Organizing Map (GHSOM), and the Spectral Clustering. In our case though, we have used the classic sequential clustering method [Friedman and Kandel1999] as it is the simplest, computationally efficient and the most natural way to cluster data once the number of classes is not known. This method has been employed for hand written text in [Nosary *et al.* 1999] for the clustering of graphemes and [Siddiqi and Vincent2008] in clustering of writing fragments.

The algorithm starts with the choice of a proximity threshold and the first element as the centroid of the first cluster. For each of the subsequent patterns, the similarity between the current element and each of the clusters is calculated. The element is then either attributed to the nearest cluster or, in case, it is not close enough to any of the clusters (with respect to the threshold), a new cluster is created.

For our implementation, the (dis)similarity between an element (an S-character image) S_i and a cluster C_j is calculated by employing Dynamic Time Warping (DTW) on the feature set between S_i and the mean (μ_j) of C_j . (DTW will be explained in detail later in section 4.5). Every time an element is added to a cluster, the mean of the cluster is also updated. The process is repeated until all the S-characters in the document have been assigned to clusters. Figure 3.33 shows the application of clustering algorithm on a document image. We can see that barring some erroneous elements, each cluster very much comprises of similar S-characters, which shows that the features we selected are robust and pertinent to the case of matching the S-character images using dynamic matching distance.

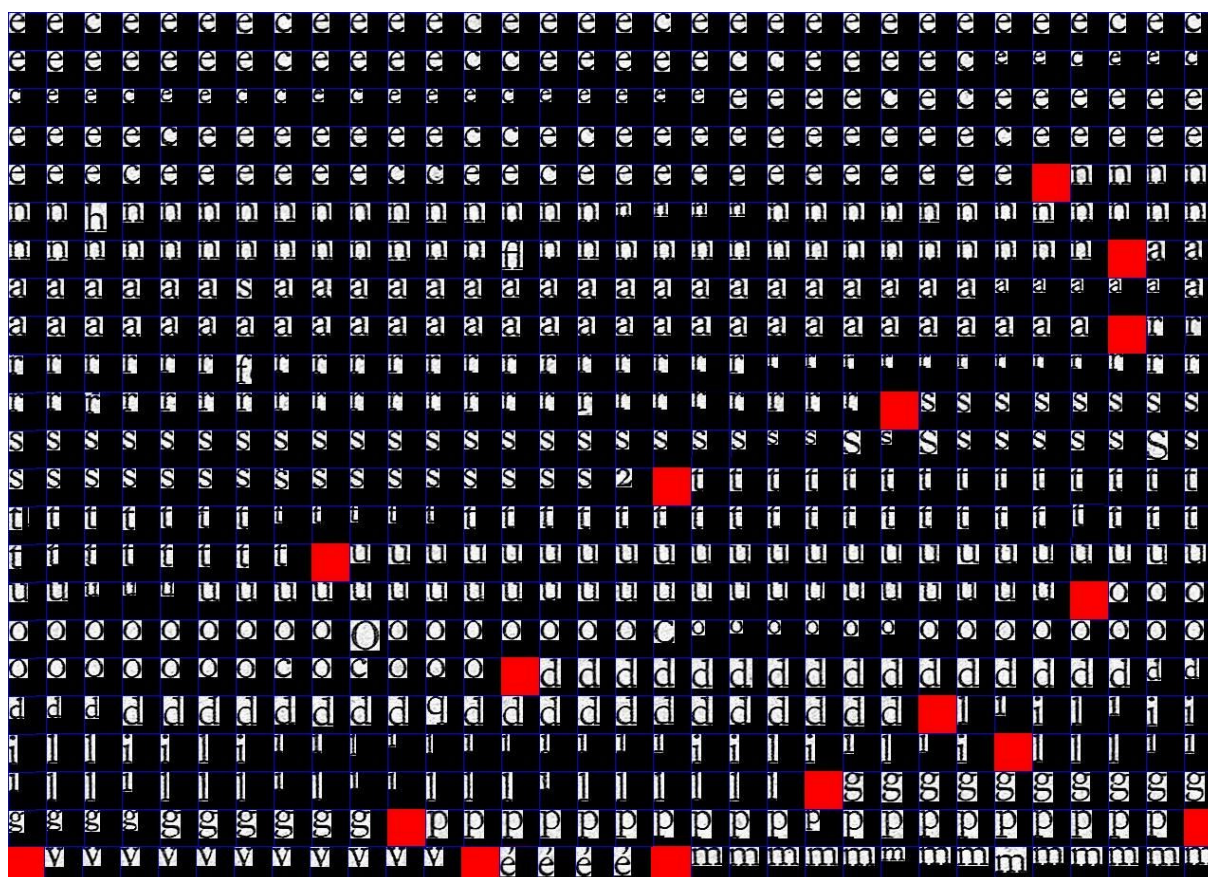


Figure 3.33 - Automatic clustering of S-characters in an image by matching the feature sequences using DTW - Red blocks represent the end of a cluster

From Figure 3.33, we can see that the matching using DTW is size independent and S-characters of different sizes are matched correctly using the above defined feature sequences.

Apart from the above six feature sequences, we find some scalar features for each S-character as well. These scalar features are there to be used in a coarse matching of S-characters. These are:

- a. Width:** Width of the bounding box (w)
- b. Height:** Height of the bounding box (h)
- c. Aspect Ratio:** The ratio of the width and height of the bounding box of an S-character (w / h)
- d. Area:** Area of the bounding box of the S-character ($w \times h$)

e. Character category: Horizontal histogram of the binary document image is most often used to identify the text lines in the document page. For a text line printed in Latin, most of the concentration of black pixels is along the top and base lines. This is because in Latin, every character touches the top and base line, but it is only for a few characters that some portion extends above the top line (ascender) or below the base line (descender). Examples of these characters could be h, p, f, etc. So, once we find the horizontal histogram of the text image, we can identify the text lines and their top and base lines. Figure below shows a text for which top and base lines are identified.

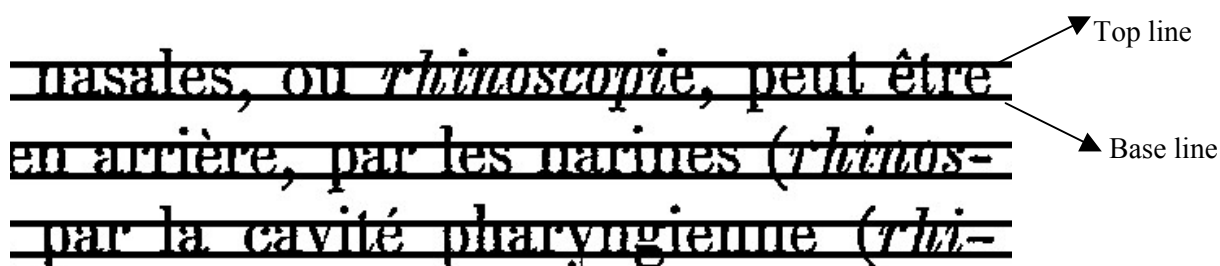


Figure 3.34 - Top and base lines drawn on the text lines

Once we identify the top and base lines, we can characterize the S-characters based on how they are written with respect to top and base lines. The S-characters which fit in between the top and base lines are put in category 1. Examples include a, c, o, etc. Similarly, those S-characters for which some of their portion extends above the top line but none below the base line are classified as category 2. Examples could be l, t, i, etc. Category 3 comprises S-characters for which some portion extends below the base line but non above the top line. Examples are g, p, etc. Lastly, the category 4 consists of S-characters which extend both above and below the top and base lines respectively. Figure 3.35 shows this categorization.

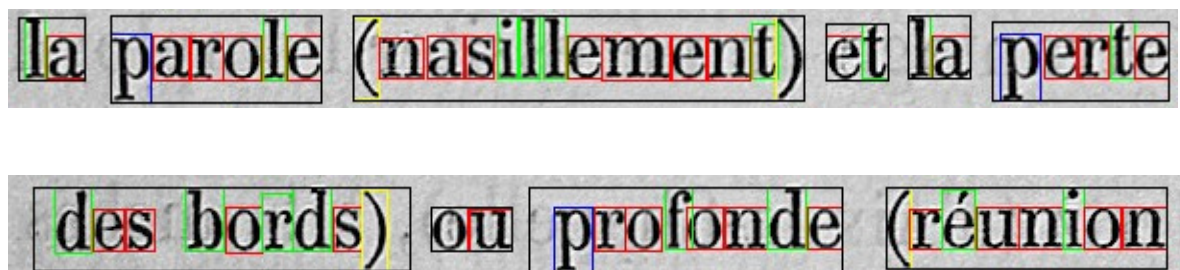


Figure 3.35 – S-characters of category 1 in red, 2 in green, 3 in blue and 4 in yellow

3.5 Image Indexing

Once all the processing is done and features are found for each S-character in the document image, an index file is generated for each image. Following data is stored in the index files for the purpose of word matching:

1. Position of each word
2. Width and height of the word's bounding box
3. Number of S-characters in the word
4. Position of each S-character in the document image
5. Width/height of the S-character's bounding box
6. Features of each S-character

Index files are created offline once. Format of the index files is kept as '*.DAT*'. Size of these data files (index files) depends upon the document image itself. For large document images (in size/resolution) the size of the index file will be large. This is because in larger images, S-character widths will be large, implying that the length of all the feature sequences will also be large as it depends on the width of S-character's bounding box. Normally, in our case, the sizes of the index files range from 450Kb to 1.75Mb for different BIUM document images.

Similarly, computational time to generate an index file depends upon the document image as well. If an image has plenty of words and S-characters, it takes long to do all the processing and finding features for all these S-characters as compared to the document images with less words. To evaluate index time for different images, we tested 48 different document images of the data set B taken from 12 varied 19th century books on an Intel core2duo 2.1GHz machine with 3GB RAM. The time for indexing different document images is shown in Figure 3.36.

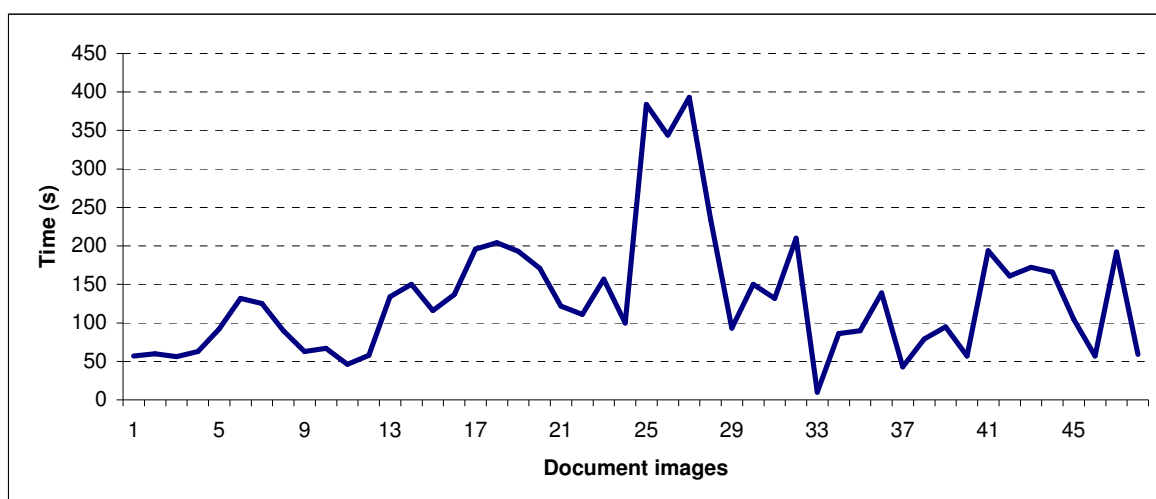


Figure 3.36 - Indexing time for different document images

We can see that indexing time varies for different document images depending upon different factors, such as, number of graphics in the image, total number of words and S-characters, font size, image resolution/size, etc. Overall, the average indexing time for an image in this document base is 130s.

3.6 Conclusion

In this chapter we discussed the document image indexing process in detail. For indexing, we divided the whole indexing process into four sub-steps. The first step is the preprocessing stage involving an optimized document binarization algorithm NICK to crisply distinguish the foreground from the background. Second step involves the word/graphic segmentation. Third step revolves around the connected component analysis of words for the extraction of S-characters which are post-processed using a 3-step process to get better S-characters. In the last step, multidimensional features are defined for each S-character of a word and are stored along with some other relevant information in an index file. Thus for each document image, an index file is created.

Now after indexing, once a query word is given to the system, the S-characters features are used to compare all stored words with the given query to retrieve all relevant words and pages. Whole of this matching and retrieval process will be discussed in detail in next chapter.

Chapter 4

Information Retrieval - Word Spotting

In the previous chapter, we saw document image indexing involving segmentation of words and figures and the extraction of S-characters for each word. Six feature sequences are defined for each S-character and are stored in data (index) files along with other information related to words and S-characters. In this chapter, we will talk about information retrieval from these indexed files by introducing a query word and retrieving all instances of words similar to the given query using multi-stage word spotting algorithms.

The chapter starts with a brief introduction of word spotting in general and also overview our methodology. This will be followed by the description of the data set used in experimental validation of the individual stages. Next we describe different ways for query formation followed by matching at word and S-character levels. We will compare the results of different word spotting algorithms proposed in the results section.

4.1 Introduction

Word spotting is a technique used for information retrieval from ancient documents using word matching as well as for creating partial indexes for historical document collections similar to indexes that can be found in the back of books. It was initially proposed by Manmatha in [Manmatha *et al.* 1996a] and [Manmatha *et al.* 1996b] and has prompted a number of publications that propose algorithms and features for the approach [Rath and Manmatha2007], [Adamek *et al.* 2007], [Rothfeder *et al.* 2003], [Rath and Manmatha2003]. The idea of word spotting is to use image matching for calculating pair-wise “distances” between word images, which can be used to cluster all similar words, occurring in a collection of documents, together. Ideally, each cluster would contain all the words with a particular label corresponding to an entry in the dictionary. The labels of the words are not known *a priori*. Then “interesting” clusters may be labeled manually and an index for the clustered collection may be built, similar to indexes in the back of books.

We use word spotting for the purpose of information retrieval from a collection of documents. The idea is to retrieve all the relevant document images which contain words similar to the input query word. The query word image is compared to all the candidate words using image matching in order to find all similar words in all the indexed documents. With this ability to search in the

historical document images with acceptable results, digital libraries will further enhance their importance in the field of research.

4.1.1 Overview – basic idea

For word spotting, we have proposed a two step retrieval system. In the first step, we narrow down the data set to be searched by using a *length-ratio filter*. It finds all eligible word candidates which could be similar to the query word. For two words to be considered eligible for matching, we have set bounds on the ratio of their lengths. If this ratio does not lie within a specific interval, we do not consider the word as a candidate for the query word. In the second step, the query word and candidate words are matched using a multi-stage retrieval system in which two words are matched using different string comparison algorithms while two S-characters are matched using an elastic DTW approach coupled with Euclidean distance [Khurshid *et al.* 2009b]. Figure 4.1 shows the main blocks of our retrieval system. Details of each of these blocks follow in the subsequent sections.

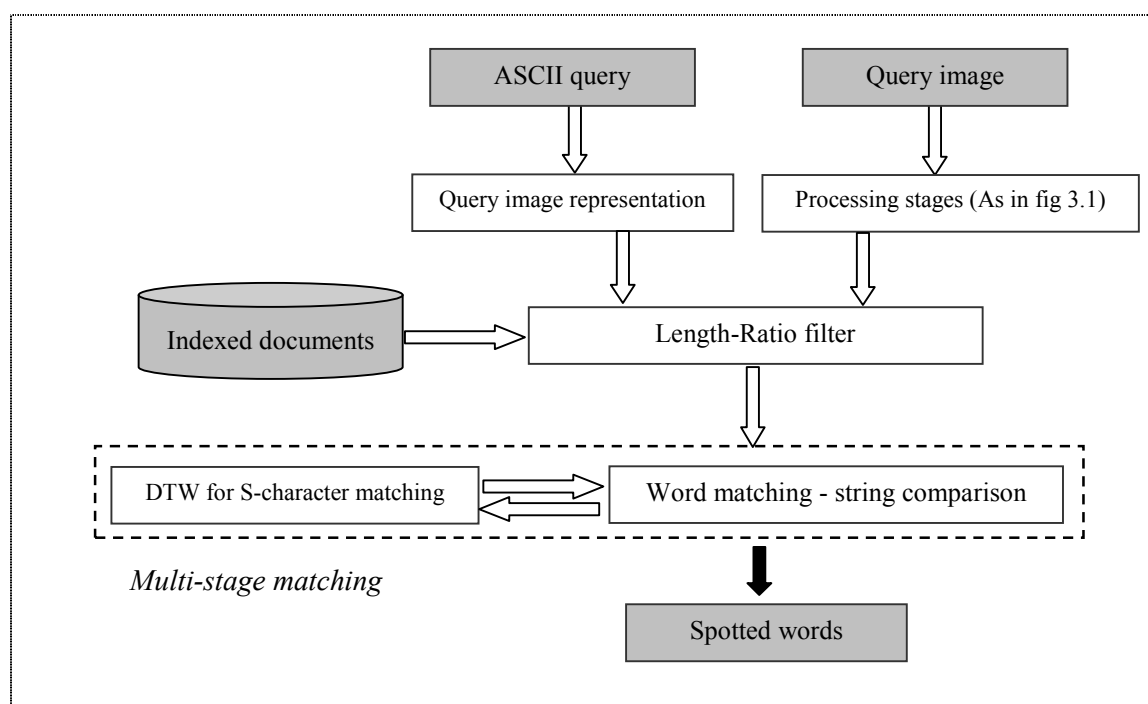


Figure 4.1 - Different stages of word retrieval system

Now before moving on to the actual matching stage, we will first see how the query can be given to the system.

4.2 Query Formation

To search required information and retrieve the relevant document images, we need to give a query to the system. For that, we have developed a prototype system with a user-friendly graphical user interface (GUI) that allows inputting the query in an easy way. User can either click on a word in the opened image in the GUI or he can type in the ASCII word.

4.2.1 Query image - Crisp selection using GUI

When an image is opened in the GUI, its index file is also loaded in memory. If the image opened is a new one i.e. non-processed, then its index file is created on run-time. Now as all the information related to the document image, its words and S-characters is available to the system, the crisp selection of words becomes easy. User clicks at a point on a word, coordinates of that point are analyzed to see which word's bounding box they belong to. Bounding box is then instantly drawn around the selected word. If the user clicks anywhere else on the image (non-text areas), no word will be highlighted as the clicked point coordinates do not belong to any word. Figure 4.2 shows an example screen shot where user clicks on the word “poulie” and it gets selected as the input query.

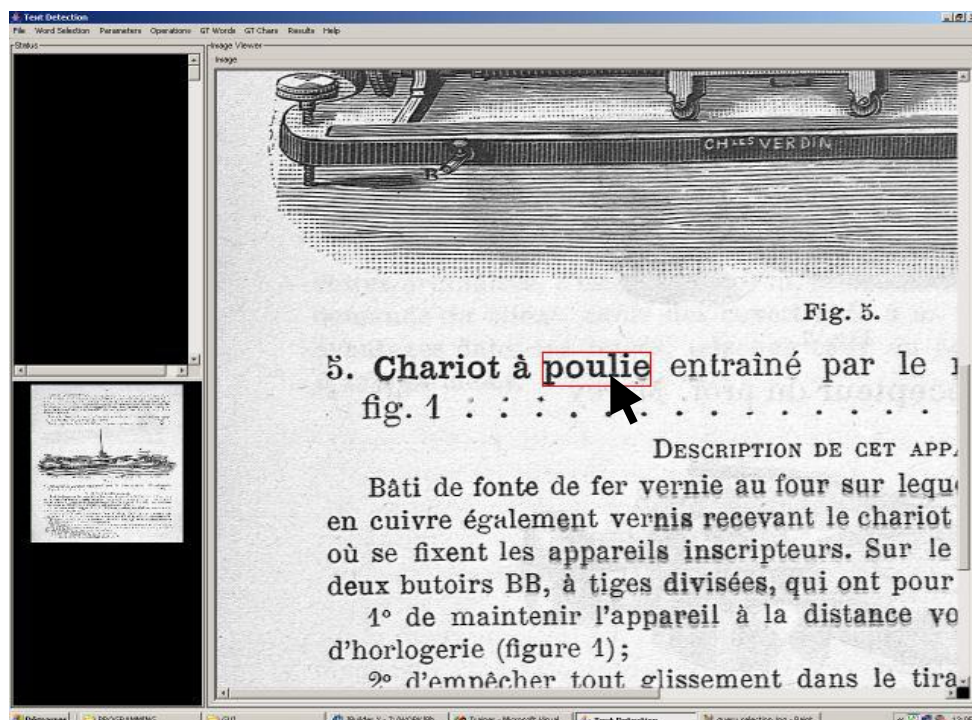


Figure 4.2 - Selection of a query word image by clicking on the document image

All information related to the word, such as its aspect ratio, number of S-characters, features of the S-characters, etc., is extracted from the index file and is stored in a local structure to facilitate in the matching process.

4.2.2 ASCII Query

The query word can also be given in ASCII format. The concept here is simple. User enters an ASCII query for which a synthetic query image is formed using the representative character images. This has been used in literature as well. [Leydier2006] uses prototype characters to form synthetic query word images. Different variations are added to the character images to be able to adapt to different styles. [Doulgeri and Kavallieratou2009] use contemporary fonts to generate synthetic character images using a specific font and size. The query word image is formed by combining these character images with a space in between two character images. This query word is then matched with the other words using word similarity measures. [Rodriguez-Serrano and Perronnin2009] have used 25 different fonts to generate different types of synthetic query word images. Local gradient histogram features are used to model the synthetic word shapes. This concept of synthetic text generation has been applied in other document analysis applications, like handwriting recognition [Helmets and Bunke2003], where training data is generated synthetically using character images of a writer, with very promising results.

To allow query in ASCII, we also use this concept of prototype characters. In our case, we collect prototype characters from document images themselves in multiple ways which we discuss here. The advantage we have over [Doulgeri and Kavallieratou2009] and others is that by defining features at character level, we don't have to form a physical image of the query word. It gives us three advantages:

- We don't need to perform the task of estimating the inter-character spacings in a word
- We don't have to resize the prototype characters to fit in a query word
- We don't have to calculate the prototype character position with respect to a baseline while creating the query word image

All these advantages are due to the fact that we match the words at S-character level and not at word level, so inter-character spacing or prototype character vertical positioning becomes irrelevant.

The prototype characters can be selected manually from the document images. This is done using the GUI of our retrieval system framework by simply clicking on an S-character image and nominating it as a prototype for that label. In different books different sort of fonts are used, but within a book the font styles are usually consistent and uniform. So for one character label, we

can have different prototypes from different books. Figure 4.3 shows some S-character images of label ‘p’ in different books.

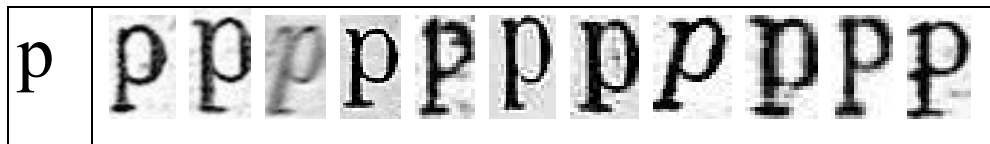


Figure 4.3 – Variations of a character in different books

We don’t store prototype character images physically in the memory. Rather, only the representative feature sequences of the prototype characters are stored in data files named after the character label. For example, a file “sp.dat” contains the features of prototype character images of ‘p’ in lower case and “bp.dat” contains features of the prototype character ‘P’ in upper case. From each book, for each character label, we select one prototype character in upper case and one in lower case.

Now during word matching, when the user types in a query word, the features of the typed characters are read from these prototype data files. If the information search is confined only to one book, then only the features of the prototype characters from the same book are used in creating the query word. Else, all the prototype characters are used in all possible combinations to make multiple variants of the query word. (Here we would like to remind that the query word is not created physically, it is only hypothetical)

Now we move on to the matching process where the given query word is compared with all other words to retrieve appropriate matches. As the data sets are usually very large, a filtering process is introduced at the beginning that filters out the word base and retains only the candidate words for a given query. For that we have implemented a length ratio filter.

4.3 Length-ratio Filter

The document set to be searched for some particular information is usually very large. There are plenty of words in the document base and to retrieve the words similar to the input query word takes a lot of time. This is because the query word is matched against all the words in the index files of the same book. To make the retrieval process time-wise more efficient, we introduce a length-ratio filter before the actual matching processes which narrows down the list of candidate words considerably.

The length-ratio filter is a function of the number of S-characters of the query word. This is because we want to match the query word with words having more or less similar number of S-characters. We don't want to match a word of length 10 if the length of the query word was say only 3. (Here by word length we mean number of S-characters in the word). So, to find what length word is to be matched with which length query, we find the ratio of the number of S-characters of the test and query words as:

$$\text{Length-ratio} = \text{Length}(\text{Test word}) / \text{Length}(\text{Query word})$$

Test word is selected as a candidate if the ratio comes out to be in a threshold interval defined as:

if (*Length-ratio* > *R1* **AND** *Length-ratio* < *R2*) ***then***
Test word is a candidate word

The values of *R1* and *R2* were chosen empirically keeping in view the character segmentation problems with an objective that we don't want to miss any good candidate even if we got more false candidates.

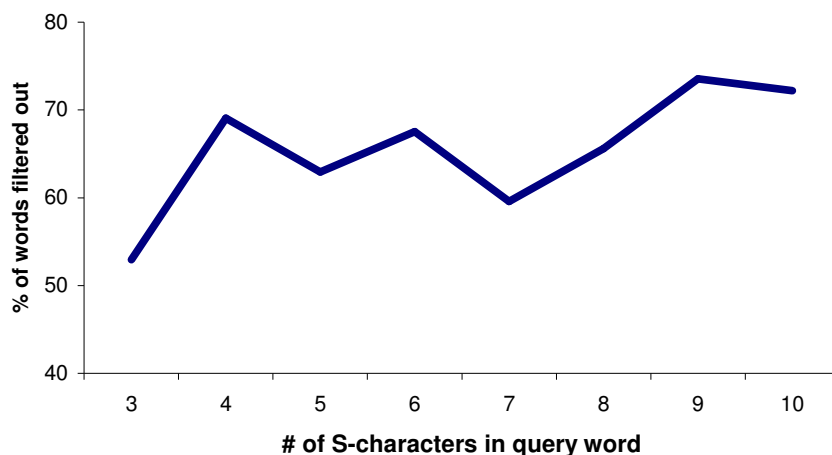
if *Length(Query word)* <= 3 ***then***
R1=0.65;
R2=1.51;
else
R1=0.70;
R2=1.43;

These values can be adjusted by the user depending on the document set. If there are many errors in character segmentation (broken or merged T-characters), then these values can be adjusted to allow more words as candidates. Similarly, if the document quality is good and character segmentation is reliable, then these values can be made more strict (close to 1) so that only the words with exact same length are matched with the query word.

We applied the length-ratio filter on 16 document images of data set A with a total of 4,594 words to analyze how many words are filtered out for different query lengths. Table 4.1 shows the number of words that are kept as candidates and also the number of words that are pruned out at this early stage. Figure 4.4 shows the percentages of words that are filtered out for different query lengths. We can see from the graph that 50 to 75% of the words are filtered out for different query word lengths.

Table 4.1 – Length-ratio filter for different word lengths

Word Length	Total Words	Words selected	Words Filtered out	% Words Filtered Out
3	4594	2162	2432	52.9%
4	4594	1422	3172	69.0%
5	4594	1702	2892	62.9%
6	4594	1492	3102	67.5%
7	4594	1857	2737	59.5%
8	4594	1580	3014	65.6%
9	4594	1215	3379	73.5%
10	4594	1278	3316	72.1%
	36752	12708	24044	65.4%

**Figure 4.4 - Percentage of words filtered out by the length-ratio filter**

4.4 Information Retrieval - Word Spotting

Now once the initial filtering process is done, the query word is then matched with all the candidate words to find relevant matches. This matching is done using a multi-stage retrieval method:

- The actual image matching is done at character level in which the sequence of feature vectors of the two S-characters are matched using elastic DTW method
- These S-character matching distances are then used in the second stage where two words are compared by matching the sequences of S-characters. Different algorithms have been proposed for this second stage of word matching which we will see in detail later on.

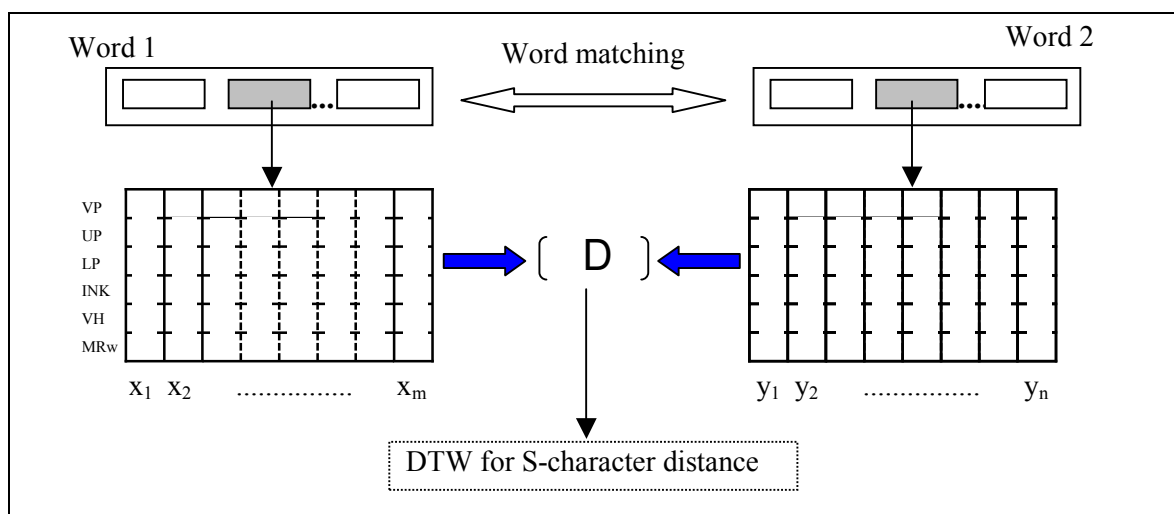


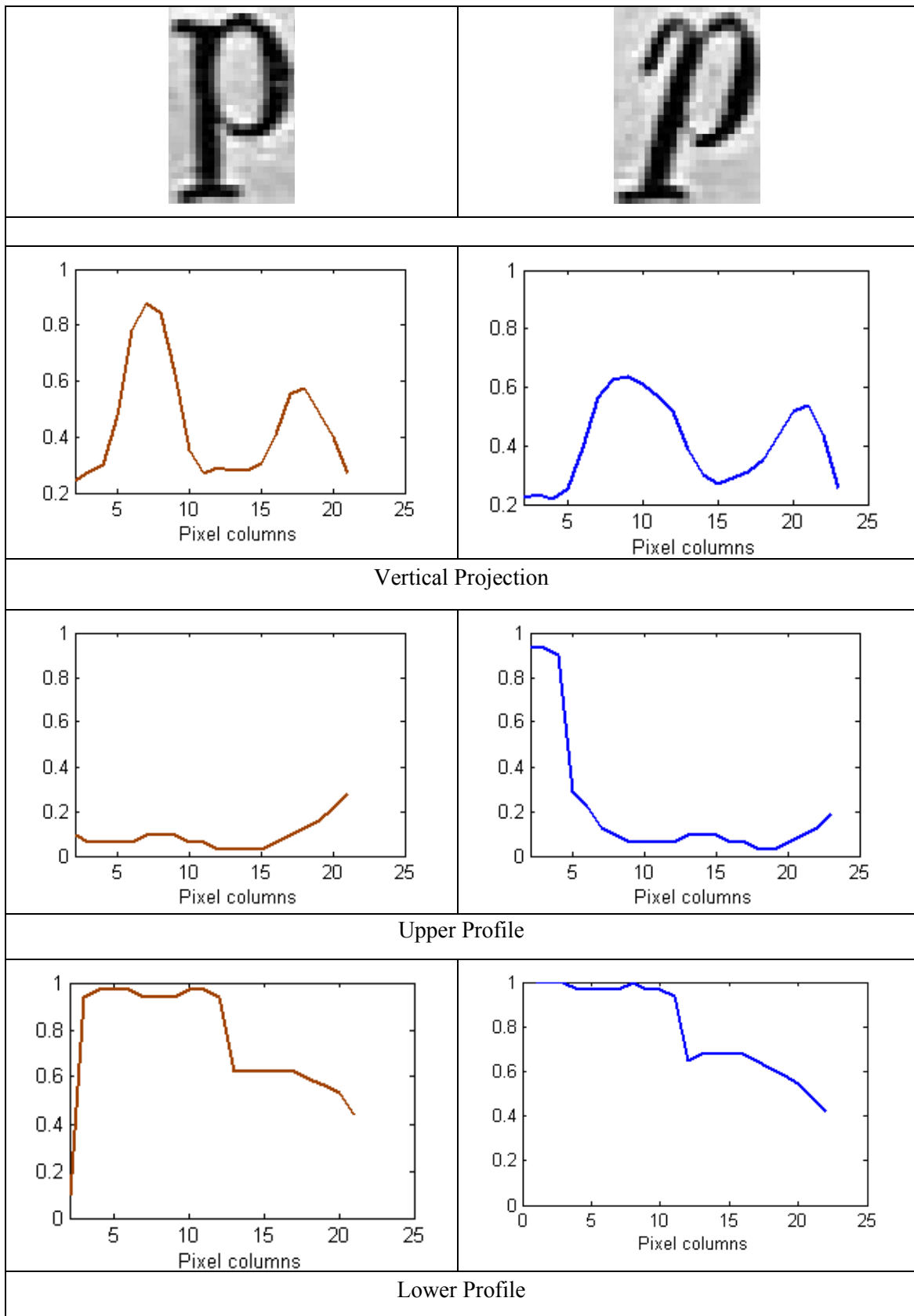
Figure 4.5 – The overall system - highlighted S-characters (in gray) of two words are matched using DTW (matrix D) while two words are compared using different techniques

Figure 4.5 depicts the overall matching system in the form of a block diagram. We can see that the S-characters of two words being compared are matched using DTW matrix D on their feature set. These individual S-character distances are used to compare two words, a word being a string of S-characters, and decide whether they are similar or not.

In the following section, we will see in detail the S-character image matching process. Once we are done with that, we will move over to the word matching process.

4.5 S-character Matching

Two S-character images are compared by matching the sequences of their feature vectors. Several image matching techniques (though at word level) have already been investigated in [Rath *et al.* 2002, Rothfeder *et al.* 2003] with the best performing being Dynamic Time Warping matching (DTW). So here, we also chose to continue with this elastic dynamic matching concept taking it further on S-character matching level. All the S-character matching distances are calculated by matching the sequences of feature vectors of the two S-character images using DTW. The advantage of using DTW in our case is that it is able to account for the nonlinear stretch and compression S-character images [Keogh and Pazzani2001] which may occur due to different font styles and sizes. Hence two same S-characters differing in dimension will be matched correctly. Figure 4.6 shows two S-character images of “p” printed in different styles and their features. We can see that the same profile features are stretched or compressed with respect to one another for the two S-characters. By comparing these features using DTW, we are able to match similar S-characters in different styles correctly.



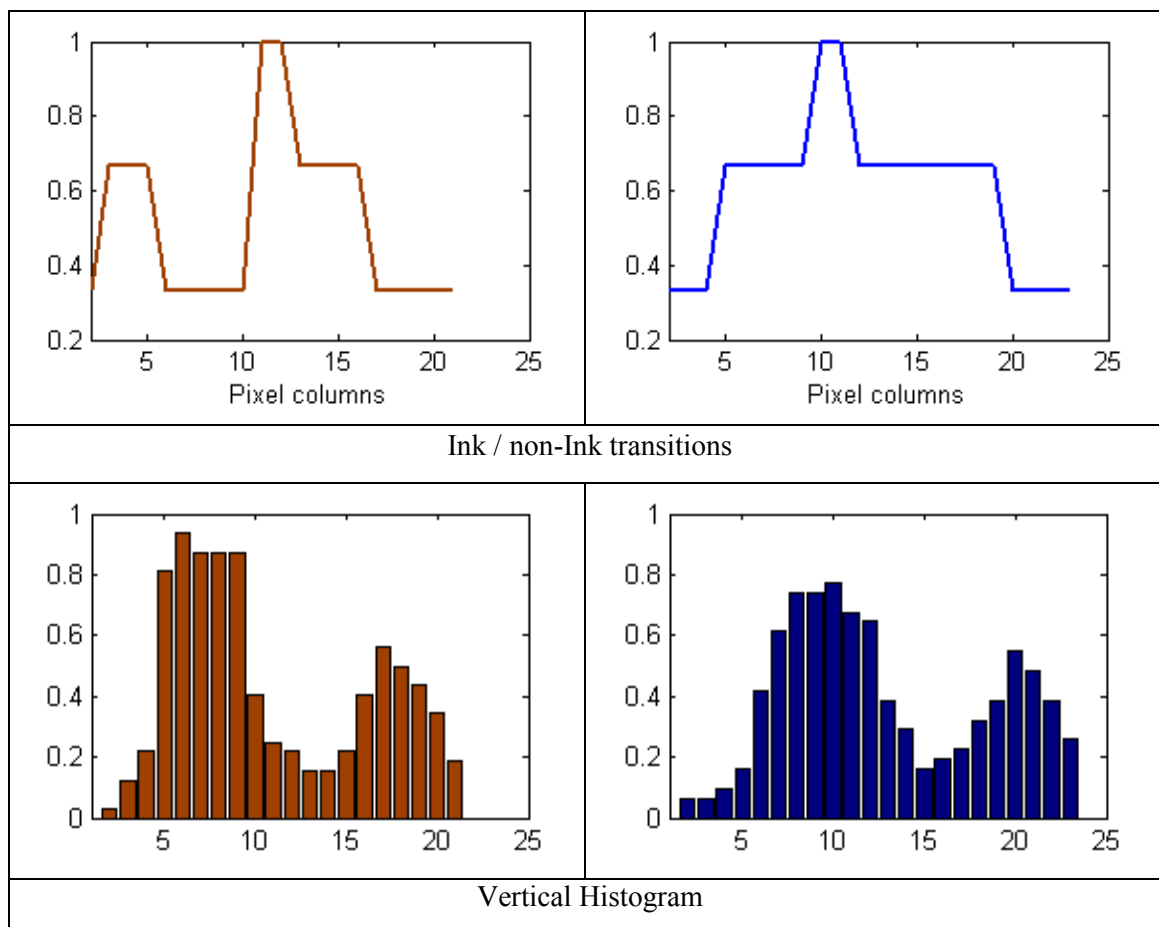


Figure 4.6 - Comparison of the features of S-character 'p' written two in different styles

DTW is described in detail in [Sankoff and Kruskal1999], [Keogh and Pazzani2001]. Its advantage over simple distance measures, such as linear scaling followed by Euclidean distance calculation, is that it determines a common “time axis” (hence the term time warping) for the compared sequences, on which corresponding locations appear at the same time [Rath and Manmatha2007]. Due to the variations in fonts as well as degradation of text due to quality of documents, profiles of two same S-characters do not generally line up very well if they are just scaled linearly (see Figure 4.7).

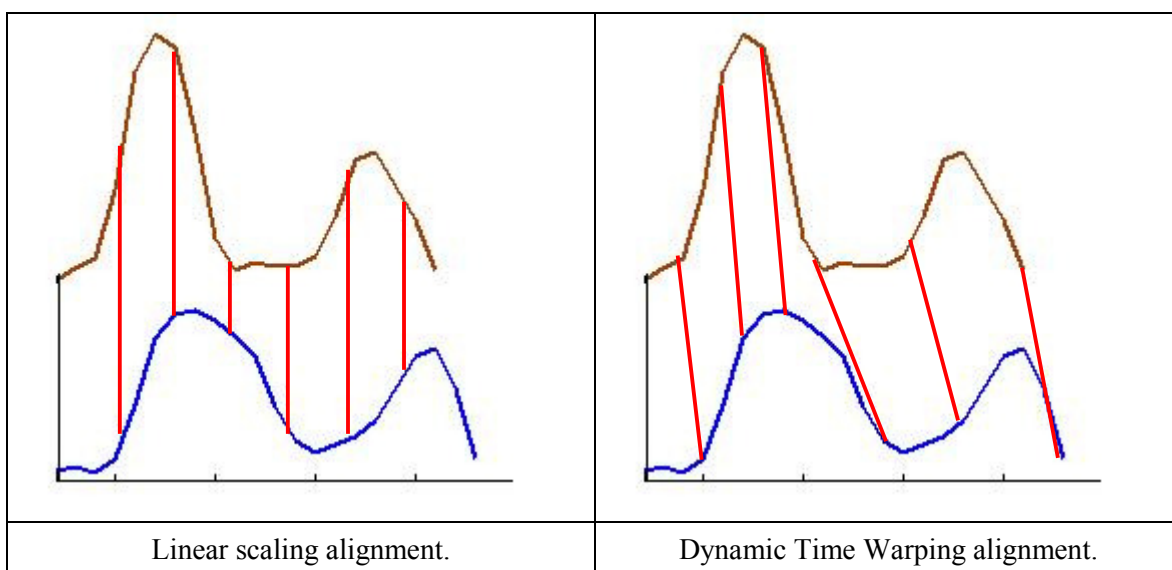


Figure 4.7 – Vertical projection profiles of ‘p’ and ‘p’, aligned using linear scaling and Dynamic Time Warping. DTW ensures that only corresponding locations will be compared

4.5.1 Dynamic Time Warping (DTW)

Dynamic Time Warping is a dynamic programming algorithm that finds correspondences in two signals and calculates a cumulative matching distance using these correspondences. By aligning corresponding samples in the two signals, a *warping path* from the lower right to the upper left of the DTW matrix arises. The matching distance between the two signals is the cumulative cost of aligning all corresponding sample pairs along the path. A local distance measure determines the matching distance between two aligned samples. DTW recovers correspondences between sample locations by finding the warping path with minimum accumulated sample alignment cost.

Consider two S-character images X and Y of widths m and n respectively are represented by vector sequences $X = (x_1 \dots x_m)$ and $Y = (y_1 \dots y_n)$ where x_i and y_j are vectors of length six ($6 =$ dimension of feature vector) (see Figure 4.5). To determine the DTW distance between these two sequences, a matrix D of order $m \times n$ is built where each entry $D(i, j)$ with $1 \leq i \leq m; 1 \leq j \leq n$ is the cost of aligning the sub-sequences $X_{1:i}$ and $Y_{1:j}$. Entry $D(i, j)$ in the matrix is calculated in the following manner:

$$D(i, j) = \min \left\{ \begin{array}{l} D(i, j-1) \\ D(i-1, j) \\ D(i-1, j-1) \end{array} \right\} + d(x_i, y_j) \quad (1 < i \leq m \ \& \ 1 < j \leq n) \quad (4.5.1)$$

The recursive definition of $D(i, j)$ based on the three given values is a *local continuity constraint*. It determines which sample pairs (positions in the matrix) may be connected to form a warping

path. The constraint in equation (4.5.1), which is also shown in graphical form in Figure 4.8, ensures that no sample in any of the two input sequences can be left out from the warping path composed of index pairs $((i_1, j_1), (i_2, j_2), \dots, (i_k, j_k))$, which aligns corresponding samples in the input sequences X and Y .

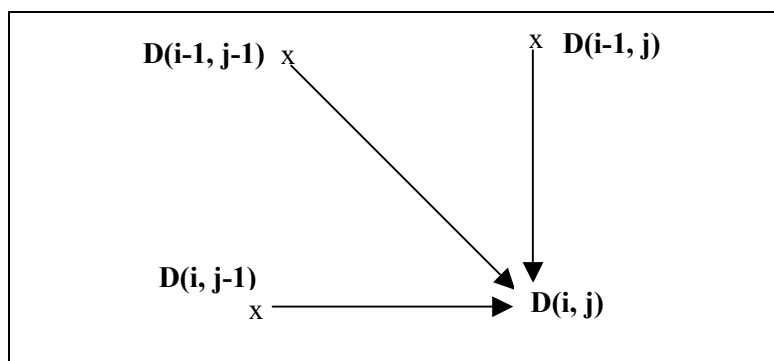


Figure 4.8 - Local continuity constraint, showing valid neighborhood relationships in a warping path

We represent S-character images with sequences of features (see chapter 3). Single dimensional sequences that are extracted from an S-character have same lengths, so an S-character can be represented by a sequence of feature vectors with length of each vector equals the number of features. Hence, when matching the S-character images, the sequences X and Y consist of vectors of dimensionality $p > 1$. This ensures that all sequences are warped in the same manner.

In order to use DTW to match multi-dimensional profiles, we need to define a distance measure $d(\cdot, \cdot)$ that determines the (local) distance between two samples in a sequence. Here, for $d(x_i, y_j)$, we have used the Euclidean distance in the feature space:

$$d(x_i, y_j) = \sqrt{\sum_{k=1}^p (x_{i,k} - y_{j,k})^2} \quad (4.5.2)$$

where the index k is used to refer to the k -th dimension of x_i and y_j and p in our case is six as we have six features.

With this distance measure defined, we can now calculate the matching distance between two S-characters by comparing their feature sequences using DTW. Table 4.2 contains pseudo-code for the DTW algorithm using the local continuity constraint from Figure 4.8.

Table 4.2 - Pseudo code for the DTW algorithm

Input: $X = (x_1 \dots x_m)$ and $Y = (y_1 \dots y_n)$

Output: DTW matrix D for the two characters

Algorithm:

$$D(1, 1) = d(x_1, y_1)$$

for $i = 2$ to m

$$D(i, 1) = D(i-1, 1) + d(x_i, y_1)$$

for $j = 2$ to n

$$D(1, j) = D(1, j-1) + d(x_1, y_j)$$

for $i = 2$ to m

for $j = 2$ to n

$$D(i, j) = \min \left\{ \begin{array}{l} D(i, j-1) \\ D(i-1, j) \\ D(i-1, j-1) \end{array} \right\} + d(x_i, y_j)$$

The entry $D(m, n)$ of the matrix D contains the matching distance between the two S-characters. For example, while matching two S-characters of length 4 and 5 using DTW, the entry $D(4, 5)$ of the matrix D will represent the matching distance between them (see Figure 4.9). But this distance is non-normalized i.e. it may vary drastically for similar S-characters having different widths. So to be able to set a threshold that applies to all types and sizes of S-characters, we need to normalize the final distance $D(m, n)$.

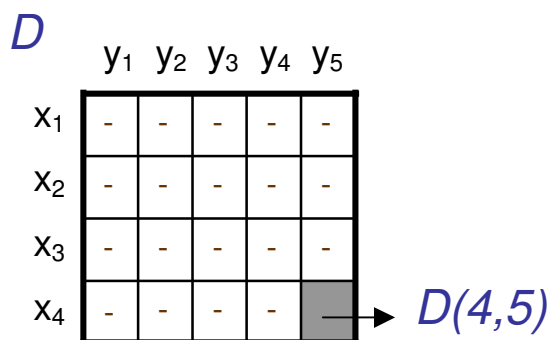


Figure 4.9 - The entry $D(4,5)$ shows the distance between two sequences X and Y

4.5.1.1 Normalization Factor K

The final distance $D(m, n)$ also depends on the widths of the two S-characters. If the width is large, this value will be high. To make it independent of the width of S-characters, we normalize this value so that one threshold can be determined for all S-character matching. Normalization has been tested in two ways:

- By finding minimal cost path
- By using average of the widths

a. Minimal cost path

Once all necessary values of D have been calculated, the warping path is determined by backtracking the minimum cost path starting from (m, n) towards $(1, 1)$. However, we are just interested in the accumulated distance along the warping path, which is stored in $D(m, n)$. As it is, this matching distance is smaller for short sequences and large for bigger sequences, so we offset this bias by dividing the final distance value by the length 'L' of the warping path. So the final matching distance the two S-characters is given by the following value:

$$\text{dist}(X, Y) = D(m, n) / L \quad (4.5.3.)$$

A drawback of this normalization approach is that if we have two S-characters which are different, we may get a large matching distance $D(m, n)$. When we divide it by L (which could be large in this case as many steps might exist in the final warping path for the two S-characters), the final distance $\text{dist}(X, Y)$ may become small which might lead to some false positives. So we introduce another normalization based on the average widths of the two S-characters.

b. Average character width

In this case, we normalize the distance $D(m, n)$ by dividing the value by the average of the widths of two the S-characters.

$$\text{dist}(X, Y) = D(m, n) / [(m+n)/2] \quad (4.5.4)$$

Normalization using average width of the two S-characters gives better results and reduces the number of false positives. This is because when two different S-characters are matched, the value of $D(m, n)$ is large. We divide this value by a smaller value (average of the widths of two S-characters is always less than the warping path), so the final distance value remains high. If two S-characters are same, then the average of their widths is almost equal to the warping path. This is because the warping path, ideally, will be the diagonal path of the matrix. Another advantage of this normalization over the warping path one is that it is computationally less expensive.

Now during DTW matching, different feature sequences are matched using Euclidean distance, it is possible that some features contribute more in the final matching distance and thus the importance of the features with lower contribution will be quite low. To make sure that each feature gets equal importance and the final matching cost is not biased towards just one feature, we need to normalize all individual feature distances. After normalization, once each feature gets equal importance in the final distance, we can analyze individual feature performances and assign weights accordingly.

4.5.2 Feature distance Normalization

Different features depict different characteristics and have largely varied values. When we compare two S-character images using just a single feature sequence, the distance value may come much different for different features. That means we cannot just simply add the distances of all the features in the Euclidean distance as we have done. We need to have some weights to balance the feature involvement in the final distance value. Although we normalize each of the features to the range $[0, 1]$, the individual distances by using one feature at a time can be of quite different dynamic ranges and combining these distances, the distance with larger magnitude might dominate the others. We need to normalize individual distances so that every feature gets an equal share of importance.

For that, we first need to analyze what sort of distances we get when we use a single feature sequence for comparisons. For the analysis, we calculated the distances of each S-character image with the others using prototype character images of different books. We also took some document images from data set B and calculated the distance of each prototype character with all the S-characters in those pages. In total, we made 85,718 different S-character comparisons using a single feature sequence at a time and then using all the features together as well. Figure 4.10 illustrates the distances between different S-characters using individual feature sequences. We can see that the values of the distances differ for different features – smallest being observed for the vertical projection. If we simply add these individual distances for two S-characters, the vertical projection distance will be totally overshadowed by distances using other feature sequences, thus leaving vertical projection with having little or no effect on the final outcome. We therefore need to normalize each of the distances to a uniform range before proceeding to their combination [Rui *et al.* 1998].

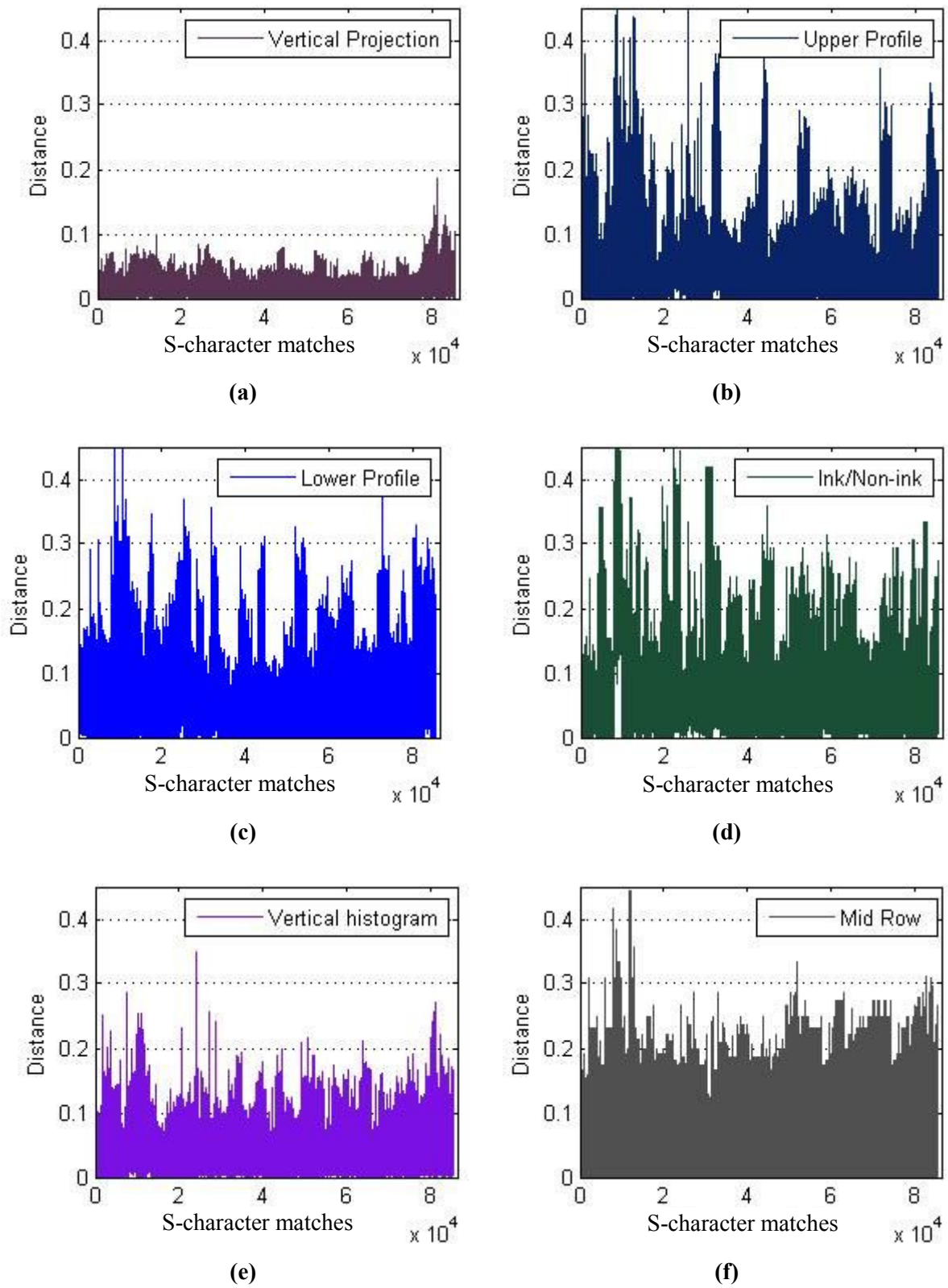


Figure 4.10 - Individual distances between S-characters using only a) Vertical projection b) Upper profile c) Lower profile d) Ink/non-ink e) Histogram d) Mid row transitional sequence

The simplest normalization technique would consist of finding the maximum and minimum values in the list of all distances (85,718 distances in our case) for a feature and normalize the sequence to the range [0,1] using the maximum / minimum values. This normalization, however, suffers from the drawback that an abnormal high value can take away most of the [0,1] range, leaving a very narrow range for the rest of the values. So it cannot be used. The distance histogram using all features is shown in Figure 4.11. We can see that the distribution almost follows a Gaussian distribution. The mean and standard deviation of the distance values using individual features is given in Table 4.3. We thus employ the Gaussian normalization [Chhikara and Folks1989] for our individual feature distances.

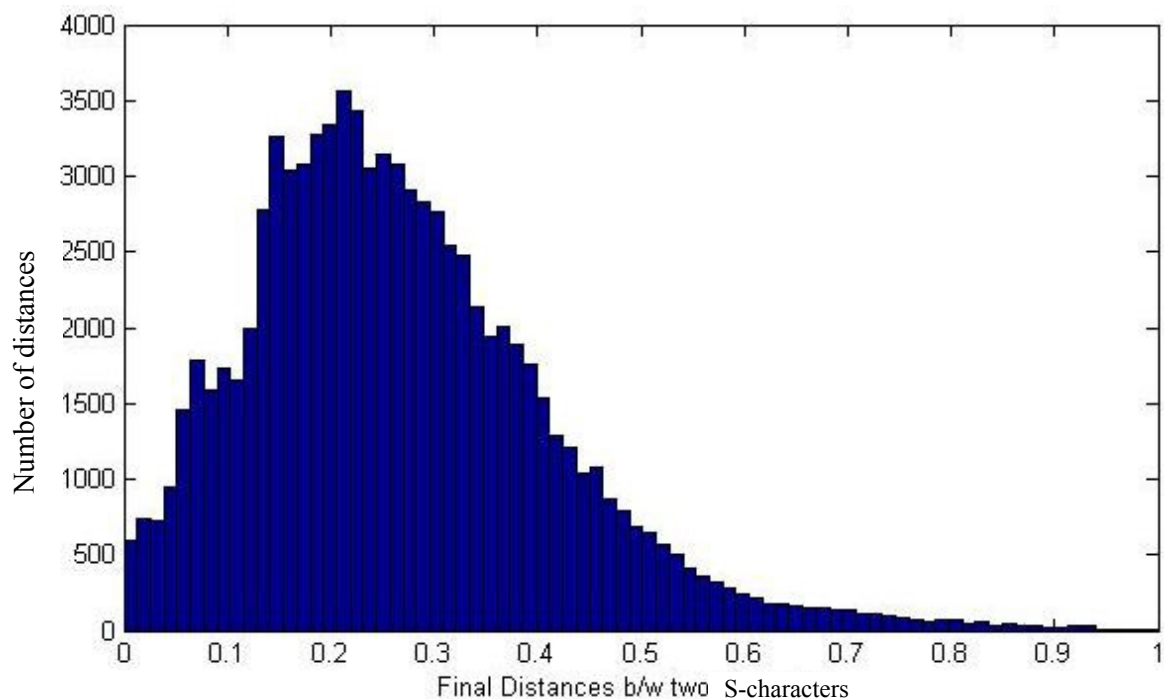


Figure 4.11 - Distance histogram using all features for 85,718 comparisons

For Gaussian normalization, we compute the distance sequence (d_i) between different pairs of S-character images I_m and I_n using feature ‘i’ and repeat it for all six features. Now treating each d_i as a data sequence we find its mean μ_{di} and standard deviation σ_{di} . These values are computed offline and are based on the assumption that the number of comparisons is large enough that the calculated values represent true mean and standard deviation of the distances between S-characters. The mean and standard deviation of the distance values using individual features is given in Table 4.3. These values are used in the normalization procedure.

Table 4.3 - Overall distance values using individual features as well as all the features for 85,718 different comparisons

	LP	UP	INK	VP	VH	MID ROW	Using All features
Maximum distance	0.4915	0.5379	0.5236	0.1868	0.348	0.4444	2.5322
Mean Distances	0.0404	0.043	0.0524	0.0168	0.0383	0.0772	0.2681
Std deviation	0.0532	0.0431	0.0468	0.0134	0.0295	0.0625	0.2485

Once these values are calculated, we employ them for distance normalization during the matching process. When two S-characters C_k and C_l are being matched, we first compute the raw distance: $d_i(C_k, C_l)$ between the feature sequence 'i' of the two. Normalized distance is found as follows:

$$i = 1 \text{ to } 6$$

$$d'_i(C_k, C_l) = \frac{d_i(C_k, C_l) - \mu_{di}}{3\sigma_{di}}$$

This normalization for a Gaussian distribution of data produces 99% of all the distances in the range $[-1, 1]$, which is finally shifted to $[0, 1]$ as:

$$d''_i(C_k, C_l) = \frac{d'_i(C_k, C_l) + 1}{2}$$

In our case, as the distribution was not a true Gaussian, more than 1% of the values can lie outside the $[0, 1]$ interval. But even in that case, the distances having a value greater than 1 are of course very dissimilar and they will have no effect on the results of matching. Figure 4.12 illustrates the normalized distances between different S-characters using individual feature sequences.

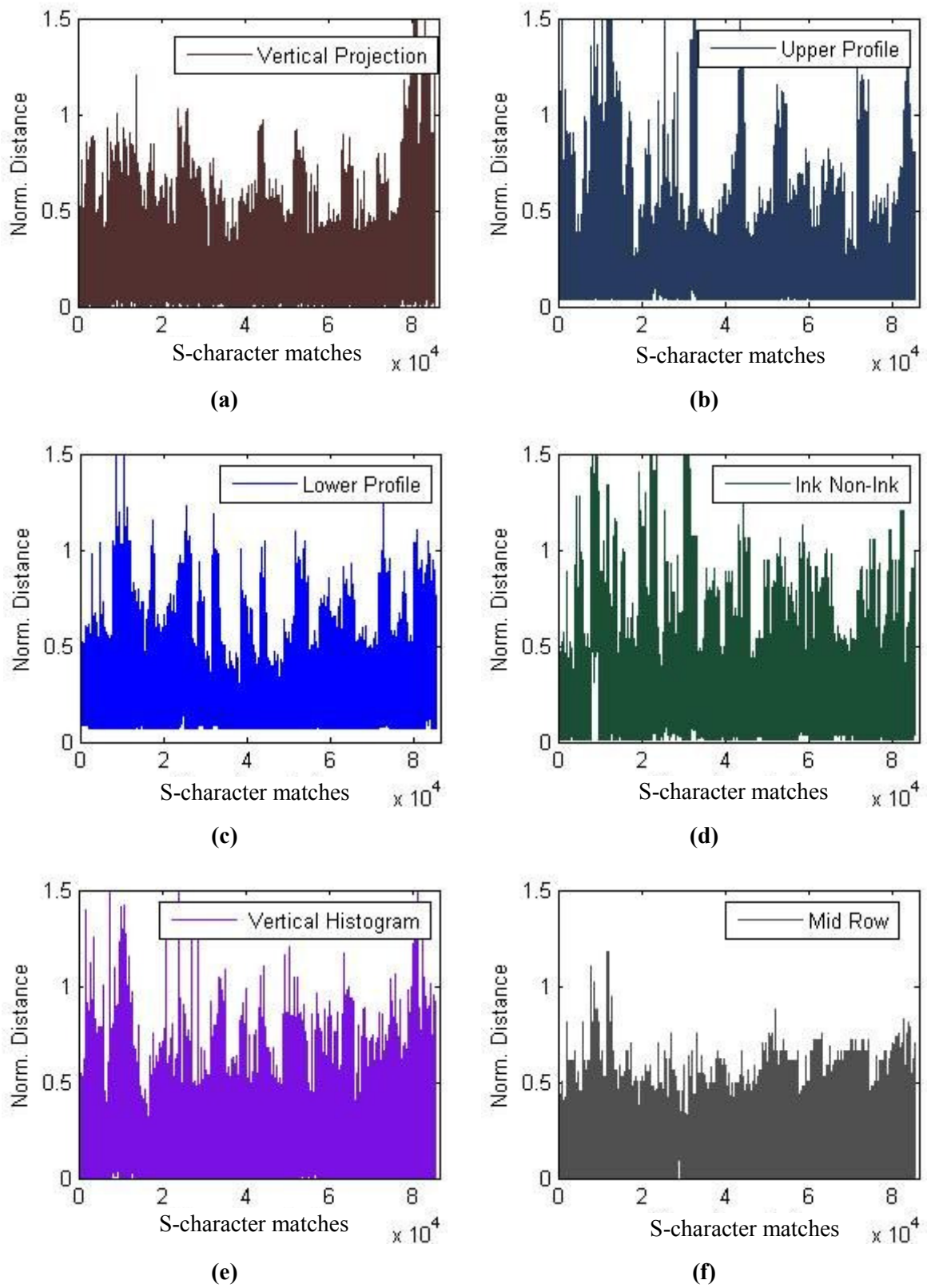


Figure 4.12 - Individual normalized distances between S-characters using only a) Vertical projection b) Upper profile c) Lower profile d) Ink/non-ink e) Histogram d) Mid row transitional sequence

The normalized distance histogram for the same 85,718 matches using combined features is shown in Figure 4.13.

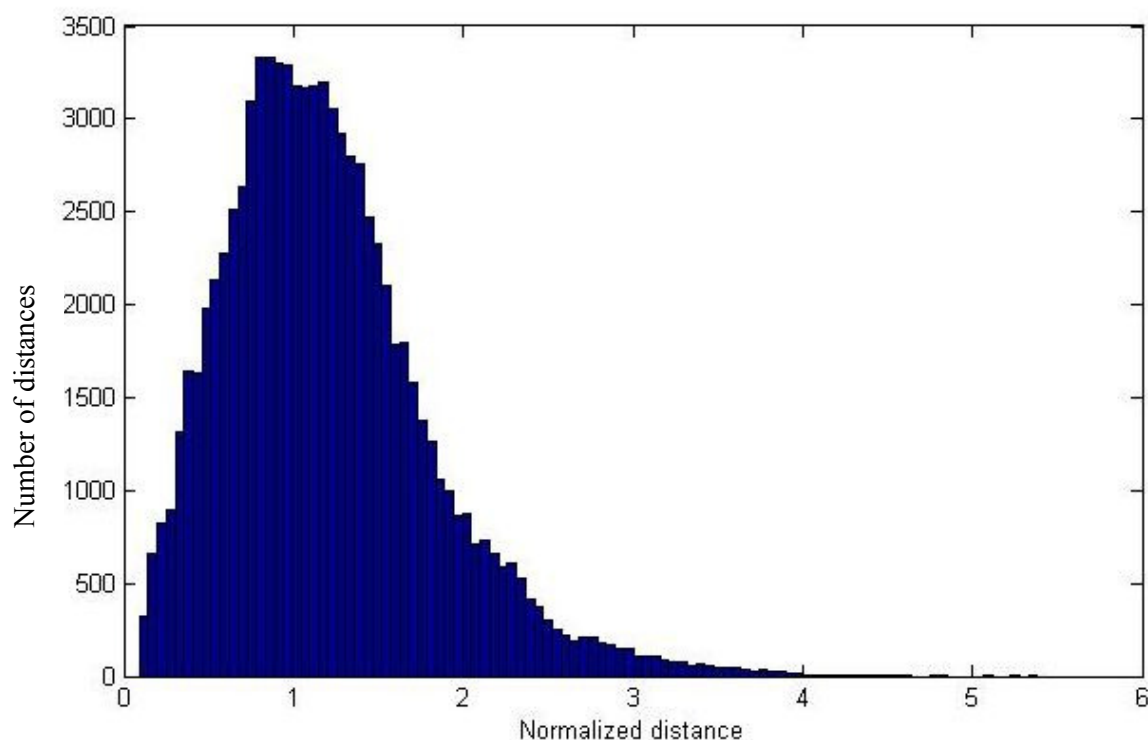


Figure 4.13 - Normalized distance histogram using all features

In this section, we saw S-character matching stage in detail. We also saw the normalization of the distances so that now when two S-characters are matched, we get a normalized distance between them. This normalized S-character distance is used during the matching of two words. We will see the individual feature performance in the experimental results section later on in this chapter.

In the next section, we will discuss word matching and propose different ways in which the S-characters of the two words are compared. It will be followed by detailed results for each of the methods.

4.6 Word Matching

The first step of word spotting process performs a *length-ratio* filtering (explained earlier in section 4.3), finds all eligible word candidates for the query word. This step, on average, eliminates more than 65% of the words. The second step is the main matching step. We have proposed four different string comparison methods to match the S-characters of the query word and candidate words (remaining words which are not filtered away by length-ratio filter). These

methods include Relative Position Correspondence (RPC), Edit distance, Merge-Split Edit distance and Linear displacement matching. We discuss them one by one in the following sub sections.

4.6.1 Relative Position Correspondence (RPC)

The proposed Relative Position Correspondence (RPC) is a string matching algorithm based on the concept of matching S-characters at relatively similar positions in the two words. It means one S-character of query word is matched with different number of relative neighboring S-characters in the candidate word to find its best match (see Figure 4.14). The number of neighboring characters to be tested against a query S-character depends upon the size of the query word [Khurshid *et al.* 2008a, Khurshid *et al.* 2008b].

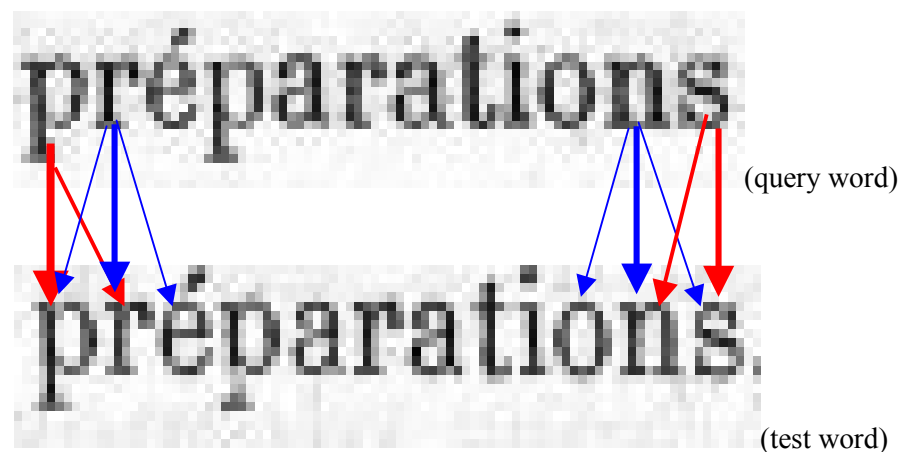


Figure 4.14 - RPC between S-characters of two words of length 12 with $X = 1$

The number of neighboring S-characters ' X ' that are to be tested against an S-character of the query word is set as:

```

if Query Length is less than or equal to 3 then
     $X = 0$ 
else if Query Length is less than or equal to 12 then
     $X = 1$ 
else if Query Length is greater than 12 then
     $X = 2$ 

```

The value of X signifies that for an S -character 'c' at n^{th} position in the query word, X S -characters present on either side of the n^{th} S -character in the candidate word have to be tested against 'c'. So the maximum possible number of comparisons for one query S -character can be:

$$\text{Maximum comparisons for a query } S\text{-character} = (2X + 1)$$

If 'c' is an extreme S -character (S -character at the start or end of the word), then X S -characters on only one side are considered as candidates as there is no S -character on the other side. Figure 4.14 shows an example with a real query word of length 12 having $X = 1$. Figure 4.15 shows the same for words of lengths 3 and 15 respectively.

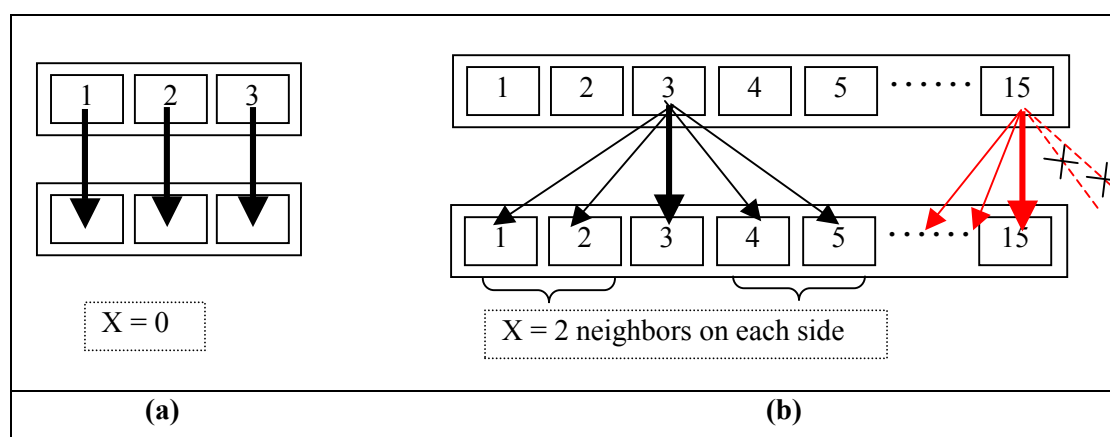


Figure 4.15 – S -character comparisons for a) query word of length 3 b) query word of length 15. S -character 1 & 15 (corner S -characters) are compared with three S -characters, S -character 2 & 14 are compared with four S -characters, while others are compared with five S -characters in the test word

The reason behind matching one query S -character with more than one candidate S -characters for longer words is to cater for some segmentation errors that might exist in the indexing process. For each query S -character, we find its best match from the inspected candidate S -characters using DTW. The best match distance is added to the total *word distance*. For smaller words though (words of length 3 or less), we only match the S -characters at corresponding positions because if we allow more matches, we usually get more false positives. This is because if we consider neighboring S -characters by having a non-zero X , a small word such as 'en' can be matched with 'ne' (Figure 4.16). So for that, we keep $X = 0$. It is a trade off though, as by keeping $X=0$, we might lose some similar words with character segmentation problems.

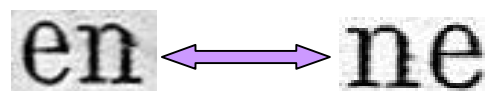


Figure 4.16 - Example of a false positive for a small word with a non-zero ‘X’

After matching the S-characters of the two words, we normalize the total word distance:

$$\text{Normalized-word-distance} = \text{Sum of all best match costs} / \text{number of matches}$$

We have another decision parameter in *similar-S-character-count* which shows the total number of same/similar S-characters in the two words. For each query S-character, the cost of the best match is compared with an empirically determined threshold (*char-threshold*). If this cost is less than the threshold, we increment the *similar-S-character-count*.

Now for two words to be ranked similar, the *similar-S-character-count* must be greater than an empirically determined count threshold ‘*T*’ and the *Normalized-word-distance* must be less than a *word-threshold* which is determined empirically through experimentation as well. If both the conditions are satisfied, the two words are ranked as similar. A brief pseudocode is given in Table 4.4 for a better understanding of the general system. The value of *T* depends upon the application requirements. If high precision is required, value of *T* should be close to the length of the query word, while if recall rate is more important, then value of *T* should be somewhere close to three-quarters the length of the query word.

The RPC method was compared with word-level matching methods like the method of *Rath et al.* [Rath and Manmatha2007] and the method of correlation in [Khurshid *et al.* 2008a], concluding that the character-feature based methods perform better than the word-feature ones. Overall, this method works fine giving us good recall rates, but precision is not that good due to the presence of false positives (details in the experimental results section). The reason behind being that the order of S-characters in a word has not been given major importance which eventually causes false positives to appear. Figure 4.17 shows an example where we get a false positive ‘entre’ for the given ASCII query ‘nette’ for $X=1$ and $T=4$ in the RPC algorithm parameters. This is because all five of the query S-characters get exactly matched with four of the test S-characters. Thus the overall word distance is low (lower than the *word-threshold*) and the parameter *similar-S-character-count* is 5 which is more than *T*. Thus the system retrieves this false positive. As there is no record for a test S-character that has already been matched with a query S-character, it can get matched again with another query S-character resulting in the false positive as shown in Figure 4.17.

Table 4.4 - RPC pseudocode

Input: Query word and Candidate word, char-threshold, word-threshold, T

word-distance=0;

similar-S-character-count=0;

number-of-matches=0;

for each query S-character

compare it with the relative (2X+1) S-characters in candidate word by DTW

*best match cost = S-character match having **minimum** distance among the (2X+1) matches*

word-distance = word-distance + best match cost

increment number-of-matches

If best match cost < char-threshold then

increment similar-S-character-count

Normalized-word-distance = word-distance / number of matches

If (similar-S-character-count > T) AND (Normalized-word-distance < word-threshold) then

*Rank as **similar** words*

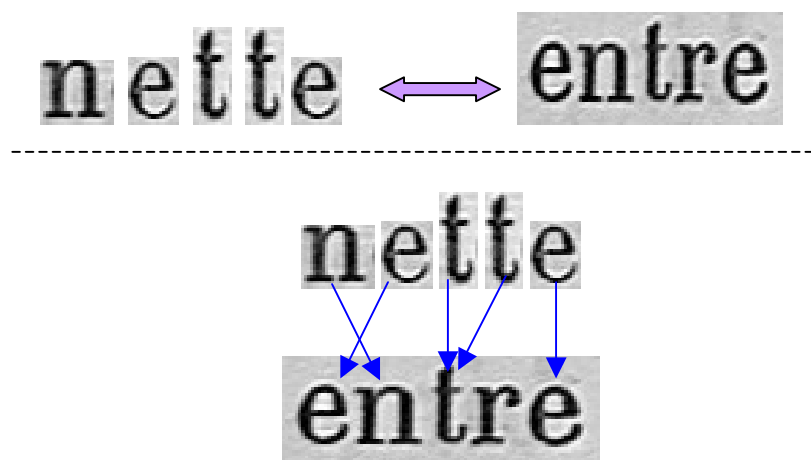


Figure 4.17 -Example of an unwanted match for a query word 'nette'

Considering the drawbacks in RPC, we introduced improvements in the result by implementing a multi-stage dynamic time warping algorithm where two words are compared using the classic Edit distance matrix (W) implementation while individual characters are compared (as before) using elastic DTW (D) coupled with Euclidean distance (see Figure 4.18).

Matrix W is calculated dynamically for the two words being matched. Each entry of the matrix W is found by calculating another dynamic matrix D for the two S-characters. It is shown graphically in Figure 4.19. This implementation of coupling the two dynamic programming systems shows significant improvements in the word retrieval results (as later shown in the result section).

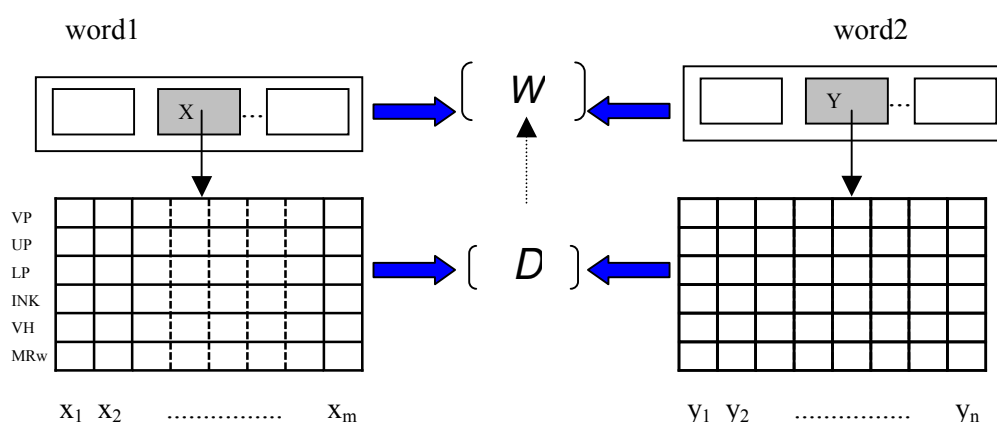


Figure 4.18 - Multistage system - features of two S-characters X and Y are matched using elastic DTW (D) while the two words are matched using Edit matrix (W)

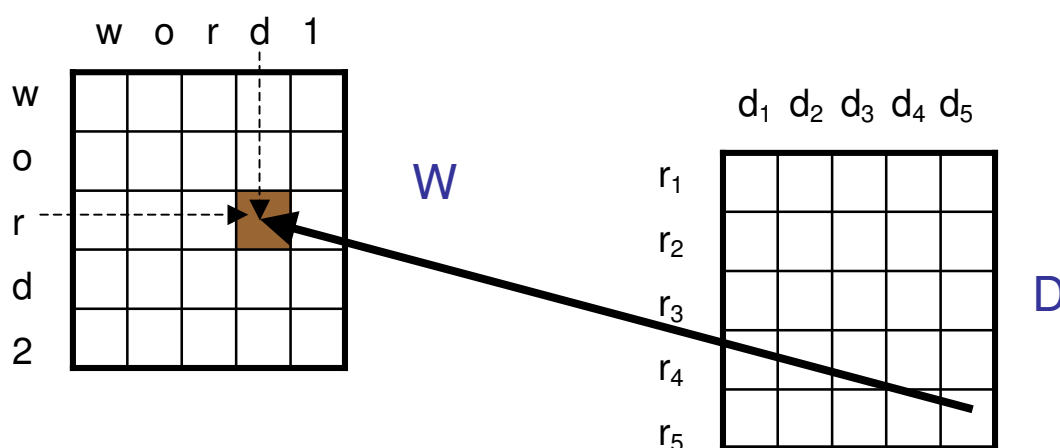


Figure 4.19 - Each entry of matrix W is dynamically calculated using a DTW matrix (D)

Here we will discuss the implementation of Edit distance algorithm for word matching in detail.

4.6.2 Edit Distance

Edit distance has been used for different string matching problems. It was first introduced by Levenshtein in [Levenshtein1966] and since then different modifications have been introduced in the algorithm. In [Wagner and Fischer1974], authors have discussed it in detail for string to string matching problems where the basic aim is to determine the distance between two strings of symbols (in our case it is a word as a *word* is actually a *string of S-characters*). This distance is measured by the minimum cost sequence of “Edit operations” needed to change one string to other. Edit operations include replacing one symbol of a string by another, inserting one symbol into a string, or deleting a symbol from the string. These three operations (Replace, Insert, Delete) constitute the basic edit operations [Wagner and Fischer1974]. Our implementation of Edit distance is given here.

Consider two words A and B which are to be matched; A having m S-characters and B having n S-characters. We treat both words as two series of S-characters represented as $A = (a_1 \dots a_m)$ and $B = (b_1 \dots b_n)$ respectively where a_i and b_j represent the individual S-characters of the two words. To determine the Edit distance between these two S-character series, a matrix W of order $(m+1) \times (n+1)$ is built where each entry $W(i, j)$ with $0 \leq i \leq m$, $0 \leq j \leq n$ is the cost of aligning the substrings $A1:i$ and $B1:j$. (To keep the pseudo code simple and compatible with the array implementation, we will use the position index from 0 to m and 0 to n . The entries (i, j) of the matrix W are calculated as:

$$\begin{aligned}
 W(0,0) &= 0 \\
 W(0, j) &= W(0, j-1) + \gamma(\Lambda \rightarrow b_j) \quad (\text{for } j > 0) \\
 W(i,0) &= W(i-1,0) + \gamma(a_i \rightarrow \Lambda) \quad (\text{for } i > 0) \\
 W(i, j) &= \min \left\{ \begin{array}{l} W(i-1, j-1) + \gamma(a_i \rightarrow b_j) \\ W(i-1, j) + \gamma(a_i \rightarrow \Lambda) \\ W(i, j-1) + \gamma(\Lambda \rightarrow b_j) \end{array} \right\} \quad (1 \leq i \leq m; 1 \leq j \leq n)
 \end{aligned} \tag{4.6.1}$$

As in the case of the S-character matching algorithm, the recursive definition of $W(i, j)$ is based on the three given values is a *local continuity constraint*. Here, Λ represents an empty character having the values in its feature vectors set to 0. The width of this empty character has been set to 25, which is normally the width of an average character. The three edit operation costs are represented by $\gamma(a_i \rightarrow b_j)$, $\gamma(a_i \rightarrow \Lambda)$ and $\gamma(\Lambda \rightarrow b_j)$.

- $\gamma(a_i \rightarrow b_j)$ is the cost of replacing a_i by b_j
- $\gamma(a_i \rightarrow \Lambda)$ is the cost of deleting a_i from the string
- $\gamma(\Lambda \rightarrow b_j)$ is the cost of inserting b_j to the string

All these character matching costs are calculated using Dynamic Time Warping, where we match the feature vectors of two S-characters. It has already been discussed in the character matching process.

With this distance measure defined, we can now calculate the matching distance between two words by comparing their S-characters using Edit distance in equation (4.6.1). Table 4.5 contains pseudo code of the implemented Edit distance algorithm implementation. The algorithm determines a path composed of index pairs $((i_1, j_1), (i_2, j_2), \dots, (i_k, j_k))$, which aligns corresponding samples in the input sequences A and B .

Table 4.5 - Pseudo code for the Edit distance algorithm

Input: Two words $A = (a_1 \dots a_m)$ and $B = (b_1 \dots b_n)$; γ (DTW function)

Output: Edit matrix W for the two words

Algorithm:

$W(0, 0) = 0$

for $i = 1$ to m

$W(i, 0) = W(i-1, 0) + \gamma(a_i, \Lambda)$

for $j = 1$ to n

$W(0, j) = D(0, j-1) + \gamma(\Lambda, b_j)$

for $i = 1$ to m

for $j = 1$ to n

$$W(i, j) = \min \left\{ \begin{array}{l} W(i-1, j-1) + \gamma(a_i \rightarrow b_j) \\ W(i-1, j) + \gamma(a_i \rightarrow \Lambda) \\ W(i, j-1) + \gamma(\Lambda \rightarrow b_j) \end{array} \right\}$$

The entry $W(m, n)$ of the matrix W contains the total distance between the two words which is the cost of matching them. For example, for two words of length 4 each, the entry $W(4, 4)$ of the matrix W will represent the cost of matching the two words using Edit distance (see Figure 4.20).

This distance is non-normalized i.e. it may vary drastically for words having different number of S-characters. To be able to match all words and set a threshold that applies to all types and sizes of words, we need to normalize the final distance $W(m, n)$.

W	Λ	b_1	b_2	b_3	b_4	
	Λ	0				
	a_1					
	a_2					
	a_3					
	a_4					$\rightarrow W(4,4)$

Figure 4.20 - The entry $W(4, 4)$ is the cost of aligning the two words of length 4

4.6.2.1 Normalization Factor K

The final distance $W(m,n)$ depends on the number of S-characters of the two words. The more the number of S-characters in two words, the larger will be this distance value. This value has to be normalized to make it independent of the size of the words, enabling to determine one threshold for all word's matching.

For that, once all necessary values of W have been calculated, the warping path is determined by backtracking the minimum cost path starting from (m, n) towards $(0, 0)$. However, we are just interested in the accumulated cost along the warping path, which is stored in $W(m, n)$. As it is, this matching cost is smaller for shorter sequences, so we eliminate this bias by dividing the total matching cost by the number of steps N of the warping path. So the final distance between the two words becomes:

$$\text{dist}(A, B) = W(m, n) / N$$

Two words are ranked similar if this final matching cost is less than an empirically found word-threshold.

This method of word retrieval using a combination of Edit distance and DTW works well and improves recall rates significantly as compared to RPC algorithm (details in results section). But there still remain some problems that are mainly related to the character segmentation problems. If the segmentation of characters is erroneous, T-character is broken into multiple S-characters or

multiple T-chatacrers are merged into one S-character, then we might not be able to retrieve the words using Edit distance. Figure 4.21 shows a couple of examples of words not matched due to segmentation errors.

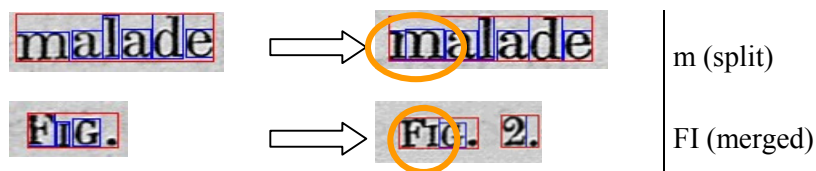


Figure 4.21 - Words not matched due to segmentation problems

To be able to retrieve these words as well, we have proposed a new Merge-Split Edit distance that takes into account the segmentation problems caused by document quality. It is discussed in detail now.

4.6.3 Merge-Split Edit Distance

Different variations of Edit distance have been proposed in literature for different matching applications [Ambauen *et al.* 2003], [Kaygin and Bulut2002], [Christodoulakis and Brey2008], [de Waard1995]. *Kaygin et al.* introduced a variation of Edit distance for shape recognition in which polygon vertices are taken as primitives and are matched using the modified Edit distance. The operations of inserting and deleting a vertex represent the cost of splitting and combining the line segments respectively [Kaygin and Bulut2002]. A minimal Edit distance method for word recognition has been proposed by [de Waard1995] in which complex substitution costs have been defined for the Edit distance function and string-to-string matching has been done by explicitly segmenting the characters of the words. Graph Edit distance using node combination and splitting has been proposed in [Ambauen *et al.* 2003] for the identification of diatoms (unicellular algae). Another recent variant of Edit distance has been proposed in [Christodoulakis and Brey2008] where apart from the classic substitution costs, two new operations namely combination and split are supported.

The need for an algorithm catering for the merge and split of T-characters arises because we may not have 100% accurate segmentation of T-characters all the time (Figure 4.22).

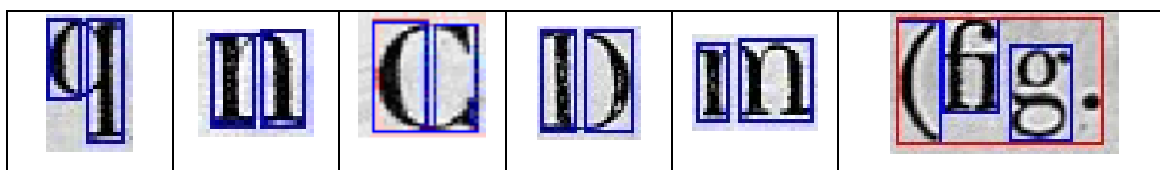


Figure 4.22 - Character segmentation errors (Split as in first five examples and merge as in last example) causing problems in matching stage.

To address the character segmentation issues, we have introduced two new merge-based S-character matching operations **Merge-T** and **Merge-Q** that enable to incorporate and model a **merge** and **split** capability respectively in Edit distance, thus overcoming the limitation of having a perfect character segmentation for good results, by catering for the broken and merged T-characters during word matching.

Consider two words A and B ; A , the query word, having s S-characters while B , the test word, having t S-characters. We treat both words as two series of S-characters, $A = (a_1 \dots a_s)$ and $B = (b_1 \dots b_t)$ where a_i and b_j represent the individual S-characters of the two words. To determine the distance between these two S-character series, we find the Edit matrix W which gives the cost of aligning the two sequences. Apart from the three classic Edit operations that we saw in the previous section, we have introduced two new operations Merge-T and Merge-Q represented as $a_i \rightarrow (b_j + b_{j+1})$ and $(a_i + a_{i+1}) \rightarrow b_j$ respectively. These two operations represent and model the merge and split functionality in the Edit distance respectively. *Merge-T function* allows one S-character a_i of the query word to be matched against two S-characters $(b_j + b_{j+1})$ of the current test word, while *Merge-Q function* allows one S-character b_j of the test word to be matched against two query S-characters $(a_i + a_{i+1})$ thus modeling a split of b_j . Here note that the split functionality is achieved by modeling of the Merge-T function and no split is physically performed. Combination of two S-characters is done by individually concatenating their six feature sequences. These Merge-Q and Merge-T functions help to cater for the segmentation inconsistencies as we will see shortly. The entries of matrix W initialized by +infinity are calculated as:

$$\begin{aligned}
 W(0,0) &= 0 \\
 W(0,j) &= W(0,j-1) + \gamma(\Lambda \rightarrow b_j) && \text{for } (0 < j \leq t) \\
 W(i,0) &= W(i-1,0) + \gamma(a_i \rightarrow \Lambda) && \text{for } (0 < i \leq s) \\
 \\
 &\text{for } (1 \leq i \leq s \ \& \ 1 \leq j \leq t) \\
 W(i,j) &= \min \left\{ \begin{array}{l} W(i-1,j-1) + \gamma(a_i \rightarrow b_j) \\ W(i-1,j-1) + \gamma(a_i \rightarrow (b_j + b_{j+1})) \quad (\text{for } j < t) \\ W(i-1,j-1) + \gamma((a_i + a_{i+1}) \rightarrow b_j) \quad (\text{for } i < s) \\ W(i-1,j) + \gamma(a_i \rightarrow \Lambda) \\ W(i,j-1) + \gamma(\Lambda \rightarrow b_j) \end{array} \right\} \tag{4.6.2}
 \end{aligned}$$

As in classic Edit distance case, Λ represents an empty character for which we have set all the values in its feature vectors to 0. The width of this empty character has been set to 25, which is normally the width of an average character. The three classic Edit operation costs are represented by $\gamma(a_i \rightarrow b_j)$, $\gamma(a_i \rightarrow \Lambda)$ and $\gamma(\Lambda \rightarrow b_j)$ where $\gamma(a_i \rightarrow b_j)$ is the cost of replacing a_i with b_j , $\gamma(a_i \rightarrow \Lambda)$ is the cost of deleting a_i and $\gamma(\Lambda \rightarrow b_j)$ is the cost of inserting b_j . The two new costs are represented by $\gamma(a_i \rightarrow (b_j + b_{j+1}))$ and $\gamma((a_i + a_{i+1}) \rightarrow b_j)$ showing Merge-Q cost and Merge-T cost respectively.

4.6.3.1 Merge-T Cost

$\gamma(a_i \rightarrow (b_j + b_{j+1}))$ shows the cost of replacing an S-character a_i of the query word by two S-characters $b_j + b_{j+1}$ of the test word. It means that if a test word's T-character was broken into two S-characters b_j and b_{j+1} , we would be able to match these with a_i using the Merge-T function. Figure 4.23 shows this case where an S-character of query word is matched against a two S-characters of test word which are infact the broken parts of a T-character.

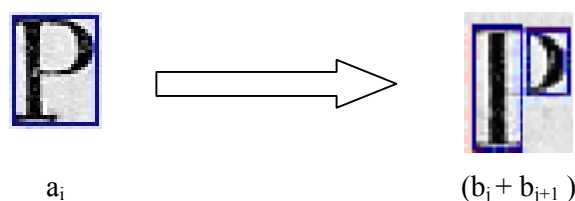


Figure 4.23 - One S-character of query word matched against two S-characters of test word

The feature sequences of b_j and b_{j+1} are concatenated and are matched against the feature vectors of a_i to get the value of $W(i, j)$. Once $W(i, j)$ is calculated, the same value of $W(i, j)$ is copied to the cell $W(i, j+1)$ signifying that we had used the Merge-T function. We keep a count of the total

number of Merge-T functions used as this information will be used in determining the minimum warping path.

4.6.3.2 Merge-Q Cost

Similarly, $\gamma((a_i+a_{i+1}) \rightarrow b_j)$ shows the cost of changing two S-characters of query word to one S-character of test word. It means that if b_j was in fact an S-character comprising two merged T-characters, we would be able to detect and match that with a_i+a_{i+1} using our Merge-Q function. Here, instead of splitting the feature vectors of b_j (which is more difficult as we do not know exactly where to split), we merge the query word S-characters, thus emulating the split function.

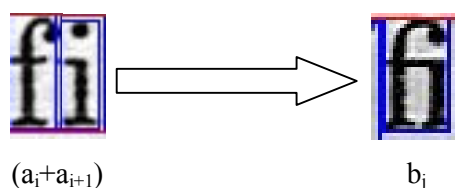


Figure 4.24 - Two S-characters of query word are matched against one test S-character

Once $W(i, j)$ is calculated, the distance value is copied to the next cell $W(i+1, j)$ and the counter is incremented, signifying the use of the Merge-Q function.

The values copied to the next cells in case of Merge-Q and Merge-T functions are not final. The values for these cells are calculated individually as well when their turn comes in the loop. The new value calculated for the next cell is usually greater than the value that was copied before. If this value is lower, we keep this new value and decrement the counter that keeps the count of the merge functions used, by one.

The basic pseudo code of the merge split Edit distance is given in Table 4.6.

Table 4.6 - Pseudo code for the Merge-Split Edit distance algorithm

Input: Two words $A = (a_1 \dots a_s)$ and $B = (b_1 \dots b_t)$; γ

Output: Edit matrix W

Algorithm:

Initialize the whole matrix with infinity

Initialize counter Count_merge_q_t = 0

$$W(0, 0) = 0$$

for $i = 1$ to s

$$W(i, 0) = W(i-1, 0) + \gamma(a_i, \Lambda)$$

for $j = 1$ to t

$$W(0, j) = W(0, j-1) + \gamma(\Lambda, b_j)$$

for $i = 1$ to s

for $j = 1$ to t

$$W(i, j) = \min \left\{ \begin{array}{l} W(i, j) \\ W(i-1, j-1) + \gamma(a_i \rightarrow b_j) \\ W(i-1, j-1) + \gamma(a_i \rightarrow (b_j + b_{j+1})) \quad (\text{for } j < t) \\ W(i-1, j-1) + \gamma((a_i + a_{i+1}) \rightarrow b_j) \quad (\text{for } i < s) \\ W(i-1, j) + \gamma(a_i \rightarrow \Lambda) \\ W(i, j-1) + \gamma(\Lambda \rightarrow b_j) \end{array} \right.$$

if $(W(i-1, j-1) + \gamma(a_i \rightarrow (b_j + b_{j+1})))$ gives the minimum cost value **then**

$$W(i, j+1) = W(i, j)$$

$Count_merge_q_t++$

if $(W(i-1, j-1) + \gamma((a_i + a_{i+1}) \rightarrow b_j))$ gives the minimum cost value **then**

$$W(i+1, j) = W(i, j)$$

$Count_merge_q_t++$

4.6.3.3 Normalization Factor K

Once all the values of W are calculated, the warping path is determined by backtracking along the minimum cost path starting from (s, t) while taking into account the number of Merge-Q and Merge-T functions used in the path way. The normalization factor K is found by subtracting the number of merge functions used in the warping path from the total number of steps in the path.

$$K = \text{Number of steps in the warping path} - \text{Count-merge-q-t}$$

So, the final matching cost of the two words is given by the distance value:

$$\text{Normalized-word-distance} = W(s, t) / K$$

Two words are ranked similar if this final matching cost is less than an empirically determined word-threshold.

Figure 4.25 shows an example of matching two words of lengths 4 and 3. Both words, in fact, have 4 T-characters each but their lengths are different because the query word is well segmented into four S-characters while in the test word, the last two T-characters are merged into one S-character. So the length of test word is 3 instead of 4. While finding W in Figure 4.25, we see that one merge operation is used for matching ‘ur’ with ‘u’, so we increment the value of counter by one.


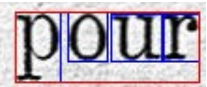
W		Test word 			
		Λ	p	o	ur
Query Word 	Λ	0.00	1.79	3.39	5.56
	p	1.78	0.02	1.62	3.79
	o	3.47	1.72	0.04	2.05
	u	5.51	3.75	2.08	0.09
	r	6.85	5.10	3.42	0.09

Figure 4.25 - Calculating Edit Matrix W for two similar words of lengths 4 and 3

$$\text{Final Word Cost} = W(4,3) / K$$

Steps in warping path = 4 (marked by arrows in the figure)

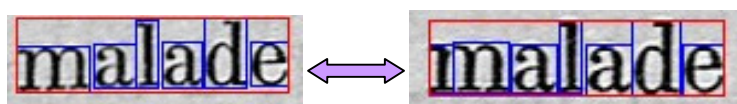
Count of Merge-Split functions used in the warping path = 1

$$W(4,3) = 0.09$$

$$K = (\text{steps} - \text{Count-merge-q-t}) = (4 - 1) = 3$$

$$\text{Final cost} = 0.09 / 3 = 0.03$$

The final cost of matching the two words comes to be 0.03 which is less than the word-threshold (set to 0.20). Lets now consider another case of a word “malade” of Figure 4.21 which was not matched using simple Edit distance as its T-character ‘m’ was split into two S-characters. The matrix W is given in Figure 4.26.




W		Test word 								
		Λ	r	n	a	l	a	d	e	
malade	Query	Λ	0.00	1.54	3.41	5.48	6.82	8.76	10.79	12.53
	m	2.26	0.17	0.17	2.24	3.58	5.52	7.55	9.29	
	a	4.37	2.27	0.55	0.22	1.56	3.49	5.52	7.26	
	l	5.78	3.69	1.96	1.00	0.38	2.00	3.88	5.62	
	a	7.89	5.79	4.07	1.00	1.41	0.41	2.26	4.01	
	d	9.90	7.80	6.08	3.01	1.42	1.68	0.55	2.29	
	e	11.57	9.47	7.75	4.68	3.09	1.64	2.09	0.64	

Figure 4.26 - Calculating Edit Matrix W for two similar words of lengths 6 and 7

Final Word Cost = $W(6,7) / K$

Steps in warping path = 7 (marked by arrows in the figure)

Count of Merge-Split functions used in the warping path = 1

$W(6,7) = 0.64$

$K = (steps - Count-merge-q-t) = (7 - 1) = 6$

Final cost = $0.64 / 6 = 0.106$

The final cost of matching the two words comes to be 0.106 which is again less than the word-threshold. So by using the proposed Merge-Split Edit distance method, we are able to correctly match these two words which was not the case using simple Edit distance method.

We also analyzed the effect of word length on the word-threshold, learning that for short words (with few S-characters), a lower threshold gives better precision and recall rates, while for longer words, a higher threshold value proves to be more effective. This is because for longer words, it is more unlikely to find similar words; so we adopt a more tolerant decision threshold.

This method of using Merge-Split Edit distance coupled with DTW improves matching results immensely as shown later on. The major drawback in this case is the time taken for retrieval.

Especially for long words, the time increases significantly as the matrix size increases along with the number of comparisons between the S-characters. There are numerous S-character comparisons while calculating the matrix W , that never contribute towards the matching of two similar words. For example, the shaded area in Figure 4.27 is the main area of interest while matching two similar words in Edit distance and computing the whole matrix is computationally expensive. To overcome this computation cost problem, another method is proposed where the whole matrix does not need to be calculated and only the ‘interesting’ comparisons are made. We call this method Linear Displacement Matching and it is explained in the next subsection.

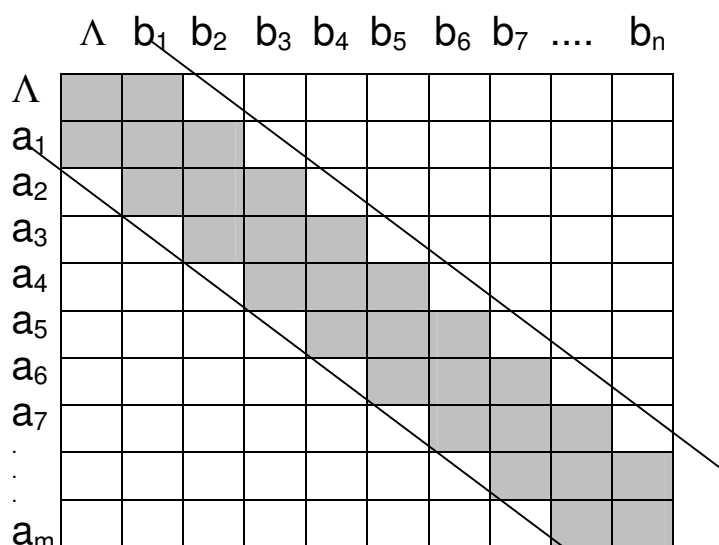


Figure 4.27 - Area of interest for the matching of two words in Edit distance

4.6.4 Linear Displacement Matching

The concept here is simple. For each iteration, three different comparisons are made between the S-characters of the two words as shown in Figure 4.28. These comparisons represent the replace, Merge-T and Merge-Q (modeling a split) operations. The function that gives the minimum cost of the three during the iteration is retained and this minimum cost is added to the total word distance. The S-characters that are used in the minimum cost function are marked and in the next iteration, the immediate next S-characters (to the ones already marked in minimum cost function) are used for comparison. It means that the S-character which has already contributed in the word distance is not used further in the following iterations. Let us see the system in detail.

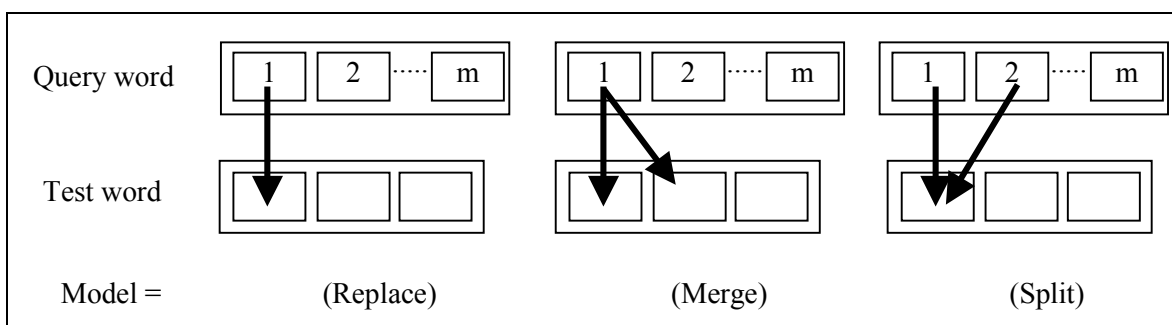


Figure 4.28 - Different operations for linear displacement matching

Consider 2 words A and B ; A , the query word, having s S-characters while B , the test word, having t S-characters. We treat both words as two series of S-characters, $A = (a_1 \dots a_s)$ and $B = (b_1 \dots b_t)$ where a_i and b_j represent the individual S-characters of the two words. To determine the distance between these two words, comparisons are made between the S-characters of the words in a linear way. Three main comparison operations that represent replace, Merge-Q and Merge-T are used to calculate the word distance. The subscripts i and j act as markers (pointers) which point to the current S-characters a_i and b_j of the two words A and B respectively. $(a_i \rightarrow b_j)$ represents ‘Replace’ function for replacing a_i with b_j . Similarly, $a_i \rightarrow (b_j + b_{j+1})$ and $(a_i + a_{i+1}) \rightarrow b_j$ represent the Merge-T and Merge-Q functionality respectively.

The important thing in this method is that an S-character that has already been used in comparison is marked and is not used again for further comparisons. The markers i and j are incremented accordingly after each iteration to keep track of the current a_i and b_j . The increment in a marker depends on the minimum cost operation used for that particular iteration. If the minimum cost in an iteration came from the replace function, meaning that a_i and b_j are matched, then both i and j are incremented by one. Similarly, if the minimum cost in the iteration came from the Merge-T function, then marker i will be incremented by one while j will be incremented by two. It means the S-characters b_j with b_{j+1} are actually two S-characters which are the broken parts one T-character and now these two parts are merged and matched with a_i . In the same way, if this minimum cost came from the Merge-Q function, then i will be incremented by two while j will be incremented by one. It signifies that S-character b_j is actually a combination of two T-characters that are merged into this one S-character b_j . This S-character is thus matched with a combination of two query word S-characters $(a_i + a_{i+1})$. Now if the S-characters of one of the words finish before the other one, then those remaining S-characters are also taken into account for the calculation of total word distance. Depending on which word’s S-characters are left, either the cost of deletion of remaining S-characters in the query word, or the cost of insertion of

remaining S-characters in the test word, is added to the total word distance. The cost of insertion and deletion of S-characters is represented by $\gamma(a_i \rightarrow \Lambda)$ which is the cost of deleting a_i and $\gamma(\Lambda \rightarrow b_j)$ which is the cost of inserting b_j . Simplified pseudo code for the proposed algorithm is given in Table 4.7.

Table 4.7 - Pseudo code for linear displacement matching algorithm

Input: Two words $A = (a_1 \dots a_m)$ and $B = (b_1 \dots b_n)$; γ

Algorithm:

Markers $i=1, j=1$

Word-distance = 0

$K = 0$ (Normalization factor keeping a count of total number of iterations)

while $i \leq m$ **AND** $j \leq n$

if ($\gamma(a_i \rightarrow b_j)$) gives the minimum cost value **then**

$i = i + 1$; $j = j + 1$;

if ($\gamma(a_i \rightarrow (b_j + b_{j+1}))$) gives the minimum cost value **then**

$i = i + 1$; $j = j + 2$;

if ($\gamma((a_i + a_{i+1}) \rightarrow b_j)$) gives the minimum cost value **then**

$i = i + 2$; $j = j + 1$;

Word-distance = *Word-distance* + minimum cost value

$K++$;

while $i < m$

Word-distance = *Word-distance* + $\gamma(a_i \rightarrow \Lambda)$

$K++$; $i++$;

while $j < n$

Word-distance = *Word-distance* + $\gamma(\Lambda \rightarrow b_j)$

$K++$; $j++$;

For the remaining
S-characters

Word-distance = *Word-distance* / K

Now to normalize the word distance, we take into account the total number of iterations which contributed to the total word distance.

$$K = \text{Number of iterations used in Word-distance}$$

So, the final normalized distance between the two words is:

$$\text{Normalized Word-distance} = \text{Word-distance} / K$$

Two words are ranked similar if this final matching distance is less than an empirically determined word-threshold. Figure 4.29 shows an example where the first iteration of a word matching scenario is analyzed.

Query word <i>A</i> with 3 S- chars (F,I,G)			
Test word <i>B</i> with 2 S- chars (FI, G)			
Operations	Replace	Merge-T	Merge-Q
	$\gamma(a_1 \rightarrow b_1)$	$\gamma(a_1 \rightarrow (b_1 + b_2))$	$\gamma((a_1 + a_2) \rightarrow b_1)$
Operation cost	0.86	1.65	0.07
Markers	$i=i+1, j=j+1$	$i=i+1, j=j+2$	$i=i+2, j=j+1$

Figure 4.29 - First iteration of the algorithm for matching the S-characters of two words. We can see that Merge-Q, the operation of matching S-characters ‘F’, ‘I’ of the query word with S-character ‘FI’ of the test word, yields the minimum cost

From the Figure 4.29, we can see that the cost of matching the S-characters *F* and *I* of the query word with the S-character *FI* of the test word gives the minimum cost. Thus the markers *i* and *j*

are incremented accordingly. In the second iteration, G of query word will be matched with G of test word. So we can have the matching result only in two iterations.

The major advantage this method has over merge split Edit distance is that though it might suffer just a slight performance loss in recognition (details in results section), computationally it is much faster than the Merge-Split Edit distance method. It is because here only the most relevant comparisons are made between the S-characters of the two words. In Edit distance method, there were plenty of comparisons which never contributed towards the matching of two similar words as was shown earlier in Figure 4.27. In linear displacement matching method though, only the relevant comparisons are made which enhances the performance of the system and makes it computationally very efficient as compared to the Merge-Split Edit distance method.

4.6.5 Computational comparison: Linear Matching and Merge-Split Edit distance

As we will see in the results section, the performance of linear displacement matching is pretty close to Merge-Split Edit distance but there is a significant improvement in the retrieval time. Data set A consisting of 20 document images, taken from 5 different books has been used for exploring the computational performance of the two methods. Different query words of different lengths are selected and searched in the data set. It allows us to give an indication of how these methods fare computationally for different query lengths. Does query length makes a difference in the retrieval time or is it query length independent? Average search time per 100 words has been calculated and the graph is plotted for this average search time (average time to match a query word with 100 words) on an Intel core2duo 2.1GHz machine with 3GB RAM.

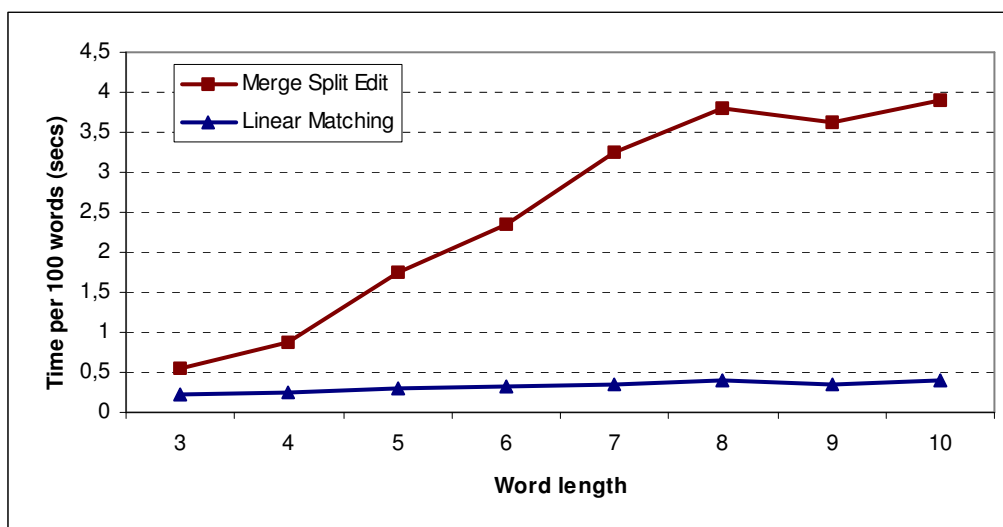


Figure 4.30 - Time taken per 100 words searched, by linear displacement matching and Merge-Split Edit distance, for query words of different lengths

From the graph in Figure 4.30, we can see that for Merge-Split Edit distance method, the time taken to match two words increases with the increase in length of query word while for linear displacement matching algorithm, time remains almost the same for different query lengths. It emphasizes the point we made earlier about Merge-Split Edit distance that the number of possible matches between the S-characters of two words increases with increase in query length and thus it takes more time to compute the whole matrix. So linear displacement matching can be preferred when high level of computational efficiency is required even if it comes at the cost of a little compromise in performance.

Now we see in detail the comparison results of the different string comparison methods studied so far to evaluate their performances.

4.7 Experimental Results

All experimentation and the comparison analysis of different method has been performed on BIUM document images [BIUM] (data sets described in chapter 1). Different experiments have been performed:

- for performance evaluation of each of the word matching algorithms proposed earlier
- for comparison of these methods with state-of-the-art in literature as well as with professional OCR software
- and for performance analysis of our S-character feature sequences to see which of them have higher contribution in correct matching of the words

All these experimental details and results will be discussed here in turn, beginning with the description of the performance measures that we have used for evaluation.

4.7.1 Performance Measures

Different performance measures have been used for feature performance evaluation and also for the evaluation of word matching methods. These are described here.

- **Precision**

Precision P is the percentage of the retrieved words that are relevant to user's information need. Here by relevant, we mean the words that exactly match the query word. Precision is defined as:

$$P = \frac{\text{SameWords Retrieved}}{(\text{SameWords Retrieved} + \text{False Positives})} \times 100$$

If all the retrieved words are similar to the given query (even if they are not the total number of similar words existing in the data set), precision will be 100%.

- **Recall**

Recall R is the percentage of the words, same as query, that are successfully retrieved from the data base.

$$R = \frac{\text{SameWords Retrieved}}{\text{Total SameWords Existing}} \times 100$$

If all the same words, to the given query, existing in the data base are retrieved (even if there are plenty of false positives along with), the recall will be 100%.

- **F-measure**

The harmonic mean of precision and recall is the traditional F-measure or F-score¹³. It is calculated as:

$$F = \frac{2 \cdot P \cdot R}{(P + R)}$$

The higher the value of F-measure, the better is the performance of the system.

- **R-score**

Sometimes, in addition to the exact same words, we need to find all relevant words which are in fact some morphological variations of the given query i.e. words having similar root to the given query. For example, if we have a query “stream”, then its variant words such as “streams” or “streaming” are considered relevant to the query. To take into account these retrieved words that are not exactly same as the query word but are very much relevant/similar to the query (they cannot be counted as false positives in information retrieval applications), we have introduced a new index called Relevance measure (R-measure) or R-score. We define it as:

$$R\text{-score} = \frac{\text{Relevant Words Retrieved}}{(\text{Relevant Words Retrieved} + \text{False Positives})} \times 100$$

¹³ http://en.wikipedia.org/wiki/Information_retrieval#Performance_measures

If this percentage is high, it means that most of the not-same-to-query words retrieved are similar and relevant to the query. The decision of awarding a word in the relevant category depends totally on the user requirements and application. In our case, if we are looking for a word and we find a similar word having similar root, we count it as relevant. Figure 4.31 gives an example.

Query	Retrieved word	Relevant / False positive
rhinoscopie	rhinoscopique	Relevant
	<i>laryngoscope</i>	False Positive
introduit	introduction	Relevant
	produit	False Positive

Figure 4.31 – Example of relevant word and false positive word for given queries

We use the above described measures for the evaluation of our different experiments.

4.7.2 Performance evaluation of the different feature sequences

Here we will discuss the descriptive powers and the contributions of the six feature sequences proposed earlier. Data set A which consists of 20 document images, taken from five different books has been used for exploring the performance of the six features. For testing, 25 different query words having 175 instances in total are selected, based on their varied lengths, styles and also context of the book.

The performance of each feature was tested individually at six different thresholds, using the Merge-Split Edit distance method, to see how their impact varies with changing thresholds. For every feature at each threshold, we calculated precision and recall percentages as well as the F-scores. Figure 4.32 and Figure 4.33 show the Precision versus Recall curve and the F-score curve respectively.

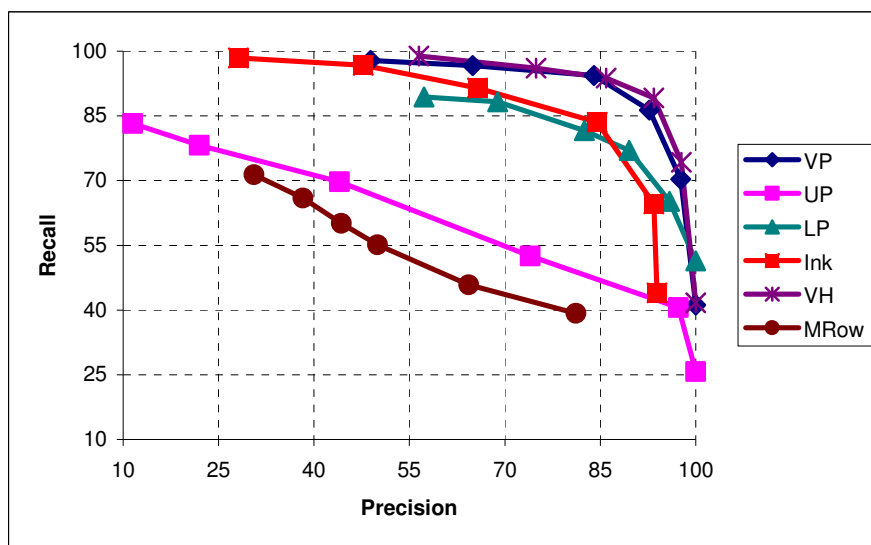


Figure 4.32 - Precision vs Recall at different thresholds for the six features

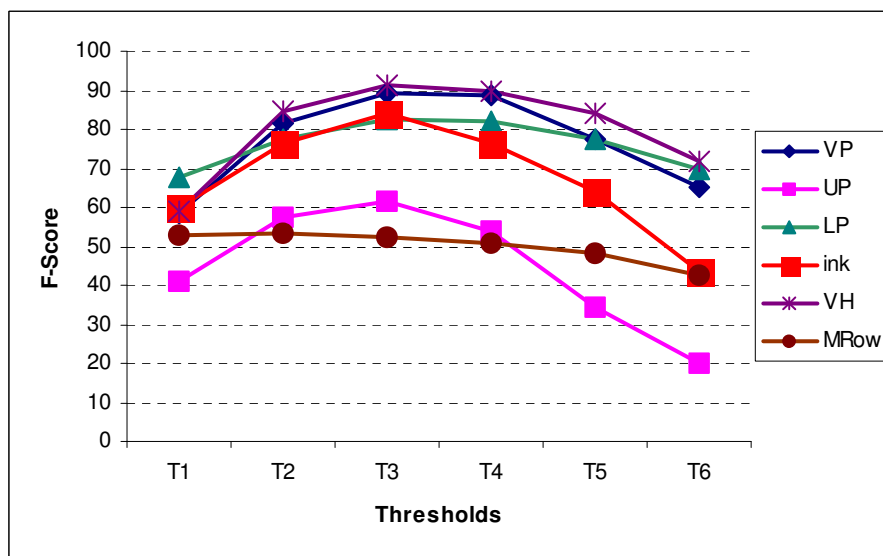


Figure 4.33 - F-Scores for different features at different thresholds

From the graphical results above, we can see that among the features, vertical projection performs the best overall. Since all these features are designed to respond to different shape characteristics of the S-character, each feature has a different response to the changing threshold values. All these features complement one another to some degree, and thus the results improve significantly when all of these features are combined (as we will see in Figure 4.39 shortly). We also tried using different combinations of better performing features but the best results are achieved if we use all six of the features.

4.7.3 Comparison of different Word matching methods

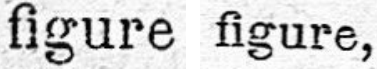
Here we will discuss in detail the comparison results of different word matching methods.

4.7.3.1 Data Set

Data set B which consists of 48 document images, taken from 12 different books and having a total of 17,010 words, has been used for exploring and comparing the different word matching methods. 60 different words having 435 instances in total are selected as query based on their varied lengths, styles and also context of the book. Some of these word instances are shown in Table 4.8. All the experiments have been performed on an Intel core2duo 2.1GHz Windows 2000 machine with 3GB RAM.

Table 4.8 - Different instances of query words from 12 different books

1	rhinoscopie <i>rhinoscopie</i> rhinoscopie
2	FIG. FIG. FIG. FIG. FIG. FIG.
3	ampoule ampoule ampoule ampoule
4	malade malade malade malade
5	cheveux cheveux cheveux cheveux
6	Fig. Fig. Fig. Fig. Fig. Fig.
7	ventricule ventricule ventricule,
8	PATHOLOGIQUE. PATHOLOGIQUE. PATHOLOGIQUE.
9	réglage réglage réglage
10	diabétique diabétique diabétique
11	Photographie photographie Photographie

.	
.	
.	
60	

4.7.3.2 Word Retrieval methods comparison - Results

The comparison of the different word retrieval methods proposed earlier is given here in Table 4.9. In addition to the perfectly detected words, we also give the number of good words which remain undetected (words missed), the relevant words detected as well as the false positives.

Table 4.9 – Experimental analysis of the four proposed string comparison methods

	RPC	Edit distance	Merge-Split Edit distance	Linear Matching
#query word instances	435	435	435	435
#words detected perfectly	401	406	427	420
#words missed	34	29	8	15
#relevant words detected	99	53	39	33
#false positives	51	16	4	3

Using RPC method, we have detected 401 words similar to the query and are not able to detect the remaining 34 word instances. Positive aspect of this method, though, is the large number of relevant words that are retrieved along with. It is useful for the applications where we don't need to match exact query words and are satisfied with almost similar (relevant) words as well. On the other hand, the number of false positives is huge which is a major drawback of this method. Using Edit distance method, the number of false positive is reduced significantly (16 now as compared to 51 in RPC).

The best recognition rate has been achieved by the Merge-Split Edit distance method where we are able to correctly detect 427 query word instances along with other 39 relevant words. The number of false positives in this case is only 4 showing the efficiency and robustness of the method. As compared to Merge-Split method, the linear method detects one less false positive but is unable to detect 15 query instances, 7 more than the Merge-Split Edit distance method.

To compare these results with existing methods and see how our proposed methods perform with respect to the state of the art, we implemented and tested the method of [Rath and Manmatha2007] on the same data set. In that method, four feature sequences are found out for word images and two words are compared by matching these features using DTW algorithm (details were given in the state of the art chapter). We also tested the data set using the

commercial OCR software ABBYY fine reader [ABBYY] to see how our method fares when compared to a professional level software. Table 4.10 gives the results of these experiments.

Table 4.10 - Results using word feature based method of *Rath and Manmatha*, and ABBYY OCR software

	<i>Rath et al. 2007</i>	ABBYY Fine Reader
#query word instances	435	435
#words detected perfectly	335	422
#words missed	100	13
#relevant words	66	0
#False positives	54	0

From the experimental data, we can see that the word feature based method [Rath and Manmatha2007] is able to correctly identify only 335 query instances thus missing out 100 of them. The number of false positives is very high as well which enforces the fact that the feature matching at word level does not give as good results as S-character level features [Khurshid *et al.* 2008a]. We also tried the method of Rath by using our six features instead of the proposed four features and the results were immediately better [Khurshid *et al.* 2008a, Khurshid *et al.* 2008b], showing that our feature set is more robust and efficient than the one used in [Rath and Manmatha2007]. For the results using ABBYY Fine reader, there is no problem of false positives as it finds only the exact words. It means that there are no relevant words as well in the OCR. Overall, it is able to detect 422 query instances correctly and couldn't detect the remaining 13.

All the experimental results have been computed for optimum matching thresholds, which are determined empirically, for different methods. Now once we have this experimental data, we can evaluate all the methods using the performance measures defined above. These measures will indicate how these methods compare against each other. We will see each measure in detail.

a. Recall

One of the most important performance measure for a word retrieval system is its recognition rate or recall rate. It is extremely important for a retrieval system to have a good recall rate, which will make sure that users get the required information. From all the methods discussed above, Merge-Split Edit distance achieves the best recall rate of all reading up to 98.16%. The computationally efficient linear displacement matching method achieves 96.55% which is not as good as the Merge-Split Edit distance method, but is very acceptable considering the computational efficiency of the method. Figure 4.34 shows the recall rates for different methods.

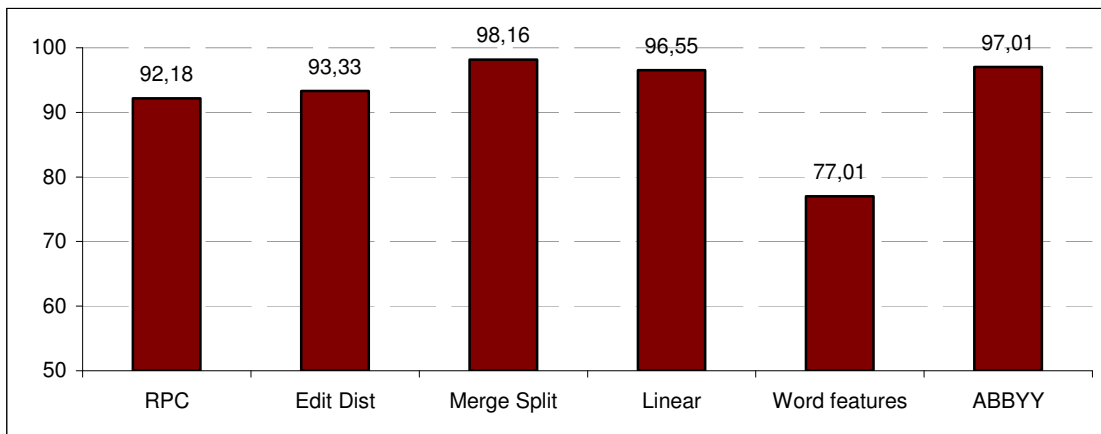


Figure 4.34 - Recall rates for different methods on our data set

For the method of [Rath and Manmatha2007], we are able to get a recognition percentage of 77% which is very less when compared with the other S-character feature based methods. It reinforces our point that S-character features are better able to represent the word image as whole as compared to word features. Even if we are not able to extract and segment the T-characters properly, we can still achieve better results by using the Merge-Split Edit distance or the linear method.

Using the ABBYY OCR, we get a recognition rate of 97% which is less than our Merge-Split Edit distance method thus highlighting the advantage our method has over the professional OCR software. The OCR software wasn't able to read some of the characters in the searched words due to poor document quality. Some of the words were even detected as graphic and not as text. There were other occasions where a whole part of text was not read properly by the OCR when the text was either in caption or was written in a small font, thus affecting the resolution. These problems are highlighted in the Figure 4.35 and Figure 4.36.

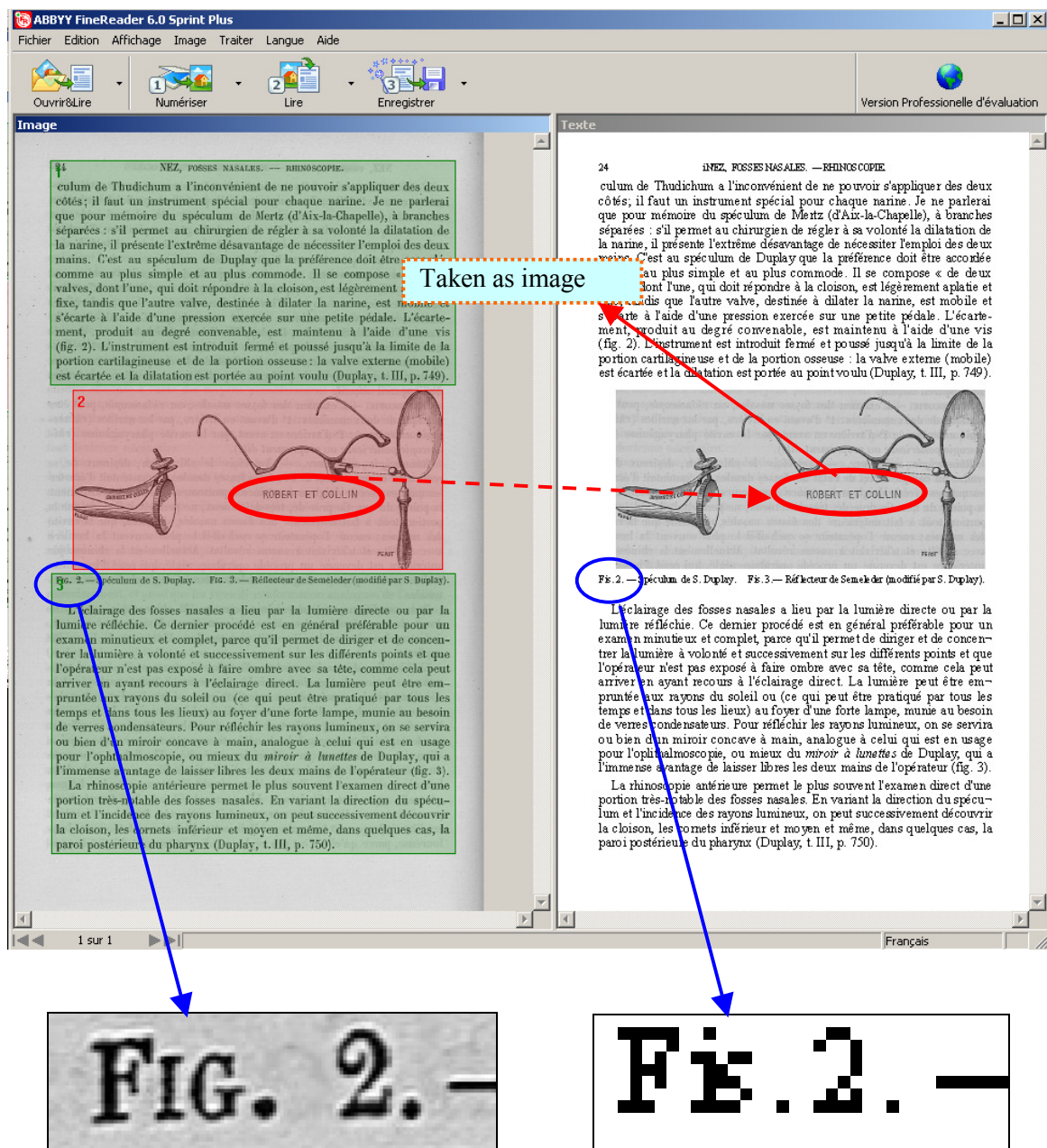


Figure 4.35 - OCR result for a page where green zones highlight the text areas while red zones highlight the graphics. OCR problems : word 'FIG' has been misread as 'Fis', while some words are taken as part of graphics and not as text

In the Figure 4.35, we show an example where the query word instance 'FIG' was read as 'Fis'. So OCR was not able to retrieve this word. All the 13 words that are missed by the OCR are due to similar problems. Another example, where text has been recognized as graphic, can be seen in Figure 4.35. None of our query word instances belonged to any of the text taken as graphic by OCR, or else the results of the OCR software could have been worse. Another example is shown in Figure 4.36 where whole part of the text, printed in smaller font, has been misread completely.

Using our proposed methods though, we are able to recognize the words (such as figure, leviers, etc.) in that small font text line as well.

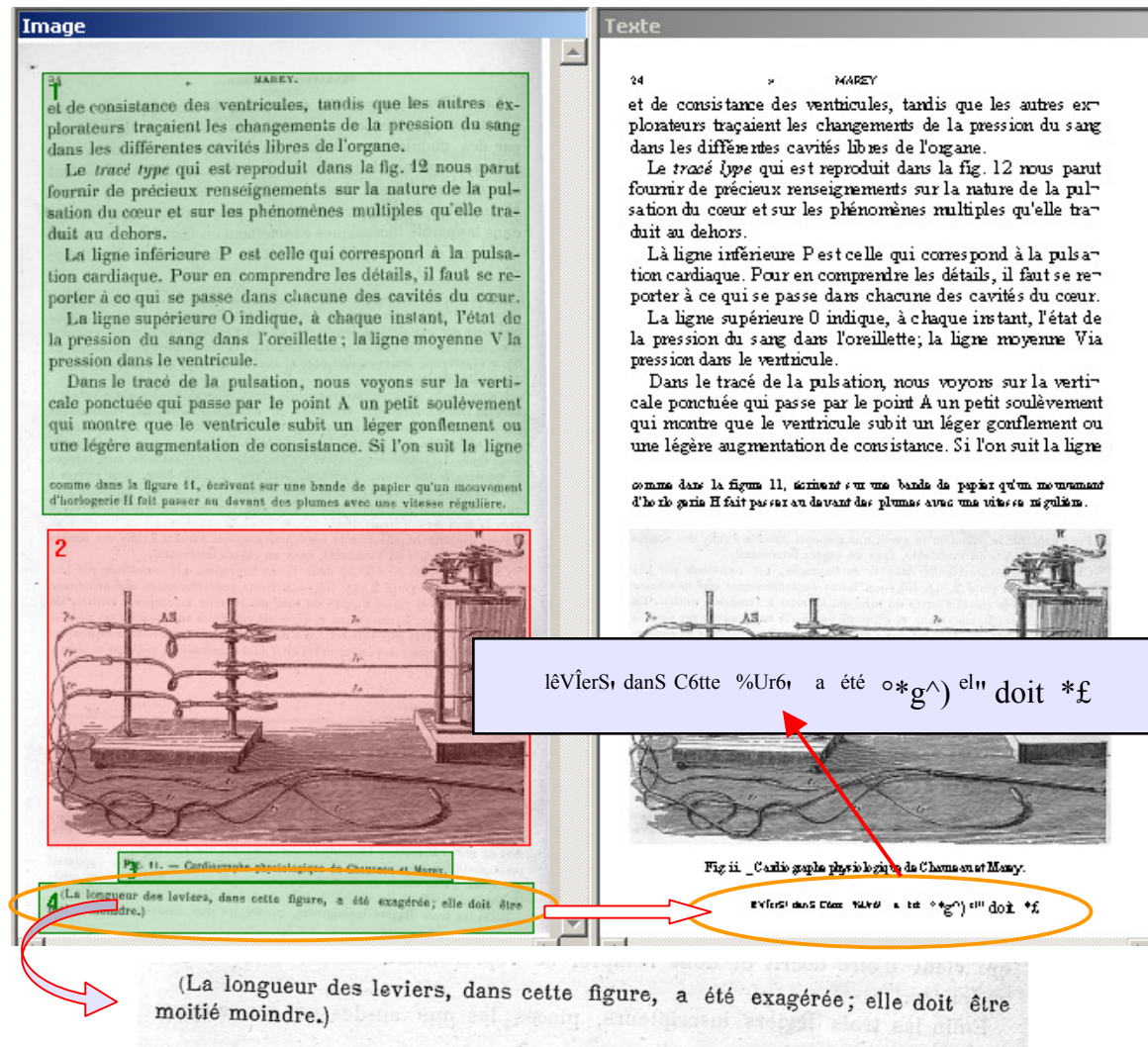


Figure 4.36 - Label text completely misread by the OCR

The recall percentage of OCR could have been lower if the selection of query words was non neutral. But we selected the query words neutrally by considering the importance and *relevance-to-book* perspective thus representing fair comparison results.

b. Precision

Precision is also a very important measure for a retrieval system. If precision percentage is low (even if the recall rate is good), there will be a lot of unwanted information for the user which serves nothing and wastes time. So precision of a system need to be high so that the user is not flocked with unwanted information along with the required parts. The precision percentages for the different methods tested is given in Figure 4.37.

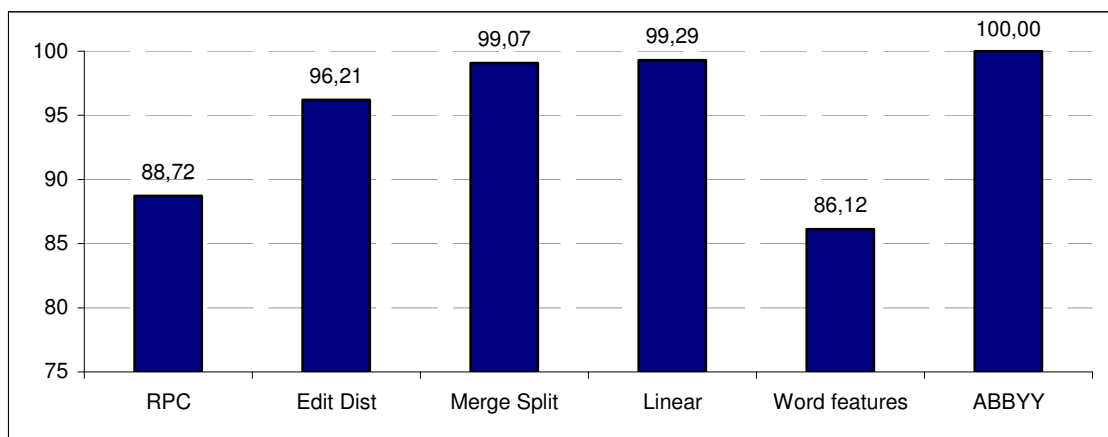


Figure 4.37 - Precision percentages for different methods

We can see from the graph that ABBYY OCR achieves a precision of 100% for our data set. It is because it retrieves only the words where all characters of the words are supposed to be well matched. There are no false positives or relevant words, but only similar words.

The Merge-Split Edit distance achieves a percentage of 99.07% while linear displacement matching has a precision of 99.29%. For RPC and word feature matching though, this percentage is below 90% thus highlighting the different shortcomings of these methods. For word feature method, lack of precision is due to the fact that when two words, which differ only in a couple of characters, are compared, their profiles are almost similar and thus the final distance value between these two words comes out to be low. It will not be the case when feature matching is done at individual character level and then string matching at word level.

c. F-measure

F-measure or F-score gives an overall measure of the system as it takes both precision and recall into account.

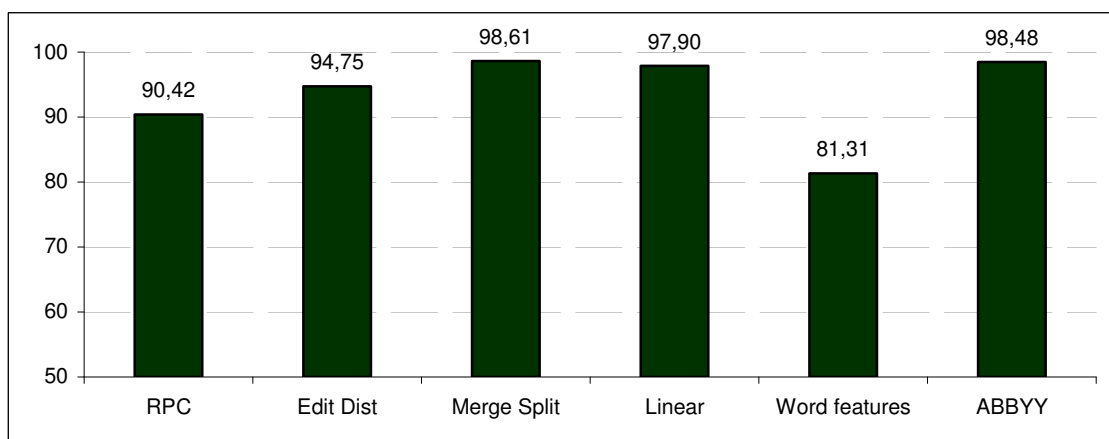


Figure 4.38 - F-scores of different methods

From precision, recall and F-scores, we can see that overall, the Merge-Split Edit distance method outperforms the rest. The main reason for that is its ability to match words, independent of the requirement of having good character segmentation. Thus the words, where T-characters are not segmented properly, are matched correctly as well. We also analyzed the Merge-Split method at different threshold values to see the variations of F-score as well as precision versus recall curve shown in Figure 4.39 and Figure 4.40.

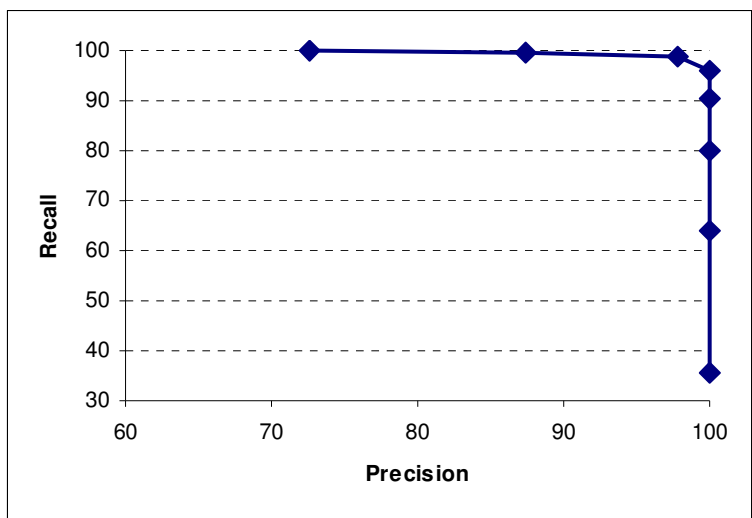


Figure 4.39 – Merge-Split Edit distance - Precision vs Recall at 8 different thresholds between 0.4 and 1.1

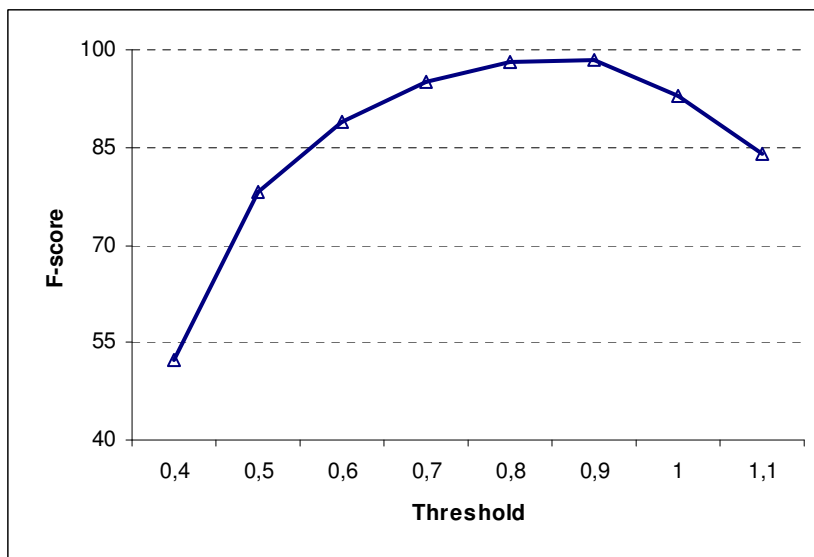


Figure 4.40 – Merge-Split Edit distance - F-score at different thresholds

From the analysis of the curves in Figure 4.39 and Figure 4.40, we can see that the Merge-Split Edit distance method is not very sensitive to the threshold variations. There is a relatively large “safe range” of thresholds for which the system gives equally promising results, thus signifying the robustness of the method.

d. R-score

One aspect of a retrieval system that is often neglected is its capability to retrieve words that are not entirely same as the query word but are almost similar in composition. Often, the systems retrieve extra words along with the searched words but how many of those extra words are relevant and how many are totally unwanted or false positives? This answer is given by this new measure called the Relevance measure which demonstrates the capability of a system to retrieve relevant words along with the exact words in response to the given query. R-scores for different methods are given in Figure 4.41.

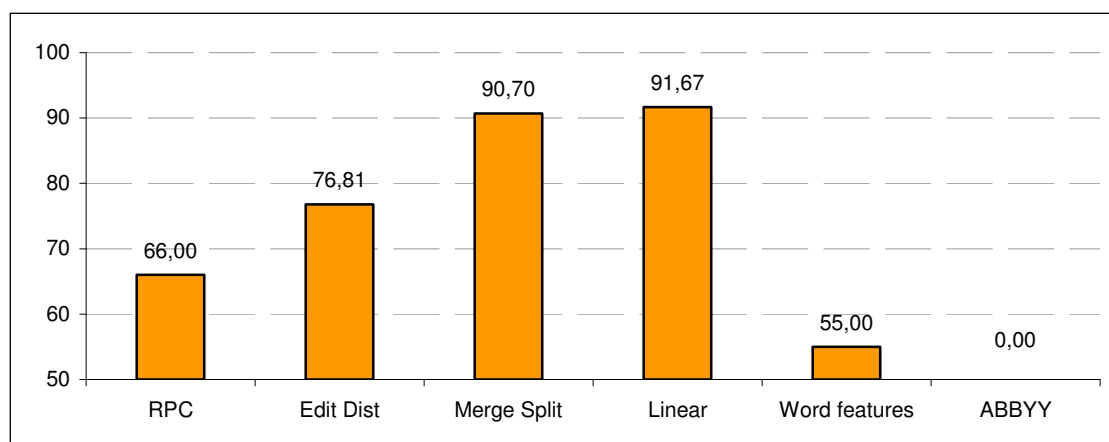


Figure 4.41 - R-scores showing the relevant word matches against total extra words detected

From the Figure 4.41, we can see that the linear displacement matching method and the Merge-Split Edit distance method have very impressive R-scores. This means that whenever they retrieve some extra words, more often than not, these extra words are relevant words and only a few times do we get some false positives. Thus the user is not flocked with unwanted data and he gets only the relevant things. For ABBYY OCR, this percentage is zero as it only gives the exact matches and no extra words (neither relevant nor false positives) are detected.

From the above discussion, we can conclude that our S-character feature based methods perform very well and especially the Merge-Split Edit distance and the linear displacement matching methods outperform others because of their capability to overcome the segmentation problems. The performance of the ABBYY OCR software is not bad either on these document

images which brings us to a question: *what is the gain of our implementation?* To respond to this question, we analyzed and tested much older document images from the 16th century where the performance of OCR was extremely poor. The details of the experiments are given here.

4.7.4 Experimental analysis using highly degraded 16th century documents

Three books from the 16th century are used for the performance comparison between our implementation and ABBYY OCR software (Sample images in Annex A.2). Portion of a document image from one of the books is given in Figure 4.42. We can see that the text is very hard to read for a non expert of the language. When the transcription of these document images is done using ABBYY, the results are extremely poor. OCR software is not able to properly extract the text areas and even if it did, there are lots of errors in character recognition.

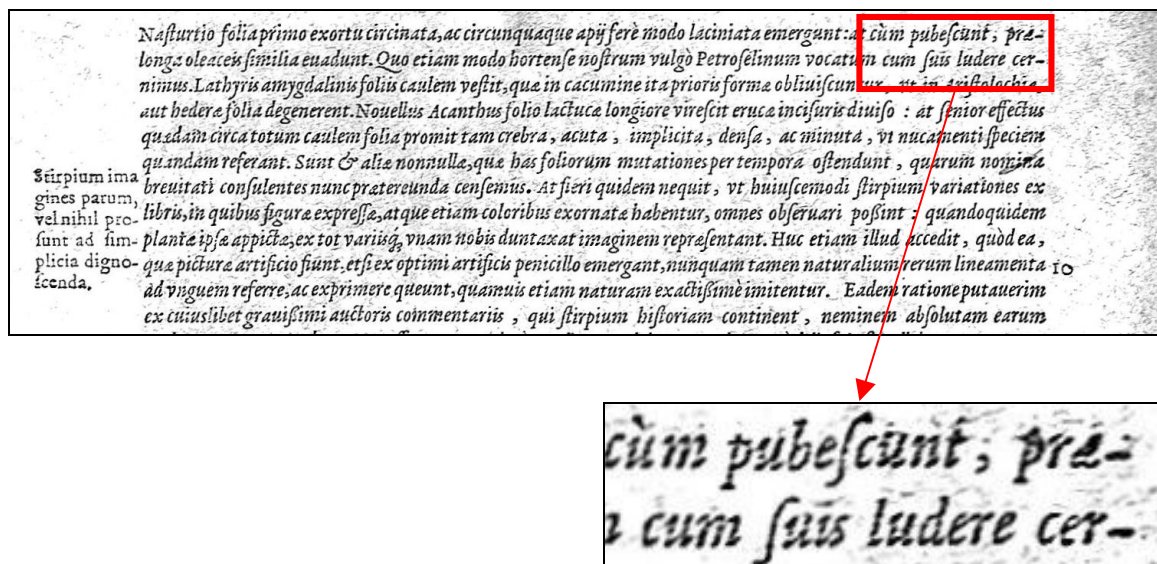


Figure 4.42 - Portion of a 16th century document image - difficult to read even from the naked eye due to document quality and also because of the old font styles

Figure 4.43 and Figure 4.44 show a couple of screens of the recognition results using ABBYY OCR software. In both the figures, we can see that some of the text areas have been wrongly identified as graphics (highlighted in red) while the text that has been extracted (green areas) itself contains many errors. That's why OCR is not feasible for information retrieval purposes in these types of documents. Text detection using the proposed method on the same image yields much better results than OCR as shown in Figure 4.45.



Figure 4.43 - Extraction of text areas using ABBYY. Some text areas are taken as graphics (green parts are segmented as text areas while red parts highlight the graphics)

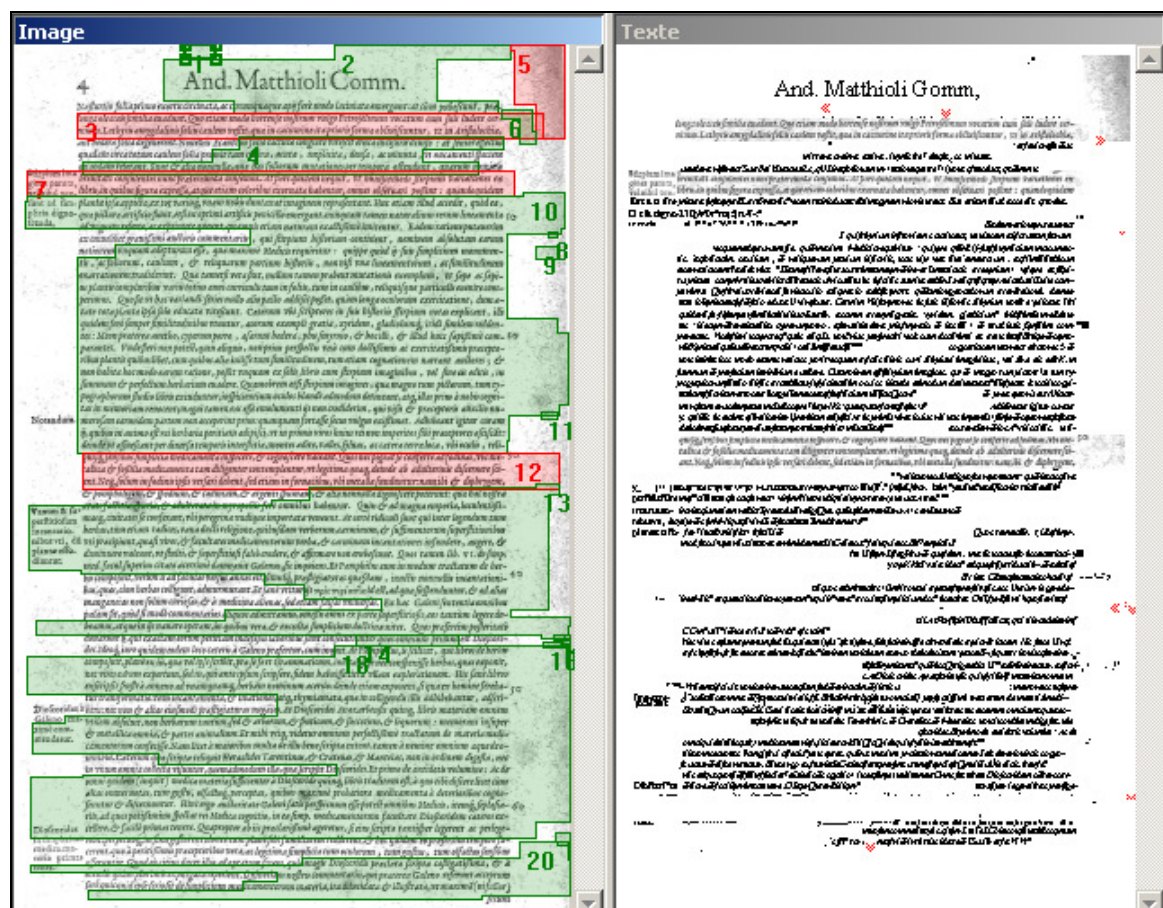


Figure 4.44 - Another example of text extraction and recognition by ABBYY software

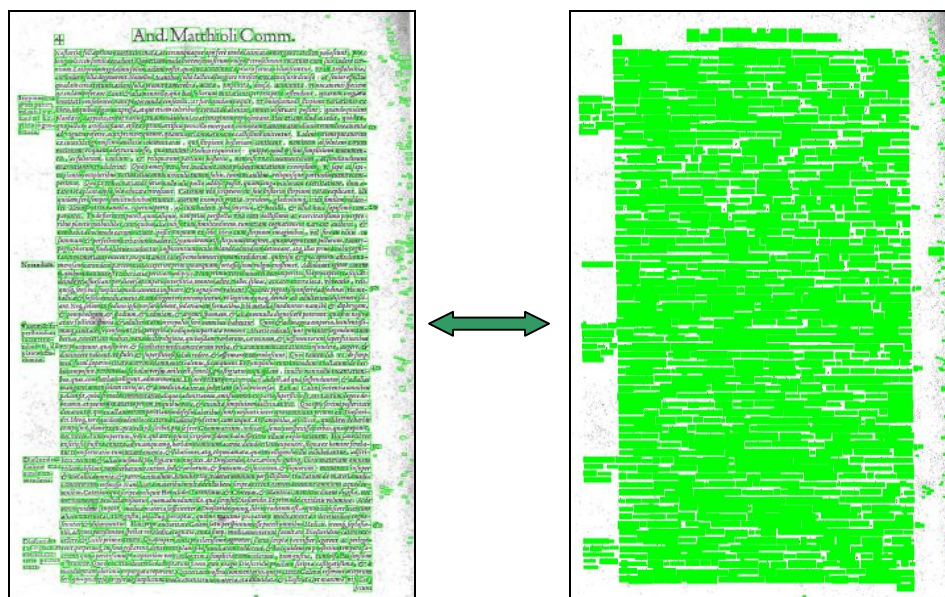


Figure 4.45 - Text extraction using the proposed method. All text areas are detected

As we can see in the document images, the text is sometimes unreadable even for the humans who are not familiar with these kinds of characters. Moreover, words are very close to each other and sometimes there is no spacing at all between them which makes word segmentation very hard. Figure 4.46 shows an example where words and S-characters are segmented in one of the document images using the approach discussed in chapter 3, with H-RLSA threshold value, determined empirically on the test data, set to five. The segmentation of words in these document images is not perfect, thus affecting the recognition rates of the system which we will see shortly.

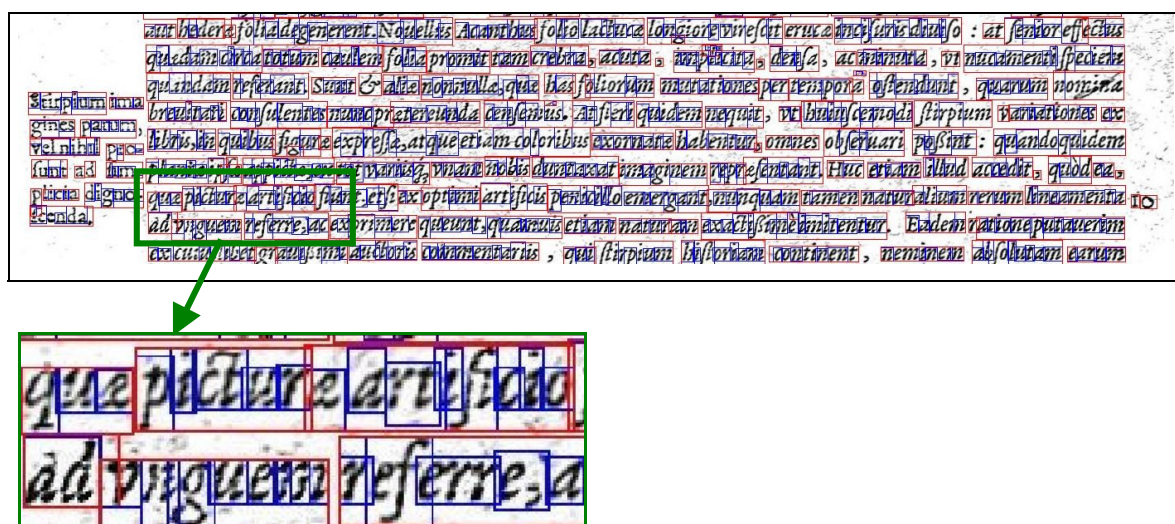


Figure 4.46 - Segmentation of words (in red bounding boxes) and S-characters (in blue bounding boxes)

To test our word matching system to analyse how it performs on these old documents, and to compare our results with OCR results, we selected a total of 12 document images, four from each of the three ancient books, with each page having on average 1200 to 1250 words. For each book, five different query word images of varied lengths and styles, are selected which are then searched in the respective books. The results are formulated and performance of the system is evaluated based on the word recognition rates (recall percentages). Table 4.11 summarizes the number of the exactly matched words (*Mat*), the relevant words (*Rel*) and the false positives (*FP*) using OCR software [ABBYY], Edit distance, Linear displacement matching method and the Merge-Split Edit distance method.

From the Table 4.11, we can see that the OCR results are extremely poor, reading a recall rate of merely 52.04%. Similar recall rate (51.46%) is observed for the Edit distance matching method at 77.87% precision. On the other hand, both Merge-Split Edit distance and linear displacement method achieve very reasonable results with recall rates of 87.13% and 85.38% respectively, at a precision of 86.63% and 88.48% respectively. A total of 78 relevant words are retrieved by Merge-Split Edit distance method while linear method retrieves 74 relevant words apart from the exact words. The OCR neither detects any relevant words nor any false positives, thus eliminating the need of having explicit columns of *Rel* and *FP* for OCR, in the results table.

The experimental results confirm the fact that for poor quality ancient document images, our approach of working at S-character level using the proposed feature set achieves far better results of information spotting than the commercial OCR software. When we analyze the results, we learn that there aren't many drawbacks in the matching stage of our system for comparing two well segmented words, even if they are printed in ancient font styles. Some problems lie in the word segmentation stage though (see Figure 4.47), which causes the recognition rate to drop a bit. The recall percentage of our system would be higher if we could have a better word segmentation, even if the individual characters of the words are not well segmented. This is due to the fact that the character segmentation issues have already been dealt with in our Merge-Split Edit distance or linear displacement methods. For word segmentation errors, e.g. if couple of words are merged together in one component, it is unlikely that the proposed system will recognize them as it does not search small sub-words within a large word component.



Figure 4.47 - Word segmentation problems: Some words are merged together as the spacing between them is very low

Table 4.11 - Performance evaluation of the different methods and the professional OCR software based on their recognition rates

Query	Total Instances	# Words retrieved by									
		OCR	Edit distance			Linear Matching			Merge-Split Edit		
		<i>Mat</i>	<i>Mat</i>	<i>Rel</i>	<i>FP</i>	<i>Mat</i>	<i>Rel</i>	<i>FP</i>	<i>Mat</i>	<i>Rel</i>	<i>FP</i>
<i>omnibus</i>	9	5	6	4	0	7	3	0	8	6	0
<i>que</i>	44	1	34	12	11	38	24	9	39	24	10
<i>medicamenta</i>	5	1	1	0	0	4	1	0	4	1	1
<i>qui</i>	22	14	8	24	10	18	37	2	18	39	2
<i>quippe</i>	8	4	6	0	0	7	0	0	7	0	1
<i>mēbranę</i>	2	1	1	1	0	1	1	1	1	1	0
<i>ab</i>	6	5	6	0	0	6	0	0	6	0	0
<i>substātia</i>	2	1	1	0	0	1	0	0	1	1	0
<i>cerebri</i>	24	24	5	0	3	22	0	2	23	0	3

4 – Information Retrieval - Word Spotting

herbes	2	2	2	0	0	2	0	0	2	0	0
herbis	3	1	1	0	0	3	2	0	3	2	1
lache	1	0	1	0	0	1	0	0	1	0	0
rhabarbe	16	12	4	0	1	13	1	0	13	1	1
odeur	13	7	10	0	1	11	1	1	11	1	1
etiam	14	11	2	0	1	12	4	4	12	2	3
TOTAL	171	89	88	41	25	146	74	19	149	78	23
Recall %		52.04%	51.46%			85.38%			87.13%		
Precision %		-	77.87%			88.48%			86.63%		

These kinds of problems could be handled by further breaking the merged words into multiple overlapping word components using some sort of window-based technique and then matching each window word individually with the query word using the proposed matching method. We believe that, though it will slow down the retrieval task considerably (as query word will be matched with the whole test word and then its sub-words), it will improve the recognition rate by overcoming the segmentation problems caused by poor quality and irregular printing style of the old documents. Even now, the results signify the effectiveness of our S-character's feature based word spotting using dynamic programming for the documents where our system outperforms the OCR software by a long way.

4.8 Conclusion

In this chapter, we explained in detail the proposed methods for word and S-character matching, thus enabling us to retrieve required information from a document base. The S-character features are matched using elastic dynamic time warping method which is scale and translation invariant. Evaluation of the different features proposed has also been discussed to see which features contribute more towards the results. Different methods have been employed to match the strings of S-characters for word retrieval. All experiments have been carried out using the BIUM document base.

From the results analysis, we can conclude that our S-character feature based methods perform very well and especially the Merge-Split Edit distance and the linear displacement matching methods outperform others because of their capability to overcome the character segmentation problems. We tested the method on very old poor document images as well and got much better recognition results than the commercial OCR software [ABBYY]. We have also tried the Merge-Split Edit distance method on different other types of documents in different languages (explained in the next chapter) and the performance is pretty good for those as well, thus highlighting the adaptability and robustness of the proposed matching method. There is always some room for improvements though in any system as Steinherz in [Steinherz *et al.* 1999] writes and we quote it here '*We would like to point out that we may be close to the limits of stand-alone word recognition, and efforts should be made to improve post-processing techniques that will take advantage of syntax, context, and other external parameters.*' In the last chapter of the thesis, we will point out and discuss different possible improvements that can be made to the system in future perspectives.

Chapter 5

Multi-context Applications

It is always good to have a model which can be applied to different systems in multi-varied contexts. We also tried our information retrieval model for different applications with extremely satisfactory results. In this chapter, we will highlight a couple of applications and will show how our system performs for these applications. The applications we discuss here include figure/caption retrieval, contemporary documents analysis and cursive oriental text retrieval.

5.1 Figure/Captions Retrieval

Indexing books to build digital libraries is most often done manually as no automatic method is yet adapted to the needs of the archivists. One of the indexing tasks is to extract figures with the associated captions and to save them in the table of figures. Figure 5.1 shows a screenshot of the digital medical library Medic@ [BIUM] with manual indexing of figure/caption pairs in a book. We propose to facilitate the task of archivists with an automatic detection of figure captions [Khurshid *et al.* 2009a].

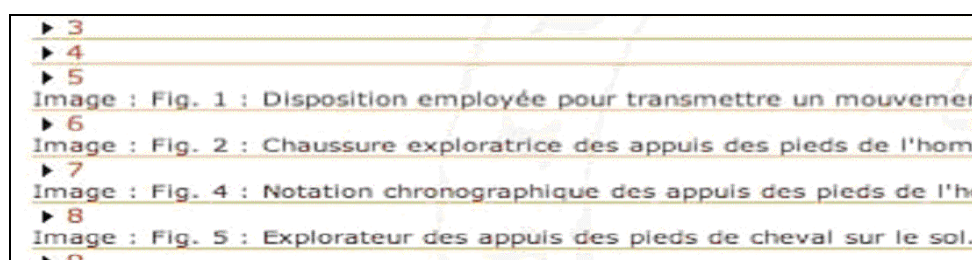


Figure 5.1 - Manual Figure/caption indexing on BIUM web base

Caption lines are extracted by merging results issued from two different systems. In system 1, the pages are preprocessed to segment graphics and extract the horizontal and vertical text lines. Text lines are sorted using spatial criteria in order to select a set of caption line candidates. In system 2, the occurrences of the *caption labels* of a book (such as 'Fig.', 'Figure') are searched by using word spotting in the candidate caption lines. The text lines explored by word spotting and the word-similarity threshold value are updated in the course of process according to the results of the word search.

5.1.1 Selection of figure caption candidates

The pages in the historical books of Medic@ contain vertical and horizontal text lines and the caption lines are not always in the direction of the main text. The proposed method detects vertical and horizontal text lines without prior assumption on their direction. It is explained in detail in [Faure and Vincent2009].

A size criterion, different from the one used in chapter 3, is used to interpret the large CCs as Graphics. They are labelled CCG and are discarded from the grouping process leading to text lines. The remaining CCs are grouped according to the main properties that enable a human reader to detect symbols alignments (proximity, similarity, direction continuity). Each CC is labelled NNH if its nearest neighbour is found in the horizontal direction or NNV if it is found in the vertical one. The labelled CCs are the input of a rule-based incremental grouping process. The sequences of consecutive NNH or NNV define horizontal and vertical alignments, grouping CCs according to proximity and continuity of direction (Figure 5.2).

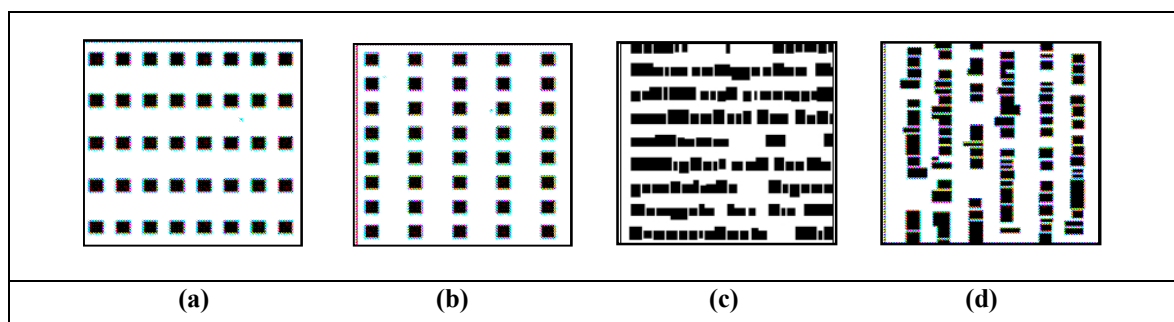


Figure 5.2 – a, b) The nearest neighbors of each black square belong to the perceived a) row or b) column. c) Filled in black, the NNH bounding boxes in horizontal text lines, d) and the NNV in vertical text lines

These first alignments are expanded in the later steps of the grouping process. An alignment is expanded along its main direction by merging it with its nearest neighbour alignment or by adding the nearest neighbour CC found in the main alignment direction. Typographic conditions are defined to take into account similarity and continuity properties: expanding an alignment by adding a CC or merging two alignments is allowed if the height of the resulting alignment is smaller than 1.5 times its height before being expanded and if the distance between the alignment and the CC or between the two alignments is smaller than twice the height of the alignment to be expanded.

This stepwise method takes advantage of emerging organisation and is easy to control. Therefore, the spatial information involved in the grouping rules is not reduced to the local

information between CCs. After each step of the grouping process, previously detected alignments are reinforced or eliminated. Conflict detection is activated after each grouping step. The main conflict is detected when a CC belongs both to a vertical and a horizontal text line (see Figure 5.3). A voting rule solves this conflict: the vertical (horizontal) line is eliminated if it contains a number of CCs smaller than the number of CCs in the horizontal (vertical) line intersecting it. To be consistent with layout conventions, a text line cannot straddle the borders of the CCG bounding boxes. Therefore, text lines can be detected outside or inside a CCG.

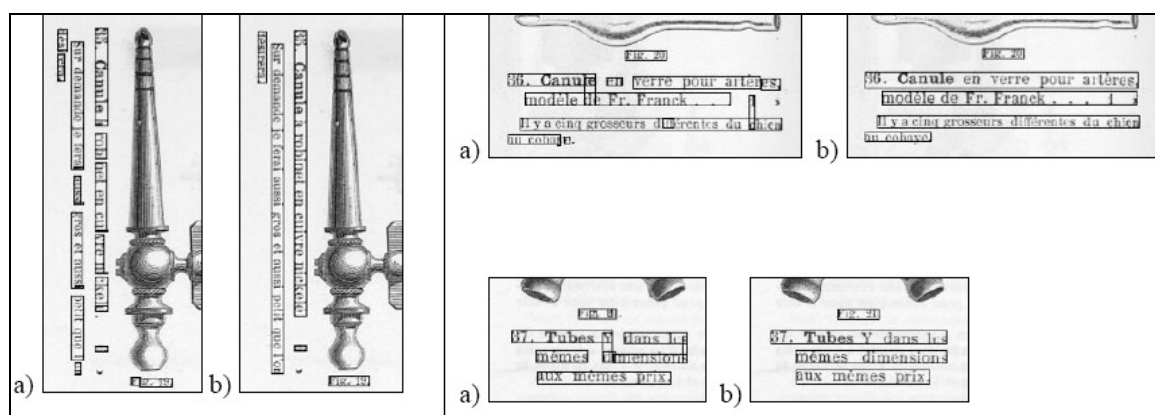


Figure 5.3 - Close views of a page: a) After grouping NNV and NNH. b) Final detection

Spatial criteria are defined to sort the detected text lines in order to select caption line candidates. Actually, these criteria are aimed at selecting a line as caption candidate which is closest to the figure. For each side of a CCG, the nearest text line is found, meaning at most four candidate lines per figure. The confidence of these text lines is increased by one if it is the closest line to the CCG or if the centre of the CCG and the centre of the text line are aligned along a vertical or horizontal direction. Text lines with a positive confidence belong to the set of caption line candidates (example in Figure 5.5a). With the current version of the system, text lines included in the CCG bounding box cannot be among this CCG caption line candidates.

5.1.2 Spatio-symbolic information fusion

The most important phase of this study is the fusion of *symbolic information* obtained by word spotting and *spatial information* used to select the caption line candidates. Once the bounding boxes of the caption candidates are obtained, word spotting is applied on them to *increase* confidence of some candidates, to *eliminate* major false positive candidates and to *find* caption lines missing in the candidate set. So, the input of the word spotting process will be a query word (caption label) and the location & direction of the candidate lines.

The caption line candidates constitute the first set of text lines that are explored by word spotting. For each book, a caption label, which can be a word such as ‘Fig’, ‘Figure’ or ‘FIGURE’, is chosen as the query word. The vertical caption candidates are rotated to become horizontal in order to apply character segmentation, feature extraction and word matching as previously defined for horizontal lines.

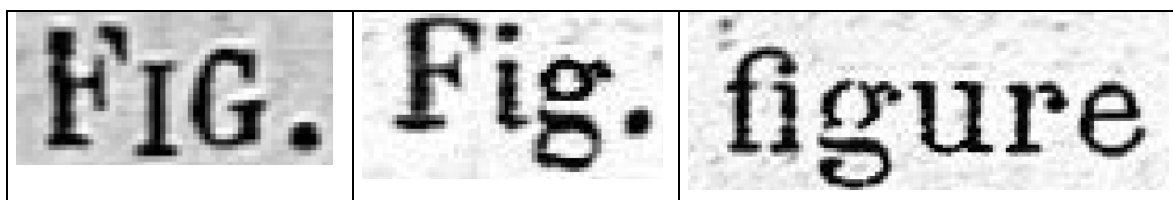


Figure 5.4 - Different Caption Labels used in Medic@ books

If no occurrence of the query word is found in the candidate lines associated with a figure, the word similarity threshold is increased. Increasing the tolerance in word matching is acceptable in this context as the search is limited to words in the caption candidate bounding boxes. It enables to detect missing occurrences of the query word without increasing false positive detection. Once a caption label is found in a caption candidate associated with a figure, its confidence is increased and the other candidate lines are neglected. Figure 5.5b) gives an example where out of three line candidates, the word ‘Fig’ is detected only in one of them. Thus this line is accepted as the caption line and the other candidates for that figure are neglected.

If the query word is not found in any caption candidate associated with a figure, this is interpreted as a missing caption line in the candidate set. Then, the query word is searched in all the lines of the page. If the query word is found in the spatial neighborhood of the figure, then the relative position of that line and the actual figure is examined to see whether or not this line may represent a caption. Figure 5.5 c,d) shows an example where the first line of the caption was not detected in the caption line candidates (this line is included within the bounding box of the figure itself). Word spotting of the word ‘Fig’ in this line permits to detect the caption line missing in the candidate set.

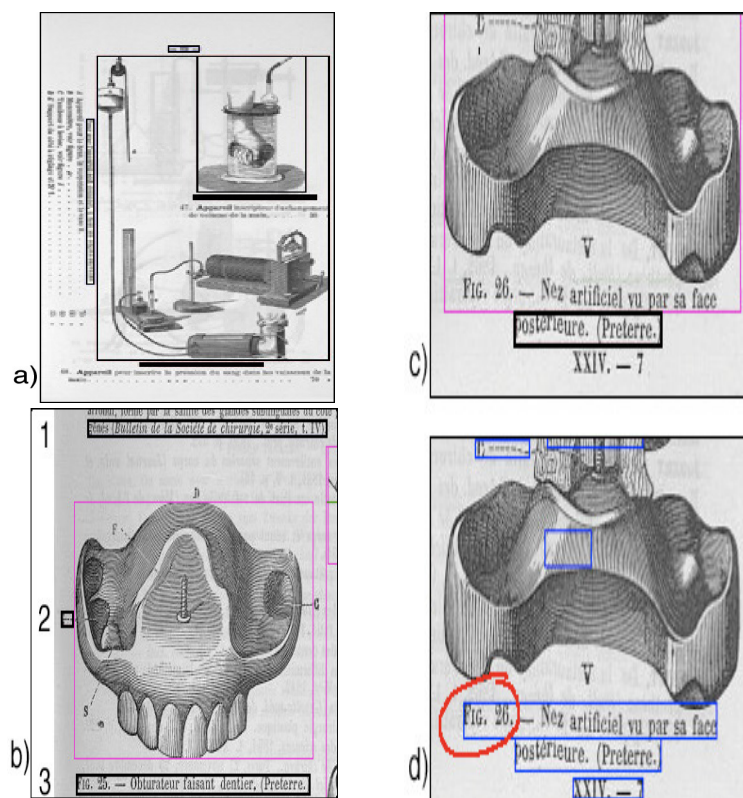


Figure 5.5 - a) Figure and caption candidates, true caption lines are filled in black. b) Three line candidates, after word spotting of “Fig” candidates 1 and 2 are eliminated. c - d) Candidates do not include the first caption line retrieved by word spotting

The results of information fusion for the detection of caption lines are analyzed to assess the number of confirmed as well as suspected caption candidates.

5.1.3 Caption Retrieval results

The fusion of text retrieval and spatial information reasoning has been tested on our data set C having three historical books printed in the 19th century. They are available in the digital library Medic@. A total of 210 figure captions exist in these books. Out of 210, 204 captions contain a caption label: the word ‘Fig’, while the other six do not contain any caption label. Spatial information leads to 449 caption candidates. This number is well above the actual caption number. Out of the ground truth 210 caption lines, 192 text lines have been detected perfectly and selected as caption line candidates, while 18 were either not detected perfectly or were not selected at all in the candidates.

Now to evaluate these caption candidates, word spotting provides information to distinguish between true and false positives. By applying our word spotting algorithm for query word ‘Fig’ in the candidate lines using a fixed threshold; we were able to detect 160 occurrences of ‘Fig’, thus

confirming 36% candidates. Increasing the threshold enabled to confirm 9 new candidates. By applying word spotting for 'Fig' in all text lines of the document images, 11 caption lines missing in the candidate set are detected. The results are summarized in table below.

Table 5.1 - Results for figure caption retrieval

Caption line candidates	
# Figure captions in ground truth	210
# Total caption candidates using spacial criteria	449
# Well detected captions in the caption candidates (Recall)	192 (91%)
# False positive caption candidates (Precision)	257 (42%)
Word spotting	
# Captions in ground truth with a label ('Fig')	204
# Caption candidates confirmed by word spotting	160
# Caption candidates confirmed by increasing word similarity threshold	9
# Captions retrieved by word spotting of query word in all text lines	11
# Total captions detected using word spotting (Recall)	180 (88%)
# False positives during word spotting in candidates (Precision)	4 (98%)

Results show the efficiency of our system to confirm or eliminate caption line candidates. The use of word spotting leads to associate 180 figures (out of 210) with a single caption line containing the word "Fig.". The six figures for which the caption does not have a label ("Fig." or other) are not in this set. A caption candidate may be associated with one or several figures and a figure may have one or several caption candidates. Once a label is recognized in a caption candidate, the other candidates (if any) associated with the same figure are eliminated.

One advantage of using word spotting is that a large number of figure and caption pairs are directly detected based on spatial and visual properties without further analysis. Another is the possibility to detect caption lines which are not in the candidate set. The errors resulting from false positive word spotting recognition do not have a strong contribution in the results.

5.1.4 Conclusion

We have shown an application of our word retrieval system where it is used for the detection of figure and caption pairs by taking advantage of the symbolic information that is often encountered in captions to label them (the word "Fig." in our data set) and to avoid a complex decision process based on visual and spatial information. Most of the figures were directly associated with a single caption by word spotting, thus further processing is activated only for a small number of figure and caption pairs, among them the figures with a non-labeled caption.

This application of our system can bring some help to speed up and semi-automatize the indexing of books and their presentation on the web. The expected final result could be the detection of figures associated with caption blocs and also the recognition of the figure numbers following the caption labels (if any). Establishing a link between figure numbers occurring in the text and the associated caption is another goal for the combination of word spotting and spatial analysis of document images.

5.2 Application on contemporary documents

After analysis our word retrieval system for historical documents, we also analyzed the contemporary document images of 20th century to see what sort of challenges they pose for a retrieval system and what sort of results our system achieves on these documents. Plenty of modern day documents are available on the internet in a low resolution in scanned images form. There are not many degraded quality issues with them as are with ancient documents but they do have some challenges of their own. One example is the ink dilation issue caused by low resolution and lossy image file storage formats which affect the text in a way that characters appear a bit dilated and are merged together in some cases as well (Figure 5.6).

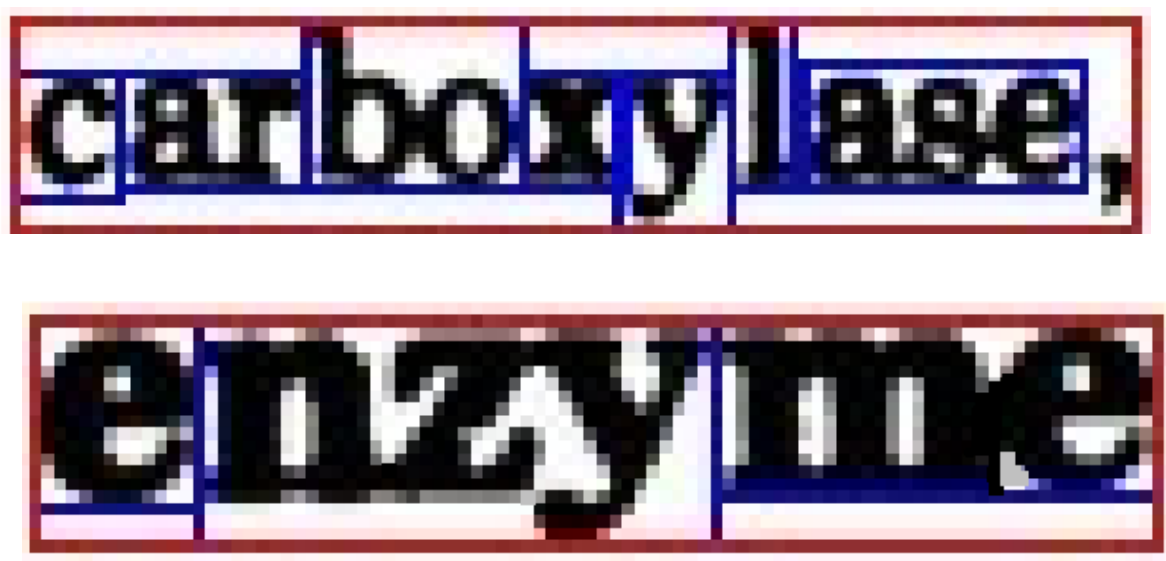


Figure 5.6 - Character segmentation issues in contemporary document images

We performed an experiment for testing our multi-stage retrieval system on different document images of bio medical research papers from the 70's decade. A total of eleven document pages containing a total of 6500 words are used for testing the retrieval process. Sample document images are given in Annex A. The prototype characters for queries are made manually using

Times New Roman font of size 12. This is because we wanted to generalize the prototypes (by not taking the prototypes from the document images themselves) so that they can be used with all types of contemporary document images. The prototype characters are made only in normal font and no prototypes are made in italic and bold fonts. This will be a good way to study how our system performs with just normal font prototypes to retrieve the word instances printed in either bold or italic or both bold-italic fonts. For query, 18 different words having 423 instances in total were selected by the experts of medical field according to relevancy to the subject. Results of the tests for an optimum average threshold value are given in Table 5.2.

Table 5.2 - Summary of word retrieval results for different query words

ASCII Query	Total Instances	Precision	Recall
citrate	15	100.00	93.75
isocitrate	16	75.00	75.00
succinyl	6	100.00	100.00
succinate	40	92.11	87.50
fumarate	19	82.76	80.00
malate	51	95.00	74.51
oxalacetate	19	100.00	73.68
acetyl	3	100.00	100.00
pyruvate	11	100.00	54.55
phosphoenolpyruvate	1	100.00	100.00
aconitase	12	85.71	100.00
synthetase	1	100.00	100.00
fumarase	27	100.00	81.08
malic	37	93.33	73.68
enzyme	67	100.00	89.55
carboxylase	2	100.00	50.00
carboxykinase	1	100.00	100.00
dehydrogenase	95	100.00	95.79

The system tested at different threshold values achieves overall recall rates in the range of 78 to 89% while maintaining a precision of 96 to 100%. When we examined the relatively low recall rate, different justifications came up for that. One reason is that there are instances of the query words that are broken into two words in two consecutive lines. Examples are given in Figure 5.7. Currently our system doesn't have this capability to extract this sort of broken words. In the future work though, it would be beneficial to have this capability added to the system.

acetylating]), aconitase (EC 4.2.1.3, citrate [isocitrate] hydro-lyase), and isocitrate dehydrogenase (EC 1.1.1.42, *threo-D₅*-isocitrate: nicotinamide adenine dinucleotide phosphate oxidoreductase [decarboxylating]). These enzymes are responsible for the synthesis of α -ketoglutarate, and they have been shown to be subject to feedback repression by α -ketoglutarate or glutamate (12, 14, 22, 30). In addition,

Figure 5.7 - Examples of words broken down in two lines

Another important thing to note is that we are able to detect most words written in **bold** or *italic* styles. A few of them though were not detected using the normal font style prototypes. So to overcome that, prototypes of bold and italic can also be made which will slightly improve the recognition rate.

5.3 Application on Cursive oriental scripts – Arabic/ Urdu

A lot of work has been going on in the field for the recognition of cursive scripts such as Arabic. We also experimented on some Arabic document images using our word spotting system to see if it can be adapted to work with other languages as well. For that, we worked on a small dataset of around 200 words taken from a couple of pages of the Quran, available on the web¹⁴. This text provided an extra challenge for our system as these pages are not post-processed after scanning and so the skew and slant is there as well. Sample page is shown in Figure 5.8. We didn't correct the skew/slant and worked on the text as it is, to see if our system is capable to spot the searched words in the original available pages on the web.

¹⁴ www.kitaabun.com/shopping3/images/



Figure 5.8 - Sample page of the Quran with the text lines slanted

The images are first converted to gray scale and then the words are processed to get the S-characters. S-characters are not fixed using our 3 pass processing stage because we want the diacretic marks (which are very important in Arabic) to remain as separate S-characters. Matching of the words was performed using the Merge-Split Edit distance coupled with DTW for S-characters. Here note that the S-characters do not represent T-characters in most cases as an S-character is usually a combination of multiple arabic alphabets (See Figure 5.9).

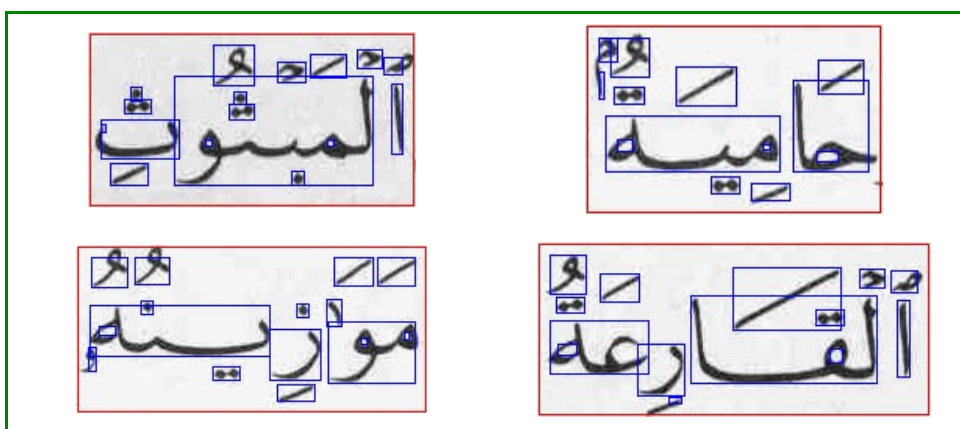


Figure 5.9 - Examples of S-characters for different words

All S-characters in a word are sorted on the basis of the start of the left side of their bounding boxes in x-direction, giving one dimensional sequence of S-characters as shown in Figure 5.10.



Figure 5.10 - Sequence of S-characters in a word

For testing, we selected five most frequently occurring words as query and for each found the top 6 matches using the Merge-Split Edit distance method. Results are shown in Table 5.3.

Table 5.3 - Results of word spotting on Arabic text – Top 6 matches along with their distances (the distance values of **false positives** and **extra** words are highlighted)

Query	سُورَةٌ	بِسْمِ	الْقَارِعَةُ	يَوْمِئِذٍ	مَنْ
Instances	5	4	4	4	3
Rank 1	سُورَةٌ	بِسْمِ	الْقَارِعَةُ	يَوْمِئِذٍ	مَنْ
distance	0.000	0.000	0.000	0.000	0.000
Rank 2	سُورَةٌ	بِسْمِ	الْقَارِعَةُ	يَوْمِئِذٍ	مَنْ
distance	0.270	0.198	0.200	0.185	0.382
Rank 3	سُورَةٌ	بِسْمِ	الْقَارِعَةُ	يَوْمِئِذٍ	مَنْ
distance	0.388	0.270	0.215	0.232	0.386
Rank 4	يَوْمِئِذٍ	بِسْمِ	يَوْمِئِذٍ	تُحَدِّثُ	بِسْمِ
distance	0.434	0.294	0.431	0.316	0.386
Rank 5	سُورَةٌ	مَنْ	مَوَازِينُهُ	عَيْشَةٍ	لِمَنْ
distance	0.491	0.368	0.476	0.328	0.483
Rank 6	سُورَةٌ	مَنْ	الْقَطْرِعَتَا	يَوْمِئِذٍ	مَنْ
distance	0.518	0.390	0.491	0.389	0.537

The results show that the adaptability power of our system which allows it to match words written in non Latin text as well. We believe that performance of the system on Oriental cursive scripts can be improved further by adding such representative features for the S-characters that could take into account the curves and circular nature of the writing style in a better way than the proposed Latin text features. The system could then be evaluated on a large set of document images to analyse the different performance measures.

Chapter 6

Conclusion and Perspectives

The importance of digital libraries for information retrieval cannot be denied. The ancient historical books contain invaluable knowledge but it is time consuming for the researchers to search the required information in these paper books. Our work in this domain aims to facilitate this information search by spotting exact query word instances in the text. With this ability to search in ancient historical documents, digital libraries will further enhance their importance.

6.1.1 Summary of the method

The overall retrieval system has been divided into two major parts: the document indexing part and the matching part for retrieval of relevant document images. For *indexing*, the document image is first binarized. A new document thresholding algorithm is defined to crisply distinguish the foreground and the background [Khurshid *et al.* 2009c]. Then, a *word/graphic segmentation* is performed using a combination of horizontal smoothing and connected component area analysis. The next step is aimed at segmenting words into characters. The extracted words are composed of a set of connected components for which several rules are defined to combine them in order to better fit character segmentation. Nevertheless, the quality of the images does not permit a character segmentation free of errors. We call the final word components as S-characters to differentiate them from the true characters. *Multidimensional features* are defined to obtain an efficient representation of the shape of an S-character. They are stored, along with some other relevant information, in the index file created for each document image.

For *word spotting*, we have proposed several word-matching methods. The main aim is to use query word to retrieve relevant document images, by searching the word images similar to the query word. Major problem was to define a word matching method which not sensitive to character segmentation errors caused by the low quality of the document images. Query word can be given either in the form of a word image or as ASCII, in this later case the word image is built automatically from characters prototypes. The query word is processed and is compared with other words in the data base through a multi-step retrieval system, using the proposed word representation as a sequence of S-characters. In the first step of the matching process, we narrow down the data set to be searched by introducing a *length-ratio filter*. It finds all eligible word candidates that could be similar to the query word. In the second step, the query word and

candidate words are matched using a dynamic retrieval system where two words are matched using a string comparison algorithm in which two S-characters are matched by comparing their features using elastic DTW method. Several string comparison algorithms were proposed and evaluated. The best performances were achieved with the Merge-Split Edit distance that takes into account the characters segmentation inconsistencies for correctly matching the two words. For accepting that two words are identical, their matching distance must be below a pre-defined threshold. All words having a matching distance less than the threshold are the retrieved words.

This work provides a thorough examination of several retrieval techniques for historical document images that allow queries in the form of a word image or ASCII text, a fact that makes this system very practical. Another important thing is that though it is a segmentation-based method, results are **not** dependant on perfect character segmentation as our Merge-Split Edit distance caters for the segmentation inconsistencies. This fact makes the system very appealing, especially for historical documents where it is not possible to have perfect character segmentation due to poor image quality. Also, feature definition at S-character level gives a better representation of words as compared to word level features [Khurshid *et al.* 2008a], and the matching of S-character features using DTW renders the system scale and translation invariant. Overall, the recognition results achieved by the system are very promising and the system has the potential to be used for different applications related to historical documents.

6.1.2 Future work – perspectives

Our prototype system is an implementation of a concept for historical document image retrieval. Because of the size of this system and different options that we give (like query in image or in ASCII, etc.), which did not allow us to work out all components to the smallest detail, and the numerous challenges the problem of historical document retrieval poses, plus the time constraints; various limitations still remain in the system that need improvement. Here we propose directions for future work, paying special attention to making our system practicable in all conditions for different types of historical document images.

Building an entire retrieval system is a big effort that requires making numerous decisions at different individual stages. Due to time and resource constraints, some of our choices in this work do not have an in-depth investigation of what the optimal alternative would be. For example, the particular choice of feature sequences to represent a character image. Features like the upper/lower profiles, projection and ink/non-ink transitions for a column image are already reported in the literature as in [Rath and Manmatha2007], [Kolcz *et al.* 2000], [Konidakis *et al.* 2008], etc., and we believe that performance may increase with the addition of new descriptors and features.

Our investigation has focused on features at S-character image level that represent the overall shape of the S-character, but it may be beneficial to use features that take into account local characteristics when making classification decisions.

Some of the system's shortcomings lie in the initial document processing. Most of our work has concentrated on 18th and 19th century document images which have more regularity as compared to the older documents of say 15th and 16th century. In order to handle a greater variety of layouts in a better way, the segmentation may be improved to take into account the possibility of presence of very small textual images or drawings within the text. Various document image distortions, such as rotated/skewed pages and other distortions that occur when old books are scanned with the binding still in place which causes individual text lines to bend and curl at the end, might also be taken into account .

The clustering for the S-character images may be improved to get the prototype character images automatically. This will be a huge plus if we are able to have prototype characters automatically for each book.

In the figure captions application part, we can detect figure numbers as well using a hypothesis that a figure number follows the caption label. We can have an automatic recognition of figure numbers using number logic system. We can also have a mechanism to automatically link the reference of a figure in the text with the figure itself using figure numbers.

A shortcoming of the system that we observed while testing the contemporary document images was the inability to retrieve words that are broken into two parts (one at the end of a line and other at the beginning of the new line). This capability can be added to the system using text lines and words position coordinates.

We have mainly focused here on the historical books of the late 18th and early 19th century. The techniques presented in this work may be validated and refined to allow successful retrieval of varied set of document images belonging to different era. Preliminary experiments though with older 15th century books images are promising.

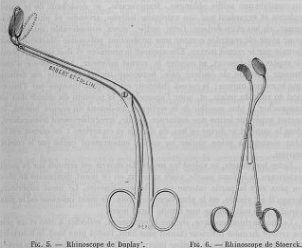
Digital libraries are a relatively recent phenomenon and require new approaches to document images retrieval. This work contributes to this field by implementing different techniques of information searching and word retrieval as well as automatic figure caption indexing for ancient

historical documents, which form a substantial portion of library collections around the world. While the present retrieval system can still be improved, we hope to have convinced the reader that this work represents a significant step towards extending retrieval capabilities to historical document images.


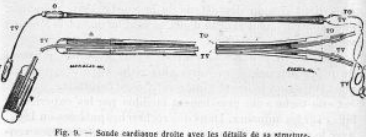
Appendix A

Sample Images

A.1 BIUM Sample images of the 12 books of 18th and early 19th century

<p style="text-align: center;">— 35 —</p> <p>se charger de vapeurs. Il faut recommander aux malades de faire très lentement les premières inhalations, de ne pas pousser la menthol à une température trop élevée, parce qu'ils sont pour les premières fois pris de toux et d'une suffocation aigüe.</p> <p>Ces petits accidents ne se renouvellent pas dans les séances suivantes. On peut ainsi faire deux ou trois séances dans la journée, une dizaine d'inspirations par séance, en employant une température d'autant plus élevée que l'aflection est plus intense ou plus rebelle (1).</p> <p>Du canal de Jacobson, de la possibilité de le reconnaître sur le vivant et de son rôle probable dans la pathogénie de certaines lésions de la cloison nasale.</p> <p style="text-align: right;">par le Docteur Potinger.</p> <p>Le canal du Jacobson représente chez l'homme un des vestiges de l'organe du même nom. L'organe de Jacobson, destiné à l'olfaction, atreint son plus grand développement chez certains mammifères. Il consiste sur le mouton, par exemple, en un tube membrano-muqueux renfermant quelques ramifications du nerf olfactif; ce tube est lui-même inclus dans un étui cartilagineux appliqué de chaque côté de la cloison des fosses nasales. Chez l'homme, l'organe n'est plus que rudimentaire : l'étui cartilagineux se réduit à de minces languettes ou à de petites baguettes de cartilage (cartilages de Jacobson, cartilages accessoires de M. Sappey) qui doublent de chaque côté la base de la lame quadrilatère et la pointe du vomer, et</p> <p>(1) Sur nos indications, MM. Herblin et M. Bardi, pharmaciens, ont construit l'un et l'autre, sur le principe sus-indiqué, de petits appareils plus élégants et plus commodes.</p> <p>(2) Ces cartilages jouent dans les épissémentes de la portion inférieure de la cloison un rôle important indiqué par M. Sissman (Congrès international de Berlin, 1890), rôle que nous essayons de décrire dans une prochaine étude.</p>	<p style="text-align: center;">— 25 —</p> <p>pagner de désordres locaux ou généraux.</p> <p>Les désordres locaux sont: du prurit, de l'agacement des gencives, parfois de la gingivite, plus rarement de la stomatite, érythémateuse ou aphtheuse, et toujours, dans ces divers cas, une salivation exagérée.</p> <p>On combat le prurit et les douleurs de la gingivite par des applications du collutoire suivant, faites, ou avec un pinceau en blaireau, ou avec un lingé:</p> <table border="0" style="width: 100%;"> <tr> <td>Chlorhydrate de coccaïne.....</td> <td>25 centigrammes.</td> </tr> <tr> <td>Eau pour dissoudre.....</td> <td>Q. S.</td> </tr> <tr> <td>Glycérine.....</td> <td>20 grammes.</td> </tr> <tr> <td>Succharine.....</td> <td>5 centigrammes.</td> </tr> <tr> <td>Essence de menthe.....</td> <td>1 goutte.</td> </tr> </table> <p>Quelquefois, une incision de la gencive, surtout quand la sortie de la dent paraît imminente, produit un soulagement manifeste en décongestionnant et en détendant la muqueuse.</p> <p>Si les accidents locaux sont assez généralement admis, il n'en est plus de même des accidents réflexes et généraux; car on peut à leur sujet enregistrer les opinions les plus contradictoires.</p> <p>D'après les uns, d'accord en cela avec la tradition (<i>Bel enfant jusqu'aux dents</i>, disaient nos pères), la dentition serait une épreuve des plus</p> <p style="text-align: center;">3</p>	Chlorhydrate de coccaïne.....	25 centigrammes.	Eau pour dissoudre.....	Q. S.	Glycérine.....	20 grammes.	Succharine.....	5 centigrammes.	Essence de menthe.....	1 goutte.	<p style="text-align: center;">NÉVRITE SEGMENTAIRE PÉRI-AXILE. 181</p> <p>Il ne sera pas inutile, croyons-nous, de revenir un instant sur les caractères anatomiques de cette névrite et de montrer, autant du moins qu'on peut en juger par ce genre de preuves, qu'il s'agit bien là d'un processus irritatif, d'une véritable inflammation parenchymateuse. Cet examen nous servira en même temps à faire ressortir les différences qui ont été signalées déjà entre ce processus inflammatoire et celui de la dégénération wallérienne.</p> <p>Du côté du protoplasma contenu dans l'intérieur de la gaine de Schwann, nous mentionnerons tout d'abord un bourgeonnement infiniment plus actif que dans la dégénération wallérienne, accompagné d'une prolifération nucléaire dont cette dégénération ne présente pas d'exemple. On peut compter (Pl. I, fig. 5) jusqu'à dix et quinze noyaux contenus dans une même masse cohérente de protoplasma, tandis qu'on rencontre six ou huit masses semblables sur la longueur d'un même segment interannulaire.</p> <p>Les lésions de la myéline nous semblent plaider dans le même sens. Finement émulsionnée dans un cas, segmentée en blocs volumineux dans l'autre (Rauvier), ce qui constitue une différence importante, la gaine de myéline peut présenter au début de la névrite péri-axile une particularité sur laquelle nous voulons appeler de nouveau l'attention, parce que nous voyons une nouvelle preuve de la nature irritative du processus dans ce cas. Gonflée et rendue plus claire dans ses couches extérieures, la gaine a subi une modification très comparable à celle qu'elle présente après l'imbibition du nerf par l'eau salée dans l'expérience de M. Rauvier. Mais, comme ici, le liquide qui</p>
Chlorhydrate de coccaïne.....	25 centigrammes.											
Eau pour dissoudre.....	Q. S.											
Glycérine.....	20 grammes.											
Succharine.....	5 centigrammes.											
Essence de menthe.....	1 goutte.											
<p style="text-align: center;">130125x1891x04</p>	<p style="text-align: center;">APHPF00384</p>	<p style="text-align: center;">EPO0228</p>										
<p style="text-align: center;">NEZ, FOSSES NASALES. — RHINOSCOPIE. 27</p> <p>confier l'abaisse-langue. Désireux de permettre au chirurgien de s'acquiescer sur des divers manœuvres opératoires, Czermak et après lui nombre de praticiens ont conseillé de supprimer l'abaisse-langue, en se servant de miroirs d'inspection à lige très-grosse faisant office de instrument. L'auteur (Siszek, Dupuy) ont imaginé de combiner le miroir et le releveur de la lacinie. L'instrument de Dupuy mérite la préférence.</p>  <p style="text-align: center;">Fig. 5. — Rhinoscopie de Dupuy. Fig. 6. — Rhinoscopie de Siszek.</p> <p>Il se compose de deux longues branches croisées à l'une, fixe, se termine par un miroir, dont l'orientation peut être réglée à volonté; l'autre, agissant d'une double levée sur la première, fait mouvoir un anneau, placé en avant du miroir et destiné à relever la lacinie et le releveur du palais (fig. 5).</p> <p>L'instrument de Siszek est représenté fig. 6. Ces instruments compliqués sont cependant inutilés d'ordinaire, car le malade cherche à arracher facilement à main levée lui-même sa langue déprimée.</p> <p>L'exigence du miroir rhinoscopique s'oppose à ce qu'on puisse obtenir d'un seul coup d'œil une vue d'ensemble satisfaisante. C'est donc en vain que, pour arriver à ce résultat, on a conseillé de placer le miroir au-dessous de la lacinie, c'est-à-dire au milieu de l'isthme du gosier. Le mieux est de déplacer l'instrument de droite à gauche et de gauche à droite, en ayant soin de ne pas le glisser sur la base de la langue qui est très-sensible et qu'il faut retirer le moins possible. On obtient ainsi des images partielles, dont la réunion permet de se faire une idée exacte de l'ensemble des fosses nasales.</p>	<p style="text-align: center;">INGUINAL. — HERNIE INGUINALE. 53</p> <p>La théorie pathologique trouve une application bien plus générale que la précédente, et s'appuie sur des faits anatomiques qui nous semblent indiscutables. Il est évident que dans les hernies anciennes et volumineuses il y a un allongement des méésentères et de l'épiploon, et qu'à l'état normal l'intestin grille arrive à peine assez loin pour s'engager par son bord libre, dans un sac herniaire. De plus, la forme si bien décrite par Kingsley, du ventre-herniaire rentrant à l'épigastre et saillant dans les régions inguinales, indique que l'intestin y pend plus bas qu'à l'état normal. Le test, suivant nous, de la théorie pathologique, c'est d'enchaîner à peu près complètement de la pathogénie des hernies le rôle des contractions abdominales, et de réduire à un minimum l'influence des apertures inguinales et de leurs anneaux fibreux.</p> <p>Quant à la théorie mécanique, elle insiste trop sur la cause déterminante des hernies, sur l'effort, aux dépens du travail pathologique qui les prépare et qu'elle méconnaît presque complètement.</p> <p>Enfin d'abord les hernies secondaires ou acquises. Quel est le premier effet de cette rupture d'équilibre, si bien signalé par Scarpa, entre la pression intra-abdominale et la résistance des parois de l'abdomen? Est-ce la production brusque d'une hernie? Tous les observateurs modernes s'accordent pour en nier la possibilité. Les parties musculaires de la paroi abdominale cèdent sous l'influence de la pression intra-abdominale; mais, grâce à leur nature, elles reprennent vite leur équilibre, ou bien elles se contractent et résistent, ou, au contraire, pour leur part à augmenter cette pression. Les parties aponeurotiques, la région inguinale notamment, si elles ne sont pas assez fortes pour résister à l'augmentation de pression qu'elles subissent, se laissent distendre, et, n'ayant que très-peu d'élasticité, elles ne peuvent revenir à leur état primitif et se déforment peu à peu. Les régions inguinales deviennent saillantes, la paroi abdominale à leur niveau est projetée en avant. Cette déformation, chez les sujets jeunes et musclés, sera marquée en partie dans la sciatique par la lésion du grand oblique tendant son aponeurose; elle devient marquée lorsqu'on fait courber le tronc en avant, et dans le décubitus latéral. On voit alors, pendant la toux et pendant l'effort, la région inguinale se soulever brusquement, tandis que toutes les parties situées au-dessous se laissent et s'aplatissent. Appelé pendant plusieurs années à examiner des jeunes gens au point de vue du recrutement militaire, j'ai pu me convaincre que cette disposition que j'indique est assez commune.</p> <p>Il n'y a pas encore de hernie, mais les intestins manquent de soutien en bas et en avant, descendant plus bas qu'à l'état normal; petit à petit leur méésentère s'allonge; il remplit la partie inférieure de l'abdomen dans la sciatique, et le ventre grand côté forme si bien décrite par Kingsley il est aplati au-dessous de l'ombilic et saillant au-dessous. Alors aussi, à chaque effort, l'abaissement intestinal et l'allongement du méésentère à de la lacinie à s'engager. En effet, grâce à la saillie de la partie inférieure de l'abdomen, les contractions du transverse agissent au-dessous du piquet intestinal abaisse. Loins de tendre à le remonter vers la colonne</p> <p style="text-align: center;">REV. 1902, 102, 17 (11).</p>	<p style="text-align: center;">KLIPPEL. — PSEUDO-PARALYSIE GÉNÉRALE ARTHRITIQUE 283</p> <p>L'arthritisme n'était pas seulement visible au niveau des artères cérébrales, on en trouvait des marques évidentes dans d'autres organes.</p> <p>Il existait une sortie chronique avec dilatation et dont la paroi interne était couverte de foyers, les uns scléreux, les autres en état de ramollissement. Le cœur gauche était hypertrophié et scléroté.</p> <p>L'un des reins présentait de la néphrite interstitielle évidemment d'origine vasculaire. A une semblable maladie, offrant la symptomatologie de la paralysie générale, l'âge du malade mis à part, mais dont les lésions sont absolument différentes, le terme de pseudo-paralysie générale nous paraît devoir convenir.</p> <p>OBSERVATION. — Pseudo-paralysie générale arthritique. — Tableau clinique de la paralysie générale. Arthrite-arthrose cardio-articulaire et vésicale. Atrophie des artères cérébrales. — Dégénérescence graisseuse des capillaires de l'écorce cérébrale et des cellules nerveuses. — Pas de lésion caractéristique de la paralysie générale vraie.</p> <p>Le nommé Dan..., âgé de soixante ans, journaliste, entra le 3 avril 1891, à Sainte-Anne, clinique de M. le professeur Bail.</p> <p>Le malade est entré avec un certificat attestant l'établissement des facultés intellectuelles, des idées délirantes de persécution, de grandeur, de mégalomanie et de manigances cliniques, des projets déraisonnés, des actes incohérents et des troubles de la parole.</p> <p>Le certificat médical le déclare atteint de paralysie générale à forme évasive, de délire des grandeurs, d'embarras de la parole, du tremblement de langue et d'ingémité papillaires.</p> <p>A ses faits déjà bien caractéristiques, et présentant le tableau ordinaire de la paralysie générale, nous ajouterons les particularités suivantes, d'après l'observation de M. Baïeux, interne distingué des hospices :</p> <p>D'après les renseignements puisés auprès d'un parent du malade, le début de la maladie remonterait à environ trois ans. Il y eut d'abord des signes d'établissement aussi bien physiques qu'intellectuels. Le malade vivait chez lui dans une malpropreté révoltante depuis deux ans. De plus, par moments, il devenait violent. Il avait toujours été très soûlé. Huit jours auparavant entré à Sainte-Anne, ces crises cessèrent pour ne plus reparaître. Amnésie, rapidement progressive, qui deux ou trois mois après son entrée était complète.</p> <p>Dans les trois premiers semaines, il présente par contre quelques idées de grandeur assez vagues et qui ne sont régulièrement dissipées; il se pécifiant très riche, avait épousé avec lui son habit de cardinal, etc. Il passait la plupart de ses journées assis sur un banc, la tête affaissée sur la poitrine, apathique, inerte; parfois il se promenait ou plutôt se traînait péniblement dans la cour; il se plaisait alors à ramasser des objets divers, des fragments de journaux et surtout des morceaux de pain qu'il avalait glotonnement. C'était pendant la journée, il dormait toute la nuit, ne parlait jamais aux autres malades.</p>										
<p style="text-align: center;">32923x24</p>	<p style="text-align: center;">32923x19</p>	<p style="text-align: center;">EPO0561</p>										

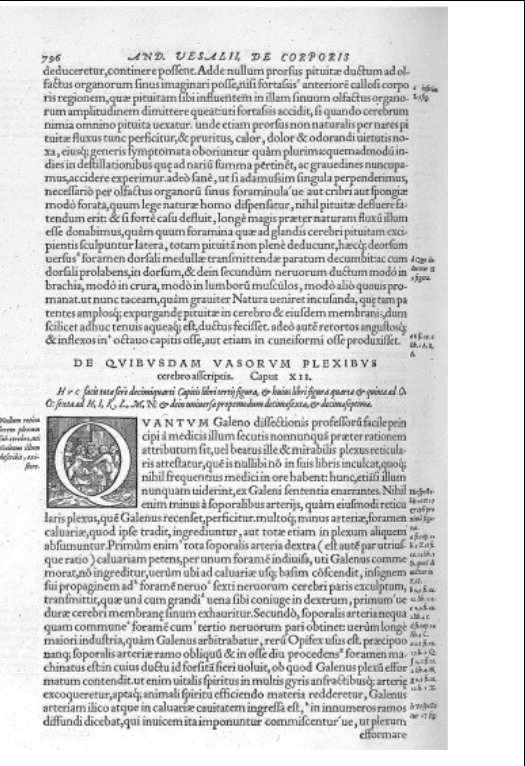
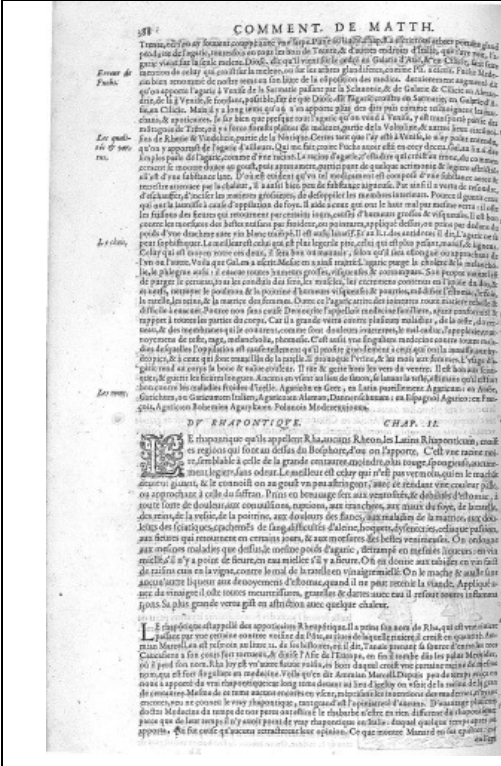
A - Sample Images

<p style="text-align: center;">— 11 —</p> <p>mention de la possibilité de lésions pancréatiques dans la glycosurie. Plus tard, en 1864, Ancelet fait une étude des plus consciencieuses sur les maladies du pancréas. Il résume un nombre considérable de faits, surtout au point de vue anatomique et, dans les cas très-rares où il s'occupe de la symptomatologie, le diabète n'est pas en cause. La même année Besson examine quelques faits pour servir à l'étude du pancréas et il est également mét sur les symptômes diabétiques.</p> <p>Recklinghausen, en 1864, rapporte deux cas d'ectasie saciforme du conduit de Wirsung, avec atrophie du pancréas, chez des diabétiques.</p> <p>Da Costa publie ensuite un mémoire sur le cancer du pancréas; il n'insiste pas sur la possibilité de troubles diabétiques. En dépit de ses observations, on constate qu'un de ses malades était atteint de diabète, et six autres eurent une soif insatiable.</p> <p>La thèse de Maigre (1866, Tumeur du pancréas) n'aborde pas cette question. Encore après-coups il est malade éprouvait constamment un besoin impérieux de boire et de manger, sans pouvoir le satisfaire.</p> <p>Dans son article magistral sur le diabète (1869), le professeur Jaccoud cite parmi les lésions les cas d'atrophie du pancréas. Mais en raison du petit nombre de faits observés il n'y attribue qu'une mince valeur. Encore faudrait-il prouver selon lui que cette atrophie ne constitue pas une lésion contingente, au même titre que les altérations des reins et du foie. Dans son Traité de pathologie interne, il expose la théorie pancréatique du diabète, de Popper, mais il la considère simplement comme une ingénieuse hypothèse. (Jaccoud, in Nouveau dictionnaire de méd. et de chir. pratiques, art. Diabète, 1869.)</p> <p>Klets (1870) rapporte un cas d'atrophie du pancréas chez</p>	<p style="text-align: center;">22 MARY.</p> <p>Je n'aurai pas à rappeler avec détails la disposition des appareils qui nous fournirent, sous forme de tracés, l'expression des différents actes qui se produisent dans une révolution du cœur. Ces expériences ont été exposées ailleurs (1) d'une manière complète. Rappelons seulement que nous introduisons par les veines ou par les artères du cou, jusque dans les cavités du cœur, des appareils qui explorent la pression, à la façon de manomètres très-sensibles; que chacun de ces explorateurs se rendait à un appareil inscripteur qui accusait par une ascension de la courbe les élévations, et par une descente les diminutions de la pression (2).</p> <p>(1) <i>Physiol. méd. de la circulation du sang</i>, 1863, p. 46 et suiv. — <i>Dictionnaire encyclopédique des sciences médicales</i>, Art. <i>CANNISGRAPHIE</i>.</p> <p>(2) Voici le principe de la transmission des pressions au moyen de tubes à air.</p>  <p style="text-align: center;">Fig. 8. — Transmission des mouvements par un tube à air.</p> <p>Soit (fig. 8) deux ampoules de caoutchouc B et A plinées d'air et reliées entre elles par un long tube de caoutchouc. Si l'on comprime l'ampoule B, une partie de l'air qu'elle renferme est expulsé par le tube et passe dans l'ampoule A qui se gonfle. Si la pression cesse, l'air repasse en B et l'ampoule A se dégonfle.</p> <p>Admettons que l'ampoule B soit introduite dans un ventricule du cœur et que l'ampoule A soit placée sous un levier semblable à celui du sphymographe, les systoles du ventricule seront signalées par l'élévation, et les diastoles par l'abaissement du levier.</p>  <p style="text-align: center;">Fig. 9. — Sonde cardiaque droite avec les détails de sa structure.</p> <p>graphé, les systoles du ventricule seront signalées par l'élévation, et les diastoles par l'abaissement du levier.</p> <p>Pour s'appliquer facilement aux besoins de la cardiographie, les ampoules</p>	<p style="text-align: center;">CHAPITRE II DE L'ESPACE</p> <p style="text-align: center;">Sa mesure et sa représentation par la Photographie</p> <p>Sommaire. — La Photographie tend à remplacer les dessins, les plans et les figures en relief. — Elle retrace les lieux de l'espace qu'un mobile a parcourus. — Trajectoire photographique des mouvements d'un point dans l'espace; sa trajectoire stéréoscopique. — Déplacements d'une droite dans l'espace; formes solides qu'elle engendre: cylindre, hyperboloïde, cône, etc. — Déplacements d'une courbe dans l'espace; photographie des formes qu'elle engendre: sphère, ellipsoïde, etc. — Images stéréoscopiques des figures à trois dimensions. — Formes engendrées par le déplacement de corps solides; effets de la lumière et des ombres.</p> <p>La Photographie tend à remplacer les dessins, les plans, les figures en relief. — La position des corps dans l'espace, leurs formes et leurs dimensions ont leur expression naturelle dans les plans géométriques. Ces plans, dessinés à une échelle connue, renferment tous les renseignements désirables. Toutefois, depuis quelques années, la Photographie tend à se substituer au dessin et le remplacera certainement dans la majorité des cas. Elle donne, en effet, avec une facilité singulière, des images d'une fidélité absolue; elle en réduit ou en agrandit les dimensions aux proportions que l'on désire; enfin elle permet d'introduire dans ces images elles-mêmes l'échelle métrique avec laquelle on mesurera les dimensions réelles des objets représentés. Pour avoir une échelle métrique sur l'image, il suffit de placer, à côté de l'objet qu'on photographie, une règle à divisions bien apparentes.</p> <p>Pour les corps à trois dimensions, quand le dessinateur en</p>																																																								
<p style="text-align: center;">TPAR1879x394</p>	<p style="text-align: center;">31093x01</p>	<p style="text-align: center;">ex32516</p>																																																								
<p style="text-align: center;">— 8 —</p> <p>(Voir <i>Compte rendu du Congrès de thérapeutique de 1889</i>, O. Doim, éditeur.)</p> <p>Voici une statistique établie sur 75 observations prises à l'hôpital Cochin, et qui font partie de l'intéressante thèse inaugurale du Dr Gaudinain (Paris 1889).</p> <p>Rhumatisme. — 49 cas, sur lesquels 3 sans résultats, 3 douteux chez lesquels il y a lieu de croire à la simulation, 6 cas où l'amélioration a été certaine et suivie en quinze jours ou trois semaines de guérison, 4 enfin où l'on a pu constater une action thérapeutique rapide, nette et très favorable, au point de vue de la disparition des douleurs.</p> <p>Cardialgie et accès angineux. — Sur 3 cas, le résultat a été remarquable au point de vue de la douleur, sans modifier bien entendu la lésion cardiaque qui existait dans les trois cas.</p> <p>Fièvre et sueurs nocturnes de tuberculeux. — Sur 4 cas seulement traités, résultats médiocres et nuls dans 2 cas. Deux malades ont seulement prétendu mieux dormir et être plus calmes.</p> <p>Métrites. — Sur 5 cas, deux bons résultats dans les phénomènes douloureux, 1 médiocre, 2 nuls.</p> <p>Gastralgies. — Deux bons résultats pour l'action sur les crampes d'estomac.</p> <p>Migraines. — Sur 8 cas, 4 bons résultats (points sous et supra-orbitaires), 3 résultats médiocres, 2 nuls (migraine dyspeptique).</p> <p>Névralgies. — C'est ici surtout que les effets de l'exalgin ont été remarquables, particulièrement dans les cas de névralgies à frigore et de douleurs dentaires. Sur 32 cas observés 13 succès rapides, c'est-à-dire en quelques heures disparition de la douleur, 14 succès moins rapides, c'est-à-dire où il a fallu réitérer, 2 résultats médiocres et 3 nuls; ces 5 derniers sont des sciaticques chroniques.</p> <p>A propos de ces observations, M. Gaudinain a remarqué que, dans la majorité des cas, dans les malades sur</p>	<p style="text-align: center;">786 SABOURAUD</p> <p>deux ou trois observations à peine (Walter-Smith, Umana) de cas où la maladie s'est développée tardivement: une fois à l'âge de la puberté, une autre fois chez une femme adulte, à la suite d'un dérangement nerveux. Hors ces cas, le malade a toujours débuté semblablement à la naissance. L'enfant vient au monde avec des cheveux normaux. Ces cheveux tombent vers la sixième semaine ou le deuxième mois, et ne repoussent pas. Quelques cheveux rares, grêles et moniliformes prennent leur place. Le monilithrix commença donc avec la vie pour ne finir qu'avec elle. Et comme les autres difformités congénitales, le monilithrix est héréditaire, dans une plus forte proportion même, que les plus transmises des malformations: les doigts palmés ou supplémentaires, l'albinisme, l'hypospadias.</p> <p>L'observation de Mac Call Anderson concerne quatorze membres de la même famille, celle de Payne, deux frères; celle de M. Hallopeau cinq individus en deux générations.</p> <p>L'observation que je viens de recueillir est je crois sous ce rapport la plus complète, car elle comprend dix-sept membres de la même famille en cinq générations.</p> <p>(7) Enfin les trichorhexies parasitaires sont guérissables, cela va sans dire, il suffit de supprimer la malpropreté causale: le monilithrix, nous l'avons vu, est permanent. Aussi est-ce un hasard d'en rencontrer un exemple à l'hôpital. Le plus souvent le patient sait son infirmité héréditaire et incurable.</p> <p>Il semble évident, après ce résumé très succinct, que la confusion entre les deux groupes d'affections soit impossible. Peut-être cependant, faut-il tenir compte dans cette rapide analyse, de quelques observations moins typiques que celles auxquelles nous avons fait allusion. Peut-être faudra-t-il dans l'avenir, distinguer dans les monilithrix non parasitaires, comme dans les trichorhexies d'origine externe, des catégories diverses, que des symptômes analogues rapprochent mais que des différences encore inaperçues séparent. Mais, cette réserve faite, il semble que les déformations moniliformes du poil appartiennent dans certains cas à une maladie parasitaire du groupe des trichorhexies externes; dans les autres à une malformation tenant à un trouble trophique non encore défini: monilithrix ou aplasie moniliforme.</p> <p>L'observation qui suit, constituant un type de la maladie décrite par Lallier et Lucas, Walter Smith et Living, Mac Call Anderson, etc., maladie à laquelle Crocker a donné son nom de monilithrix, nous semble propre à éclairer la question que nous venons de résumer.</p> <p style="text-align: center;">III</p> <p>Le 5 juin 1892, entra dans le service de mon cher et éminent maître,</p>	<p style="text-align: center;">— 82 —</p> <p>128. Nécessaires d'instruments pour la vivisection dans la physiologie expérimentale. Choix de D^r Laborde et de Ch. Verdin 750 »</p> <p>Ce nécessaire, renfermé dans une boîte en ébène ayant 50^m/₁₀ de long, 30^m/₁₀ de large et 10^m/₁₀ d'épaisseur, comprend les instruments ci-dessous :</p> <p style="text-align: center;">INSTRUMENTS POUR LA VIVISECTION ET LA DISSECTION</p> <table border="0"> <tr> <td>1 Couteau à cartilages.</td> <td>2 Pinces à pression continue petites courbes.</td> </tr> <tr> <td>1 Scalpel fort.</td> <td>4 Pinces à pression continue moyennes droites et courbes.</td> </tr> <tr> <td>7 Scalpels ordinaires dont 2 convexes.</td> <td>6 Pinces à pression continue petites droites.</td> </tr> <tr> <td>2 — — moyens.</td> <td>1 Pince porte-aiguille de Collin.</td> </tr> <tr> <td>1 — fin.</td> <td>2 — — complètes à ressort de Claude Bernard.</td> </tr> <tr> <td>2 Aiguilles de Deschamps délicate, à droite.</td> <td>1 Pince petite de Linton coude avec ressort.</td> </tr> <tr> <td>2 Aiguilles de Deschamps petites, droite et gauche.</td> <td>1 Tincotome de Long.</td> </tr> <tr> <td>1 Aiguille de Deschamps droite et gauche.</td> <td>1 Costotome grand modèle.</td> </tr> <tr> <td>2 Aiguilles de Deschamps délicate, courbées à plat.</td> <td>1 Aiguille de Reverdin fixe.</td> </tr> <tr> <td>1 Aiguille de Deschamps droite et gauche.</td> <td>2 Sondes cannelées de 12^m/₁₀ en acier.</td> </tr> <tr> <td>2 Aiguilles de Deschamps délicate, courbées à plat.</td> <td>2 — — — 12^m/₁₀ —</td> </tr> <tr> <td>1 Aiguille de Deschamps droite et gauche.</td> <td>2 — — — 12^m/₁₀ —</td> </tr> <tr> <td>1 Instrument pour la section du grand sympathique.</td> <td>4 Marteau à crochet n° 4.</td> </tr> <tr> <td>1 Instrument pour la section de la 3^e paire.</td> <td>1 Perforateur avec manivelle et une couronne de 12^m/₁₀.</td> </tr> <tr> <td>1 Instrument pour la section du pneumo-gastrique.</td> <td>1 Petite acie à dos mobile, monture métallique.</td> </tr> <tr> <td>1 Couteau à manche mobile pour le bulbe.</td> <td>1 Levier à rugine.</td> </tr> <tr> <td>2 Ecarteurs doubles en acier nickelé, de Paronoff.</td> <td>1 Gorge forte coude de Ribet.</td> </tr> <tr> <td>2 Crochets doubles en S.</td> <td>1 Entérotoque de Pansa.</td> </tr> <tr> <td>2 Pinces à verrou démontant.</td> <td>1 Ciseaux n° 6 avec lames pointues.</td> </tr> <tr> <td>1 — 3 griffes.</td> <td>1 — — à lames mousses.</td> </tr> <tr> <td>1 — 9 —</td> <td>1 — à dissection dont 2 courbes.</td> </tr> <tr> <td>1 — à dissection à mors fins.</td> <td>1 — — droits à 2 pointes.</td> </tr> <tr> <td>1 — moyenne.</td> <td>1 — — fins à lames mousses.</td> </tr> <tr> <td>2 — fines; droite et courbe.</td> <td>1 — — lames pointues.</td> </tr> <tr> <td>4 — de Pons, de 12^m/₁₀.</td> <td>6 Serre-fines assorties.</td> </tr> <tr> <td>2 — — de 14^m/₁₀.</td> <td>2 Rouleaux de fil d'argent.</td> </tr> <tr> <td></td> <td>1 Pompe à sang avec aiguille et robinet à 2 volets.</td> </tr> <tr> <td></td> <td>3 Panses-fils dont 2 droites et 1 courb.</td> </tr> </table>	1 Couteau à cartilages.	2 Pinces à pression continue petites courbes.	1 Scalpel fort.	4 Pinces à pression continue moyennes droites et courbes.	7 Scalpels ordinaires dont 2 convexes.	6 Pinces à pression continue petites droites.	2 — — moyens.	1 Pince porte-aiguille de Collin.	1 — fin.	2 — — complètes à ressort de Claude Bernard.	2 Aiguilles de Deschamps délicate, à droite.	1 Pince petite de Linton coude avec ressort.	2 Aiguilles de Deschamps petites, droite et gauche.	1 Tincotome de Long.	1 Aiguille de Deschamps droite et gauche.	1 Costotome grand modèle.	2 Aiguilles de Deschamps délicate, courbées à plat.	1 Aiguille de Reverdin fixe.	1 Aiguille de Deschamps droite et gauche.	2 Sondes cannelées de 12 ^m / ₁₀ en acier.	2 Aiguilles de Deschamps délicate, courbées à plat.	2 — — — 12 ^m / ₁₀ —	1 Aiguille de Deschamps droite et gauche.	2 — — — 12 ^m / ₁₀ —	1 Instrument pour la section du grand sympathique.	4 Marteau à crochet n° 4.	1 Instrument pour la section de la 3 ^e paire.	1 Perforateur avec manivelle et une couronne de 12 ^m / ₁₀ .	1 Instrument pour la section du pneumo-gastrique.	1 Petite acie à dos mobile, monture métallique.	1 Couteau à manche mobile pour le bulbe.	1 Levier à rugine.	2 Ecarteurs doubles en acier nickelé, de Paronoff.	1 Gorge forte coude de Ribet.	2 Crochets doubles en S.	1 Entérotoque de Pansa.	2 Pinces à verrou démontant.	1 Ciseaux n° 6 avec lames pointues.	1 — 3 griffes.	1 — — à lames mousses.	1 — 9 —	1 — à dissection dont 2 courbes.	1 — à dissection à mors fins.	1 — — droits à 2 pointes.	1 — moyenne.	1 — — fins à lames mousses.	2 — fines; droite et courbe.	1 — — lames pointues.	4 — de Pons, de 12 ^m / ₁₀ .	6 Serre-fines assorties.	2 — — de 14 ^m / ₁₀ .	2 Rouleaux de fil d'argent.		1 Pompe à sang avec aiguille et robinet à 2 volets.		3 Panses-fils dont 2 droites et 1 courb.
1 Couteau à cartilages.	2 Pinces à pression continue petites courbes.																																																									
1 Scalpel fort.	4 Pinces à pression continue moyennes droites et courbes.																																																									
7 Scalpels ordinaires dont 2 convexes.	6 Pinces à pression continue petites droites.																																																									
2 — — moyens.	1 Pince porte-aiguille de Collin.																																																									
1 — fin.	2 — — complètes à ressort de Claude Bernard.																																																									
2 Aiguilles de Deschamps délicate, à droite.	1 Pince petite de Linton coude avec ressort.																																																									
2 Aiguilles de Deschamps petites, droite et gauche.	1 Tincotome de Long.																																																									
1 Aiguille de Deschamps droite et gauche.	1 Costotome grand modèle.																																																									
2 Aiguilles de Deschamps délicate, courbées à plat.	1 Aiguille de Reverdin fixe.																																																									
1 Aiguille de Deschamps droite et gauche.	2 Sondes cannelées de 12 ^m / ₁₀ en acier.																																																									
2 Aiguilles de Deschamps délicate, courbées à plat.	2 — — — 12 ^m / ₁₀ —																																																									
1 Aiguille de Deschamps droite et gauche.	2 — — — 12 ^m / ₁₀ —																																																									
1 Instrument pour la section du grand sympathique.	4 Marteau à crochet n° 4.																																																									
1 Instrument pour la section de la 3 ^e paire.	1 Perforateur avec manivelle et une couronne de 12 ^m / ₁₀ .																																																									
1 Instrument pour la section du pneumo-gastrique.	1 Petite acie à dos mobile, monture métallique.																																																									
1 Couteau à manche mobile pour le bulbe.	1 Levier à rugine.																																																									
2 Ecarteurs doubles en acier nickelé, de Paronoff.	1 Gorge forte coude de Ribet.																																																									
2 Crochets doubles en S.	1 Entérotoque de Pansa.																																																									
2 Pinces à verrou démontant.	1 Ciseaux n° 6 avec lames pointues.																																																									
1 — 3 griffes.	1 — — à lames mousses.																																																									
1 — 9 —	1 — à dissection dont 2 courbes.																																																									
1 — à dissection à mors fins.	1 — — droits à 2 pointes.																																																									
1 — moyenne.	1 — — fins à lames mousses.																																																									
2 — fines; droite et courbe.	1 — — lames pointues.																																																									
4 — de Pons, de 12 ^m / ₁₀ .	6 Serre-fines assorties.																																																									
2 — — de 14 ^m / ₁₀ .	2 Rouleaux de fil d'argent.																																																									
	1 Pompe à sang avec aiguille et robinet à 2 volets.																																																									
	3 Panses-fils dont 2 droites et 1 courb.																																																									
<p style="text-align: center;">90958x1028x012</p>	<p style="text-align: center;">EPO0843</p>	<p style="text-align: center;">53034x01</p>																																																								

Web reference of each book is given along with. Complete book can be viewed on the link [BIUM] by looking for this particular reference.

A - Sample Images

A.2 BIUM Sample images of the older books of the 16th century



A.3 Contemporary document images

Journal of Bacteriology, Apr. 1975, p. 224-234
Copyright © 1975, American Society for Microbiology

Vol. 122, No. 4
Printed in U.S.A.

Regulation of the Dicarboxylic Acid Part of the Citric Acid Cycle in *Bacillus subtilis*

MARGARETA ÖHNÉ
Department of Bacteriology, Karolinska Institute, S 104 01 Stockholm 60, Sweden
Received for publication 24 January 1975

The regulation of α -ketoglutarate dehydrogenase, succinate dehydrogenase, fumarate, malate dehydrogenase, and malic enzyme has been studied in *Bacillus subtilis*. The levels of these enzymes increase rapidly during late exponential phase in a complex medium and are maximal 1 to 2 h after the onset of sporulation. Regulation of enzyme synthesis has been studied in the wild type and different citric acid cycle mutants by adding various metabolites to the growth medium. α -Ketoglutarate dehydrogenase is induced by glutamate or α -ketoglutarate; succinate dehydrogenase is repressed by malate, and fumarate and malic enzyme are induced by fumarate and malate, respectively. The addition of glucose leads to repression of the citric acid cycle enzymes whereas the level of malic enzyme is unaffected. Studies on the control of enzyme activities *in vitro* have shown that α -ketoglutarate dehydrogenase and succinate dehydrogenase are inhibited by oxalacetate. Enzyme activities are also influenced by the energy level, expressed as the energy charge of the adenylate pool. Isocitrate dehydrogenase, α -ketoglutarate dehydrogenase, succinate dehydrogenase, and malic enzyme are inhibited at high energy charge values, whereas malate dehydrogenase is inhibited at low energy charge. A survey of the regulation of the citric acid cycle in *B. subtilis*, based on the present work and previously reported results, is presented and discussed.

The control of the citric acid cycle in *Bacillus subtilis* has been studied by several authors. Most investigations have concerned the first three enzymes of the cycle, citrate synthase (EC 4.1.3.1, citrate acetyl-CoA lyase [coenzyme A acetylating]), aconitase (EC 4.2.1.3, citrate [succinate] hydro-lyase), and isocitrate dehydrogenase (EC 1.1.1.42, *isero*- β -isocitrate: nicotinamide adenine dinucleotide phosphate oxidoreductase [decarboxylating]). These enzymes are responsible for the synthesis of α -ketoglutarate, and they have been shown to be subject to feedback repression by α -ketoglutarate or glutamate (12, 14, 22, 30). In addition, the synthesis of aconitase is induced by citrate (30). *In vitro* studies on citrate synthase and isocitrate dehydrogenase have shown that these two enzymes are inhibited by adenine triphosphate (ATP) (12, 13).

In contrast, very little is known about the control of the dicarboxylic acid part of the citric acid cycle, responsible for the oxidation of α -ketoglutarate to oxalacetate. Mutants of *B. subtilis* defective in α -ketoglutarate dehydrogenase, succinate dehydrogenase (EC 1.3.99.1, succinate [acceptor] oxidoreductase), fumarate (EC 4.2.1.2, *malate* hydro-lyase), or malate dehydrogenase (EC 1.1.1.37, *malate*:nicotinamide adenine dinucleotide oxidoreductase) have been isolated and characterized genetically (23) and enzymatically (8, 23), but their regulatory properties have not been studied.

The finding that *B. subtilis* mutants lacking malate dehydrogenase are able to convert malate to glutamate, led Carlis and Hanson (8) to postulate the existence of malic enzyme (EC 1.1.1.40, *malate*:nicotinamide adenine dinucleotide phosphate oxidoreductase [decarboxylating]) in this organism. Malic enzyme catalyzes the oxidative decarboxylation of malate to pyruvate, an essential anaplerotic reaction during growth on C₃ compounds. This activity was recently demonstrated in extracts of *B. subtilis* (10).

In this work, I describe the regulation of the dicarboxylic acid portion of the citric acid cycle as well as the regulation of malic enzyme in *B. subtilis*.

MATERIALS AND METHODS

Bacteria. All bacterial strains used in this study were derived from *B. subtilis* 168. The isolation and characterization of the citric acid cycle mutants have been reported earlier (31, 33). Mutants lacking malate dehydrogenase (EC 1.1.1.37, *malate*:nicotinamide adenine dinucleotide oxidoreductase) were identified as follows: **Ch K3**, **Ch K5**, **Ch F78**, and **Ch G4**; mutants lacking succinate dehydrogenase (EC 1.3.99.1, succinate [acceptor] oxidoreductase) were identified as follows: **Ch K1**, **Ch K2**, **Ch K3**, **Ch K4**, **Ch K5**, **Ch K6**, **Ch K7**, **Ch K8**, **Ch K9**, **Ch K10**, **Ch K11**, **Ch K12**, **Ch K13**, **Ch K14**, **Ch K15**, **Ch K16**, **Ch K17**, **Ch K18**, **Ch K19**, **Ch K20**, **Ch K21**, **Ch K22**, **Ch K23**, **Ch K24**, **Ch K25**, **Ch K26**, **Ch K27**, **Ch K28**, **Ch K29**, **Ch K30**, **Ch K31**, **Ch K32**, **Ch K33**, **Ch K34**, **Ch K35**, **Ch K36**, **Ch K37**, **Ch K38**, **Ch K39**, **Ch K40**, **Ch K41**, **Ch K42**, **Ch K43**, **Ch K44**, **Ch K45**, **Ch K46**, **Ch K47**, **Ch K48**, **Ch K49**, **Ch K50**, **Ch K51**, **Ch K52**, **Ch K53**, **Ch K54**, **Ch K55**, **Ch K56**, **Ch K57**, **Ch K58**, **Ch K59**, **Ch K60**, **Ch K61**, **Ch K62**, **Ch K63**, **Ch K64**, **Ch K65**, **Ch K66**, **Ch K67**, **Ch K68**, **Ch K69**, **Ch K70**, **Ch K71**, **Ch K72**, **Ch K73**, **Ch K74**, **Ch K75**, **Ch K76**, **Ch K77**, **Ch K78**, **Ch K79**, **Ch K80**, **Ch K81**, **Ch K82**, **Ch K83**, **Ch K84**, **Ch K85**, **Ch K86**, **Ch K87**, **Ch K88**, **Ch K89**, **Ch K90**, **Ch K91**, **Ch K92**, **Ch K93**, **Ch K94**, **Ch K95**, **Ch K96**, **Ch K97**, **Ch K98**, **Ch K99**, **Ch K100**.

228 ÖHNÉ J. BACTERIOL.

TABLE 1. Specific activities of succinate dehydrogenase, fumarate, and malate dehydrogenase in *B. subtilis* wild type and different mutants

Strain*	Enzyme defect†	Specific activity‡		
		SDH	FUM	MAL
Wild type	None	1,170	1,500	130
Ch K3	KDH	2,185	850	23
Ch F78	SDH	<1	920	32
Ch G4	FUM	1,925	<1	15
Ch K1	MAL	180	ND†	<1

* Bacteria were grown in NSMP medium and were harvested at 16 h.

† KDH, α -Ketoglutarate dehydrogenase; SDH, succinate dehydrogenase; FUM, fumarate; and MAL, malic enzyme.

‡ ND, Not determined.

Fig. 4. Effect of fumarate on growth (A) and the level of fumarate (B) in *B. subtilis* wild type. Symbols: ●, unamplified NSMP medium; ○, NSMP plus 25 mM fumarate. Arrows indicate *t*₀ in the unamplified culture.

Fig. 5. Effect of fumarate and malate on growth (A, B) and the levels of succinate dehydrogenase (C, D) and malic enzyme (E, F) in *B. subtilis* wild type (A, C, E) and *ChG4* (B, D, F). Symbols: ●, unamplified NSMP medium; ○, NSMP plus 25 mM fumarate; ▲, NSMP plus 25 mM malate.

Fig. 6. Effect of glutamate and aspartate on growth (A) and the level of α -ketoglutarate dehydrogenase (B) in *B. subtilis* wild type. Symbols: ●, unamplified NSMP medium; ○, NSMP plus 25 mM glutamate; ▲, NSMP plus 25 mM aspartate. Arrows indicate *t*₀ in all three cultures.

Fig. 7. Effect of fumarate and malate on growth (A, B) and the levels of succinate dehydrogenase (C, D) and malic enzyme (E, F) in *B. subtilis* wild type (A, C, E) and *ChG4* (B, D, F). Symbols: ●, unamplified NSMP medium; ○, NSMP plus 25 mM fumarate; ▲, NSMP plus 25 mM malate.

Fig. 8. Effect of fumarate and malate on growth (A, B) and the levels of succinate dehydrogenase (C, D) and malic enzyme (E, F) in *B. subtilis* wild type (A, C, E) and *ChG4* (B, D, F). Symbols: ●, unamplified NSMP medium; ○, NSMP plus 25 mM fumarate; ▲, NSMP plus 25 mM malate.

Vol. 122, 1975 CITRIC ACID CYCLE OF *B. SUBTILIS* 227

Fig. 4. Effect of fumarate on growth (A) and the level of fumarate (B) in *B. subtilis* wild type. Symbols: ●, unamplified NSMP medium; ○, NSMP plus 25 mM fumarate. Arrows indicate *t*₀ in the unamplified culture.

Fig. 5. Effect of fumarate and malate on growth (A, B) and the levels of succinate dehydrogenase (C, D) and malic enzyme (E, F) in *B. subtilis* wild type (A, C, E) and *ChG4* (B, D, F). Symbols: ●, unamplified NSMP medium; ○, NSMP plus 25 mM fumarate; ▲, NSMP plus 25 mM malate.

Fig. 6. Effect of glutamate and aspartate on growth (A) and the level of α -ketoglutarate dehydrogenase (B) in *B. subtilis* wild type. Symbols: ●, unamplified NSMP medium; ○, NSMP plus 25 mM glutamate; ▲, NSMP plus 25 mM aspartate. Arrows indicate *t*₀ in all three cultures.

Fig. 7. Effect of fumarate and malate on growth (A, B) and the levels of succinate dehydrogenase (C, D) and malic enzyme (E, F) in *B. subtilis* wild type (A, C, E) and *ChG4* (B, D, F). Symbols: ●, unamplified NSMP medium; ○, NSMP plus 25 mM fumarate; ▲, NSMP plus 25 mM malate.

Fig. 8. Effect of fumarate and malate on growth (A, B) and the levels of succinate dehydrogenase (C, D) and malic enzyme (E, F) in *B. subtilis* wild type (A, C, E) and *ChG4* (B, D, F). Symbols: ●, unamplified NSMP medium; ○, NSMP plus 25 mM fumarate; ▲, NSMP plus 25 mM malate.

Vol. 122, 1975 CITRIC ACID CYCLE OF *B. SUBTILIS* 233

to grow on minimal glucose plates without added succinyl CoA (or succinate). Since the glyoxylate shunt is not operative in *B. subtilis* (17), unutilized (oxidized) succinyl CoA must be synthesized by reduction of oxalacetate in these mutants. We do not know to what extent the reductive branch contributes to succinyl CoA synthesis in the wild-type strain. The regulatory mechanisms identified for Fig. 5 are all seen to promote the oxidative functions of the cycle, but it cannot be excluded that the reductive branch plays an important role under other growth conditions than those used here.

In conclusion, the proper functioning of the citric acid cycle in *B. subtilis* depends on a number of regulatory mechanisms responding to the energy balance and the nutritional status of the cell. Substrate induction mechanisms permit the efficient utilization of certain citric acid cycle intermediates as carbon sources. Feedback repression and feedback inhibition mechanisms control the production of intermediates needed in biosynthetic pathways. The energy charge control system adjusts the activity of the cycle to the energy demand of the cell. Finally, when the ATP requirement can be met with glycolysis, the synthesis of the citric acid cycle enzymes is repressed so that only the biosynthetic functions are fulfilled.

Information on the regulation of the citric acid cycle in other bacteria is scarce. It is known, however, that in *E. coli*, aconitase, isocitrate dehydrogenase are repressed by glutamate (24), α -ketoglutarate dehydrogenase is probably induced by α -ketoglutarate (3); citrate synthase and isocitrate dehydrogenase are inhibited by ATP (13); and all the citric acid cycle enzymes are repressed by glucose (21). Thus, at least some features of the control of the citric acid cycle are similar in as widely different species as the facultative gram-negative *E. coli* and the aerobic, gram-positive, spore-forming *B. subtilis*.

ACKNOWLEDGMENTS

This work was supported by grant B75-165-3038-06 from the Swedish Medical Research Council and grants from Karolinska Institute's Forskningsfond.

Technical assistance was provided by Labetz Klitz.

LITERATURE CITED

- Adair, A. C., R. B. Kearney, and M. May. 1974. Regulation of malic enzyme in the regulation of succinate dehydrogenase. *J. Biol. Chem.* 249:1027-1032.
- Andersen, J. A., M. Marini, and Chiu-Chia Chen. 1965. Optimal conditions for propagation by a defined *in vitro* medium of *Bacillus subtilis* K 12. *Biotechnol. Bioeng. Commun.* 10:79-92.
- Anwarhaghighi, C. R., and R. D. Dixon. 1963. Regulation of α -ketoglutarate dehydrogenase formation in *Bacillus subtilis*. *J. Biol. Chem.* 238:2984-2988.
- Ashkin, D. R. 1965. The energy charge of the adenylate pool as a regulatory parameter in bacteria with feedback inhibition. *Biochemistry* 4:400-404.
- Bachmann, R. D., W. H. Peterson, and D. E. Green. 1965. The membrane systems of the mitochondrion. I. The D fraction of the outer membrane of beef heart mitochondria. *Arch. Biochem. Biophys.* 118:153-164.
- Carlis, R. A., and R. S. Hanson. 1971. Isolation and characterization of *Bacillus subtilis* mutants defective in the citric acid cycle. *J. Bacteriol.* 106:660-665.
- Chappuis, A. G., L. Fink, and D. E. Ashkin. 1971. Adenylate energy charge in *Escherichia coli* during growth and starvation. *J. Bacteriol.* 108:1070-1066.
- Dierckx, M. C., and E. P. P. P. 1973. Role of pyruvate carboxylase, phosphoenolpyruvate carboxylase, and malic enzyme during growth and sporulation of *Bacillus subtilis*. *J. Biol. Chem.* 248:6042-6049.
- Fennel, S. K., and H. W. Tabor. 1973. Physiological effects of succinyl-CoA deficiency in *Bacillus subtilis*. *J. Bacteriol.* 114:202-204.
- Fleischer, Y. R., and R. S. Hanson. 1969. Coarse and fine control of citrate synthase from *Bacillus subtilis*. *Biochim. Biophys. Acta* 184:282-287.
- Fleischer, Y. R., and R. S. Hanson. 1970. Regulation of the citric acid cycle in bacteria. A comparison of citrate synthase and different factors. *Biochim. Biophys. Acta* 222:235-244.
- Fortgang, P. 1970. The regulation of aconitase and succinate dehydrogenase in operon mutants of *Bacillus subtilis*. *Biochim. Biophys. Acta* 222:290-296.
- Fortgang, P., and E. P. P. 1968. Analysis of operon mutants of *Bacillus subtilis* in the citric acid cycle. *J. Bacteriol.* 94:1431-1438.
- Freese, E., F. Fortgang, R. Schmitt, W. Kladt, E. Chappuis, and G. Pasticus. 1969. Biochemical genetics of citrate synthase genes. *Z. Naturforsch.* 24b:1-11. *Comp. Rend. (Paris)* 269:145-148.
- Freese, E., and U. Fortgang. 1969. Growth and sporulation of *Bacillus subtilis* mutants blocked in the pyruvate dehydrogenase complex. *J. Bacteriol.* 99:745-756.
- Freese, E., B., and C. E. Martin. 1973. Developmental block in citric acid cycle mutants of *Bacillus subtilis*. *J. Bacteriol.* 118:1466-1469.
- Freese, E., S. W. Park, and M. Capel. 1964. The regulatory significance of aconitase dehydrogenase in *Bacillus subtilis*. *Proc. Natl. Acad. Sci. U.S.A.* 51:106-111.
- Ghai, O. K., and W. W. Kay. 1973. Properties of an adenylate energy charge system in *Bacillus subtilis*. *J. Bacteriol.* 114:67-73.
- Gray, C. J., J. W. Whittney, and W. R. Mouton. 1964. Regulation of metabolism in facultative bacteria. II. Effects of acetone, succinate, and succinyl-CoA on the formation of Krebs cycle enzymes in *Escherichia coli*. *Microbios* 10:173-181.
- Hanson, R. S., and D. P. Cox. 1967. Effect of different metabolites on the synthesis of citric acid cycle enzymes. *J. Bacteriol.* 93:1777-1787.
- Hanson, R. S., V. R. Stinson, and G. H. Havelson. 1963. Biochemistry of sporulation. I. Metabolism of citrate by vegetative and sporulating cells. *J. Bacteriol.* 86:461-466.
- Hanson, R. S., V. R. Stinson, and G. H. Havelson. 1963. Biochemistry of sporulation. II. Enzymatic changes during sporulation of *Bacillus cereus*. *J. Bacteriol.* 86:467-474.

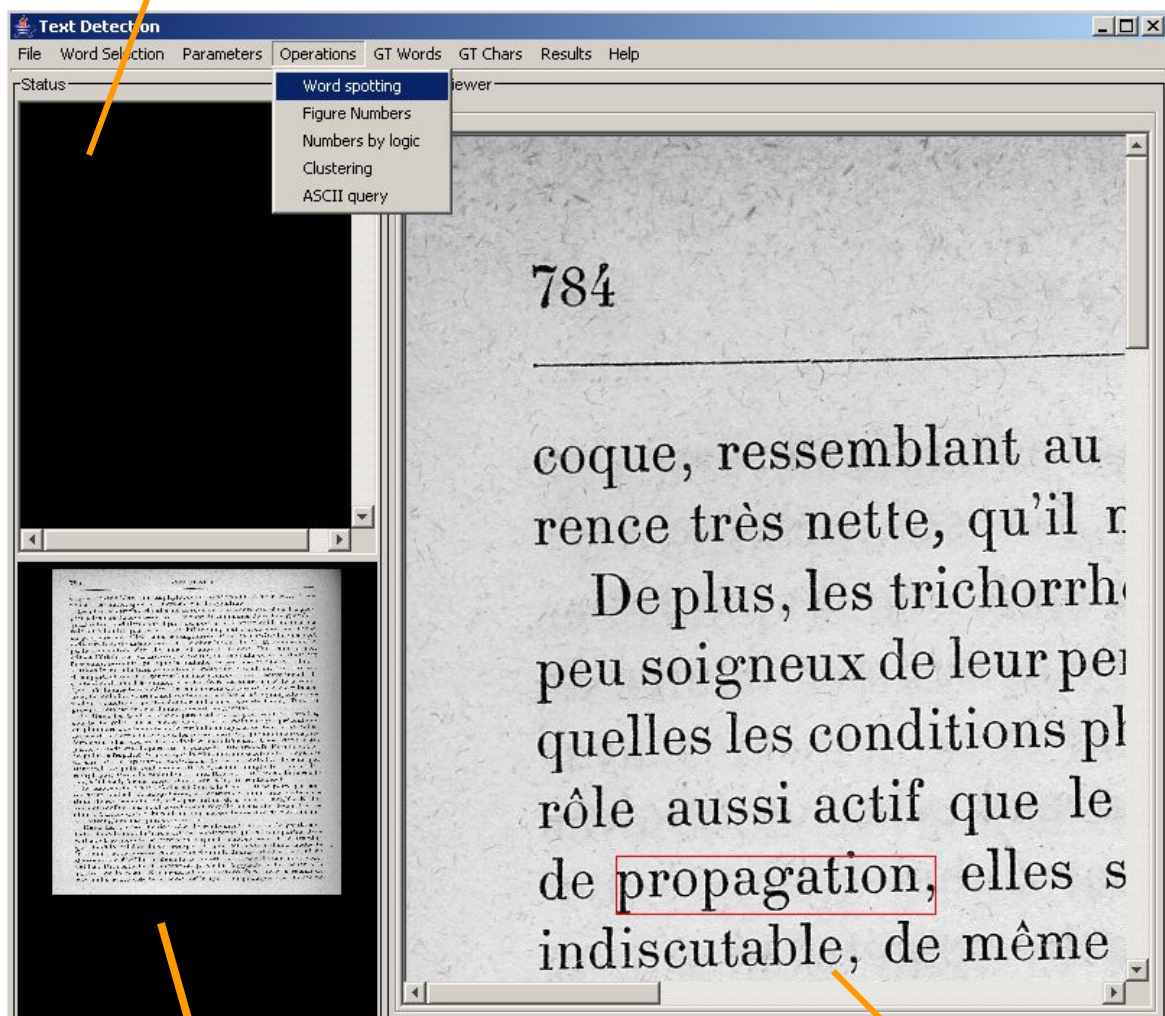
Appendix B

Graphical Interface of our system

Sample screen shots of the document retrieval graphical user interface are shown here.

1. User opens a document image using File -> open and then clicks on a word to select it as a query word image to be used in word spotting (operations -> word spotting).

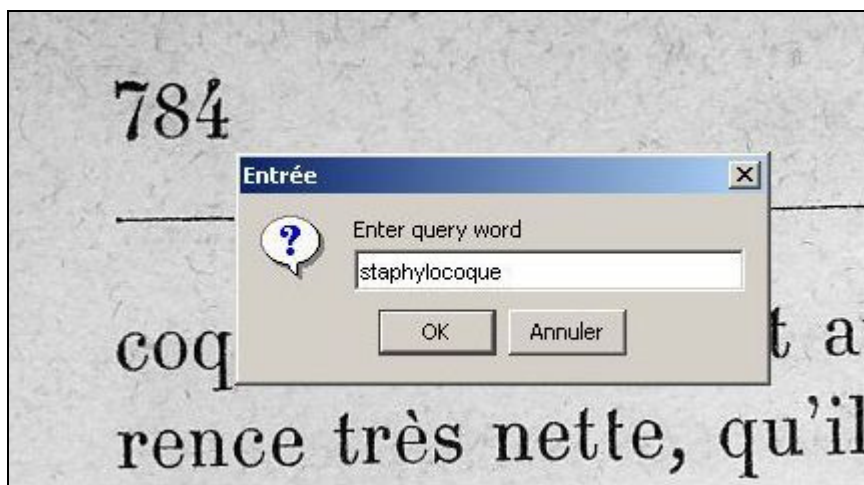
Status panel indicating the progress during search and names of the document images retrieved at the end



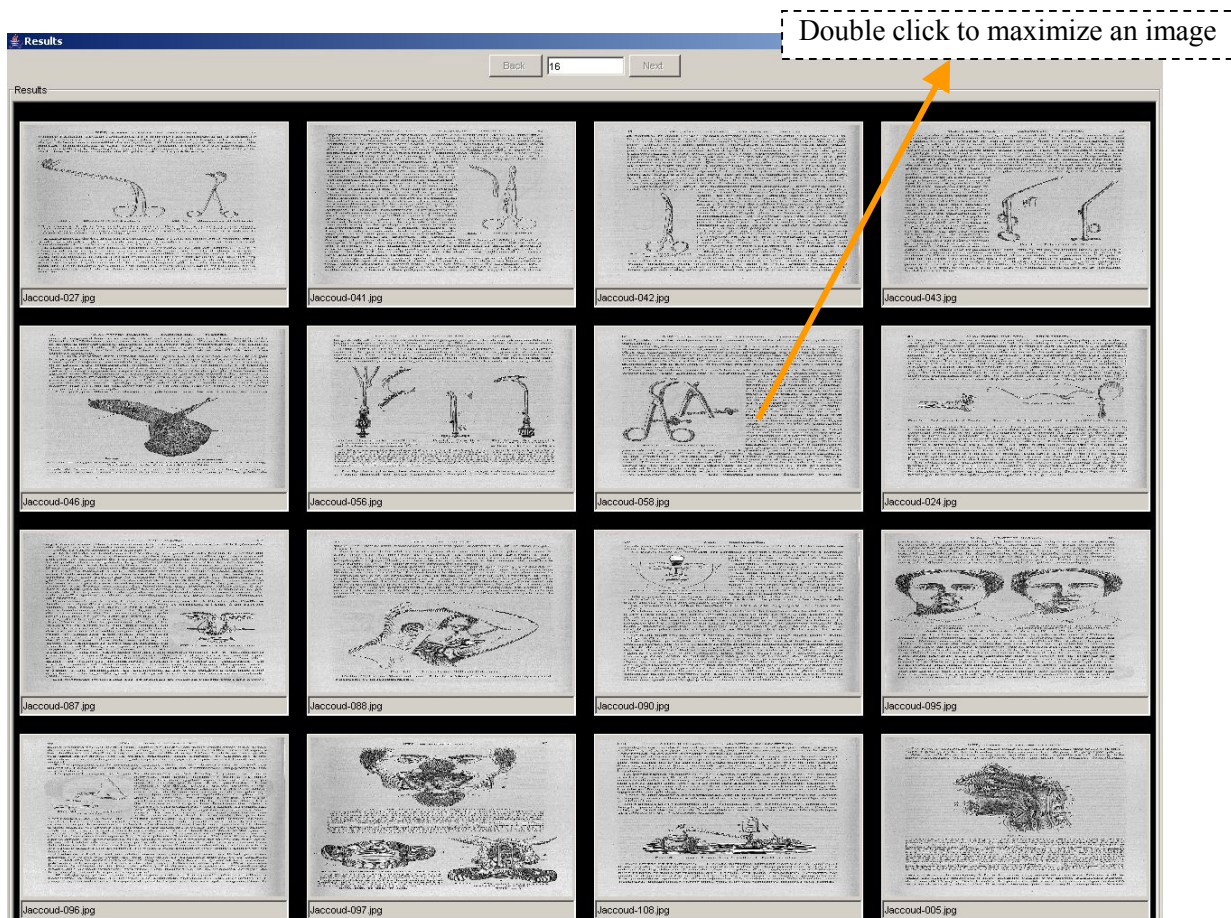
Preview of the whole page

Document image in large size

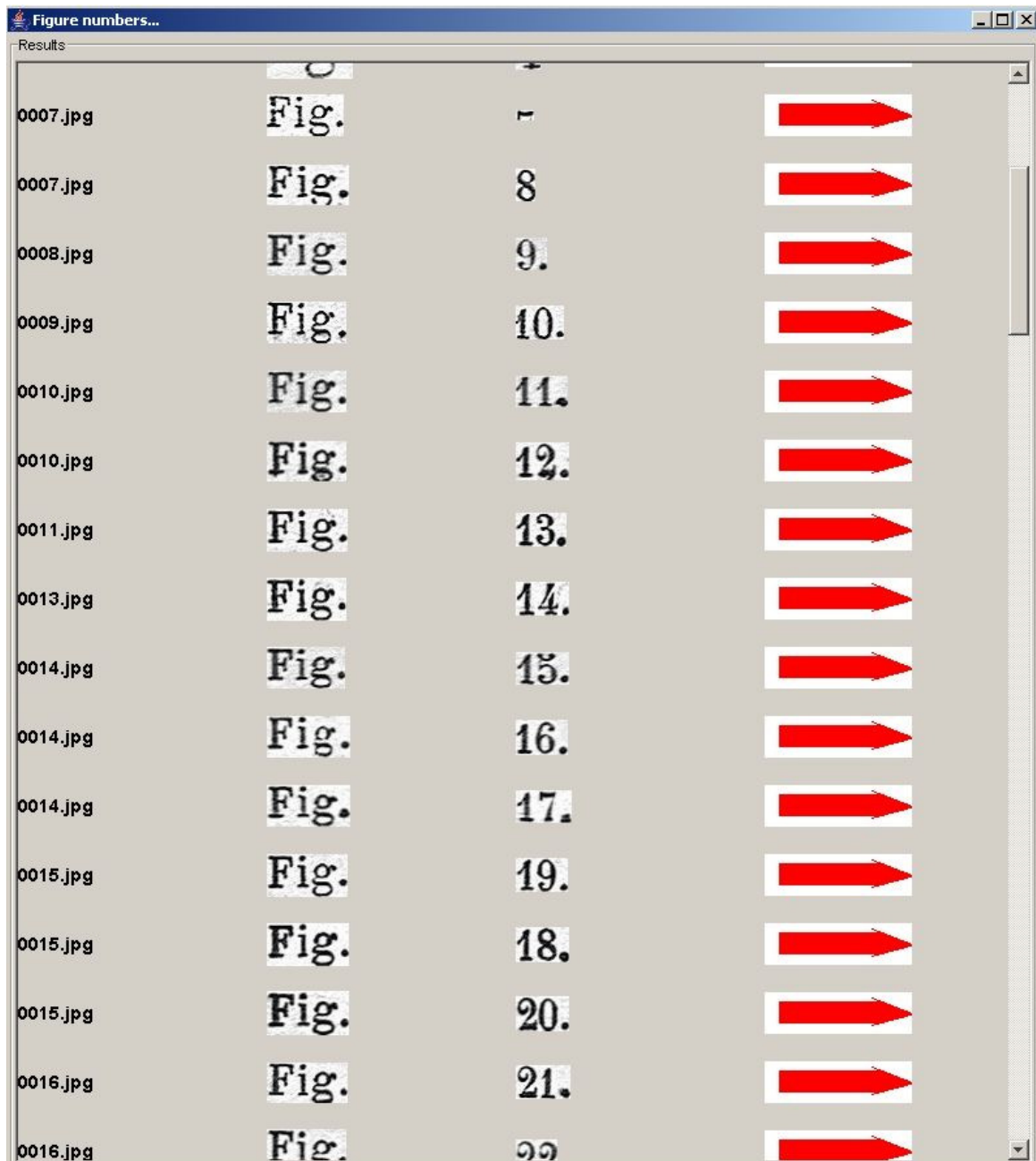
- The query word can also be given in ASCII format. User clicks (Operations -> ASCII query) to give the query in ASCII which is then searched in the document base.



- All the document images containing the query word are retrieved in an image preview frame in a chronological order. Double clicking on a document image will open it in a view pane. It can be zoomed in, zoomed out and scrolled in that pane.



- (Operations -> Fig numbers) gives us all the caption labels in a results panel. The caption label (Fig, Figure, etc.) are retrieved along with the document image name and figure numbers (figure number = word component following the caption label). By clicking on the red arrow next to a figure number, the complete document image is opened in a separate panel.



Appendix C

Summary in French

L'importance des bibliothèques numériques pour la recherche d'information ne peut pas être niée. Les livres historiques anciens contiennent une information de valeur inestimable. Mais quand les livres anciens ne sont pas transformés en version électronique, le temps nécessaire pour rechercher l'information dans ces livres papier est considérable. Néanmoins la présentation sur écran des images des documents numérisés n'est pas suffisante pour rendre l'information accessible. Notre travail, dans ce domaine, vise à faciliter la recherche de l'information en repérant des occurrences de mots dans les images des pages. Avec cette capacité de recherche dans les documents historiques anciens, les bibliothèques numériques augmenteront encore plus leur importance. Repérer des mots (*Word Spotting*) dans les documents écrits avec l'alphabet latin a suscité récemment une attention considérable. Bien que beaucoup de travaux aient été déjà effectués dans le domaine de la caractérisation des mots, il reste toujours un champ de recherche car les résultats obtenus jusqu'ici ne sont pas suffisants pour traiter des volumes de données importants; en particulier si la base de documents se compose d'un ensemble de documents imprimés anciens de qualité relativement dégradée, ce qui est propre aux documents composés à la main et imprimés sur des presses mécaniques.

Les bibliothèques et les musées à travers le monde contiennent des vastes collections de documents historiques anciens imprimés ou écrits dans différentes langues. En général, seul un groupe restreint de personnes est autorisée à accéder à ces collections, car la préservation des originaux est très préoccupante. Ces dernières années, les bibliothèques ont commencé à numériser des corpus de documents historiques qui ont de l'intérêt pour un large éventail de personnes, dans le but de rendre les contenus des documents disponibles, via les médias électroniques, tout en évitant la manipulation des originaux. Les collections historiques intéressent plusieurs catégories de lecteurs, comme les historiens, les étudiants et les universitaires qui ont besoin d'étudier les originaux historiques. Malheureusement, la numérisation seule ne suffit pas à rendre les collections de documents historiques utiles à des fins de recherche. Après numérisation, les documents sont au format d'image électronique qui permet de les visualiser et de les

rendre accessibles pour de nombreux lecteurs distants de leur lieu de stockage par l'intermédiaire d'Internet, DVD, ou autres médias numériques. Toutefois, la taille d'une collection est souvent importante et le contenu est généralement peu structuré, ce qui rend difficile de trouver rapidement des documents particuliers ou des passages présentant d'intérêt pour le lecteur. Diverses solutions à ce problème reposent entièrement sur un travail humain qui pourrait être envisagé. Un moyen simple pour structurer une collection de documents historiques est d'ordonner chronologiquement les pages et d'annoter manuellement chacune d'elle pour l'indexation. Un très haut niveau de détail dans l'annotation de contenu peut être atteint avec la transcription. Elle permet la recherche plein texte à l'aide d'un moteur traditionnel de fouille de texte. Mais les OCRs ne permettent pas de réaliser une transcription de qualité suffisante ce qui nécessite une intervention humaine. Il existe des logiciels professionnels d'OCR conçus pour différentes langues, en particulier les alphabets latins, qui donnent d'excellents résultats sur des images scannées des documents contemporains de bonne qualité. Mais lorsqu'ils sont utilisés sur d'anciens documents qui souffrent de dégradations due à l'encre fanée, le papier taché, la poussière et d'autres facteurs, les résultats de la reconnaissance baissent notablement. Le coût de la transcription semi-automatique augmentant avec la taille de la base, l'utilisation d'une autre méthode de reconnaissance automatique apparaît comme une solution potentielle.

Le *Word Spotting* est une alternative relativement nouvelle pour la recherche d'information dans les images de documents anciens. Le *Word Spotting* consiste à formuler un requête sous forme d'une image de mot pour rechercher toutes les images de documents ou les passages contenant des mots similaires à la requête donnée. Des recherches ont déjà été menées dans ce domaine et différentes méthodes ont été proposées pour un *Word Spotting* efficace, mais il y a toujours une certaine marge d'erreur et la possibilité d'amélioration. En outre, le *Word Spotting* requiert le choix d'une image comme mot de requête ce qui peut poser problème pour les utilisateurs qui préféreront formuler leur requête en tapant une suite de caractères. Il est ainsi plus réaliste de proposer du *Word Spotting* à partir de requêtes ACSII. Notre principale motivation dans la recherche menée et l'objectif visé était *de proposer un système de recherche d'informations efficace qui pourrait fonctionner avec de bonnes performances en reconnaissance pour de grands volumes d'images de documents imprimés et la possibilité*

de formuler les requêtes sous forme d'image de mot ou de texte ASCII ainsi que de rechercher des mots dans des régions d'intérêt définies au préalable.

Etat de l'art

La plupart des travaux dans le domaine du *Word Spotting* sur l'écrit porte sur les textes manuscrits. La raison pour cela étant principalement que l'écriture manuscrite n'est pas reconnue par les OCRs du commerce qui sont conçus pour la reconnaissance de caractères dans des documents imprimés. Malgré le fait que les performances des OCRs soient assez bonnes pour des documents contemporains et des images de bonne qualité, lorsque les documents imprimés sont plus anciens, les polices de caractères et les images dégradées posent problème de sorte que les taux de reconnaissance des OCRs chutent. La difficulté de la reconnaissance des caractères dans des documents historiques a été souvent soulignée, par exemple B. Gatos remarque : «l'OCR est un problème très difficile à résoudre, en particulier pour des documents historiques [imprimés] ». Dans ce cas, le *Word Spotting* devient une solution de replie pour suppléer aux OCRs. Historiquement, les méthodes de *Word Spotting* ont été divisées en méthodes holistiques et méthodes analytiques. Les méthodes de reconnaissance holistique prennent une image de mot comme unité de base et ne la segmente pas en unités plus petites. Les méthodes analytiques, disposent de techniques de segmentation avec lesquelles page ou une image de mot est subdivisée en unités plus petites qui peuvent être reconnues de manière indépendante ou après regroupement de certaines. Le *Word Spotting* a surtout été appliqué à des documents écrits à la main, où la segmentation des caractères est un gros problème, ce qui explique que la plupart des méthodes présentées dans la littérature sont basées sur l'approche holistique.. Mais dernièrement des travaux utilisant des méthodes analytiques ont aussi été développées dans le but d'obtenir de meilleurs taux de reconnaissance. Ici, nous allons analyser les deux types d'approches en discutant des différentes méthodes existantes appartenant à chacune d'elles.

Les méthodes holistiques [Gatos et Pratikakis2009], [Rath et Manmatha2007], [Adamek *et al.* 2007] considèrent l'image d'un mot comme une unité qui ne sera pas davantage segmentée. Dans le domaine des documents anciens, d'autres facteurs rendent une approche holistique plus attractive, tels que leur capacité à tolérer un niveau de bruit élevé et des variations de police de caractère dans le texte, ce qui peut compliquer la segmentation en caractères. En utilisant les approches holistiques, des mots entiers peuvent être comparés directement avec des taux de reconnaissance acceptable. Adamek *et al.* [Adamek *et al.* 2007] ont présenté l'appariement des

contours de mot pour leur reconnaissance holistique dans des manuscrits historiques. Les contours fermés de mots sont extraits et mis en correspondance en utilisant une technique de contours élastiques. Gatos *et al.* dans [Gatos et Pratikakis2009] ont présenté une approche où l'image du mot requête est recherchée dans des régions d'intérêt. Pour chaque image requête, 15 cas de requête différents sont obtenus en appliquant des rotations et des variations d'échelle. Cinq ensembles de vecteurs d'attributs sont trouvés pour le mot de la requête. Les vecteurs de caractéristiques correspondants à la requête sont comparés aux vecteurs des zones rectangulaires de test en utilisant une mesure de distance. Rath et Manmatha [Rath et Manmatha2007] ont présenté une approche qui implique de grouper des images de mot dans des clusters de mots similaires, en employant l'appariement d'images de mot. Ils proposent quatre caractéristiques de profil pour les images de mot qui sont alors appariées en utilisant différentes méthodes. Leur travail a été effectué sur des documents manuscrits historiques. [Rothfeder *et al.* 2003] a employé les correspondances entre les points anguleux pour classer des images de mot par similitude dans des manuscrits historiques. Le détecteur de points anguleux de Harris est utilisé sur les images de mot. Des correspondances entre ces points sont établies en comparant des fenêtres locales et en utilisant la somme des carrés des différences. La distance euclidienne entre les points mis en correspondance donne une mesure de similarité entre mots.

Les méthodes analytiques segmentent les images de mot en unités plus petites qui peuvent être reconnues indépendamment ou quand elles sont groupées [Vamvakas *et al.* 2008], [Marti et Bunke2001], [Terasawa *et al.* 2009]. Les caractères sont les composantes de base dans les langues alphabétiques. Toutefois, on ne peut pas déterminer les points de segmentation sans reconnaître d'abord les caractères. Cette contrainte a amené les chercheurs à envisager plusieurs hypothèses de segmentation d'images en unités plus petites, telles que les suites verticales des pixels ou les composantes connexes, ou à briser un mot en unités plus petites que l'on suppose être des caractères, qui sont alors reconnues [Lu et Shridhar1996]. Bien que la segmentation en caractères soit difficile, les approches analytiques ont tendance à donner de meilleurs résultats pour la reconnaissance, ceci étant dû à leur capacité à se concentrer sur les caractéristiques locales des mots. Un autre avantage majeur de toutes les méthodes basées sur la segmentation est leur souplesse par rapport à la taille et la nature du lexique qui est dû au fait que ces méthodes conçoivent le mot comme une suite de caractères alphabétiques [Steinherz *et al.* 1999]. [Vamvakas *et al.* 2008] ont proposé une méthodologie d'OCR pour générer des fichiers ASCII pour une nouvelle image de document basée sur l'apprentissage. L'image du document est segmentée en mots, et pour chaque mot ses caractères sont extraits par une approche ascendante en utilisant l'analyse des composantes connexes. Chaque image de caractère est, en premier lieu, normalisée pour tenir dans une fenêtre de taille prédéfinie. Le caractère est ensuite représentée par

un vecteur de caractéristiques de dimension fixe qui correspondent à des propriétés évaluées dans la zone image qu'il occupe. Dans [Terasawa *et al.* 2009], les auteurs ont introduit une méthode par fenêtre glissante fondée sur *HOG* (histogramme de la pente orientée). Une fenêtre rectangulaire est appliquée sur chaque ligne de texte, elle glisse dans la direction d'écriture. Pour chaque sous-image rognée par la fenêtre, un vecteur de caractéristiques *HOG* est calculé. Les caractéristiques sont comparées en utilisant une méthode de *Dynamic Time Warping* (*DTW*).

Méthode proposée

Notre méthode est basée sur l'extraction de différentes caractéristiques multidimensionnelles pour les images de caractère avant de comparer les mots. Par opposition à [Rath et Manmatha2007] où des caractéristiques sont extraites à partir de l'image entière du mot, nous segmentons les caractères du mot. Les caractéristiques sont ensuite extraites à partir des images des caractères. De ce fait on extrait l'information du mot étudié avec plus de précision que [Rath et Manmatha2007], nous le montrerons plus tard en comparant les résultats obtenus. L'image du document est d'abord binarisée en utilisant notre algorithme NICK qui est une amélioration de la formule de Niblack originale. Le texte et les zones graphiques des images de document sont séparés et les mots du document sont extraits en appliquant la technique du *Run Length Smoothing Algorithm* (*RLSA*). Les mots correspondent aux composantes connexes de l'image obtenue après traitement par *RLSA*. Pour chaque mot détecté, les caractères qui le composent sont trouvés en revenant aux composantes connexes de l'image binarisée. Les erreurs de segmentation en caractères sont réduites en utilisant un processus de réparation en trois étapes. On obtient alors les caractères que l'on appelle les S-caractères. Sur les S-caractères, un ensemble de caractéristiques sont extraites. Toutes ces informations sont sauvegardées dans le fichier d'index. Les mots seront recherchés dans les mots de l'index en mettant en correspondance les caractéristiques des S-caractères qui les composent et celles des S-caractères du mot de la requête. La mise en correspondance des S-caractères s'effectue en utilisant des algorithmes de comparaison de séquences de type *DTW* et distance d'édition. La méthode combine des opérations de comparaison qui se font au niveau des caractères et au niveau des mots.

Les documents sont traités de manière à pouvoir disposer des éléments nécessaires pour retrouver un mot donné en phase de recherche d'information. Les différentes étapes du traitement des documents sont effectués hors ligne pour créer un fichier d'index pour

chaque image du document. Les coordonnées de chaque mot, le nombre de caractères dans le mot, la position des caractères aussi que les caractéristiques de chaque caractère, sont stockés dans les fichiers d'index. La construction des fichiers d'index permet d'accélérer le traitement lors de la sélection d'un mot requête. Le choix du mot se fait en cliquant sur le mot dans l'interface graphique de notre système de traitement des documents. La requête est traitée de la même manière au document global et les caractéristiques des S-caractères de la requête sont appariés avec les caractéristiques des S-caractères des mots déjà stockés dans le fichier d'index. Les mots pour lesquels la distance est inférieure à un seuil sont les mots acceptés. Le traitement concernant l'indexation des documents est décrit en détail dans le chapitre 3, et les techniques de réparation des mots dans le chapitre 4.

Les différentes étapes du traitement des documents sont effectués hors ligne pour créer un fichier d'index pour chaque image du document. Pour la binarisation, il n'est pas raisonnable d'utiliser un seuil global fixe de binarisation pour tous les documents. La qualité des résultats en recherche de mot dépend de la qualité de la binarisation. Aussi, nous avons modifié l'algorithme de Niblack pour le rendre plus efficace pour les documents anciens. On l'appelle NICK. Les résultats obtenus sont extrêmement satisfaisants. Par comparaison avec la formule de Niblack, nous constatons de meilleurs résultats pour des images ne présentant pas, ou très peu, d'éléments imprimés. Les images des pages sont binarisées puis traitées par *RLSA*. Les composantes connexes de ces images traitées sont les mots détectés. Nous appliquons un *RLSA* horizontal avec le seuil égal à neuf. Cette valeur est liée à la moyenne des largeurs des composantes connexes de l'image binarisée. Les mots trouvés ne correspondent pas toujours à des mots, les composantes graphiques provoquent aussi la détection de mots. Elles vont être éliminées des mots sur des critères de taille. Les composantes graphiques vérifient les conditions où leur aire est cinq fois l'aire moyenne (la moyenne des aires de toutes les composantes connexes dans cette image particulier), et leur hauteur et quatre fois la hauteur moyenne. Les résultats prouvent que cette méthode fonctionne pour presque tous les types de documents pour séparer le texte des illustrations. Un mot est un ensemble de caractères et ces caractères, dans le cas idéal, devraient être les composantes connexes. En fait une composante connexe ne correspond pas toujours à un caractère à cause de fusions, de

ruptures et des points diacritiques. Nous effectuons donc une analyse des composantes connexes extraites sur les images de mot pour améliorer la segmentation en caractères. Cette amélioration porte sur le regroupement de composantes connexes dans le cas de caractères fragmentés et de caractères comportant des marques diacritiques, ainsi que l'élimination de pseudo-caractères dans le mot. On appelle S-caractères les composantes issues de la segmentation du mot. Une méthode de réparation en trois étapes permet de réduire les erreurs de segmentation en augmentant le nombre de S-caractères correspondant à des composantes alphabétiques (les T-caractères). Dans la première étape de réparation, nous considérons la projection des S-caractères sur l'horizontale. Les composantes qui ont une partie commune dans leurs projections horizontales sont regroupées. A cette étape, les caractères accentués, les *i* et *j* sont reconstruits. Dans la deuxième étape, le recouvrement des S-caractères est considéré. Les S-caractères dont les boîtes englobantes ont une intersection non nulle sont regroupés. Cette étape concerne des caractères comme *r*, *g* etc. Dans la troisième étape, nous enlevons les signes de ponctuation (comme " , " ou " . ") qui ont été inclus dans le mot car ils ne sont pas séparés par un espace suffisamment grand. Pour cette étape, nous considérons la moyenne des aires des S-caractères du mot et supprimons tous les S-caractères qui ne remplissant pas une condition sur l'aire de la boîte englobante du S-caractère (elle doit être supérieure à 0,4 fois l'aire moyenne des S-caractères du mot pour que le S-caractère soit accepté). Les signes de ponctuation et les tout petits S-caractères constituant du bruit dans notre problématique sont marqués dans cette étape et ne sont plus considérés comme faisant parti du mot. Après ces trois étapes, nous avons extrait 98,6% des S-caractères correspondant à des T-caractères sur un ensemble des 48 images de documents de BIUM. Mais il reste toujours des problèmes de segmentation des caractères qui seront traités plus tard au niveau du mot.

Nous avons utilisé un ensemble de six séquences de caractéristiques pour représenter les images des S-caractères. Contrairement à [Rath et Manmatha2007], où il n'y a que quatre caractéristiques pour caractériser l'image du mot dans son ensemble, nous avons défini six caractéristiques pour les images de S-caractères, ce qui nous donne une meilleure représentation des mots dans un espace de caractéristiques, comme les résultats nous le révélerons plus tard. Les deux caractéristiques ajoutées permettent de mieux spécifier les formes. Ces six caractéristiques sont :

Le profil de projection verticale - la somme des valeurs d'intensité dans la direction verticale; il est calculé dans l'image du S-caractère en niveaux de gris et normalisé pour obtenir des valeurs entre 0-1.

Le profil supérieur - dans l'image binarisée du S-caractère, pour chaque colonne, nous retenons la distance entre le rectangle circonscrit au S-caractère et le premier pixel du S-caractère.

Le profil inférieur - comme dans le profil supérieur, ici nous trouvons la distance entre le dernier pixel noir du S-caractère et la boîte englobante. Les profils inférieur et supérieur sont normalisés entre 0 et 1.

L'histogramme vertical – Le nombre de pixels noirs dans une colonne de l'image binaire.

Les transitions encre/non-encre - pour capturer la structure intérieure d'un S-caractère, nous calculons le nombre de transitions de non-encre à encre dans chaque colonne de l'image.

Les transitions dans la ligne centrale - pour la ligne centrale de l'image du caractère, nous trouvons le vecteur des transitions encre/non-encre. Nous plaçons un 1 pour chaque transition et 0 pour toutes les non-transitions dans la rangée. Au départ nous avons considéré les transitions sur les trois rangées centrales et nous avons appliqué un OU logique sur celles-ci pour obtenir un vecteur transitoire moyen. Il suffisait qu'il y ait un pixel encre dans l'une des trois rangées centrales pour considérer que la colonne correspondante était un pixel encre. En comparant les deux approches, nous avons constaté que l'emploi d'une seule ligne centrale donne de meilleurs résultats et nous avons retenu cette caractéristique.

Pour chaque mot, nous obtenons les caractéristiques pour chacun de leurs S-caractères. Après le traitement des images d'un document, le fichier d'index est créé dans lequel sont stockés: la localisation des mots, la position de chacun de ses S-caractères et les caractéristiques de chaque S-caractère qui apparaissent comme une chaîne.

Pour que deux mots soient considérés éligibles pour un appariement, nous avons d'abord placé des limites sur le rapport du nombre de S-caractères correspondant à la requête au nombre de S-caractères correspondant au mot à comparer. Si le rapport est inférieur à une valeur spécifique, nous n'essayons pas d'apparier les deux mots. Si le mot est retenu pour un appariement avec le mot de la requête, une procédure d'appariement en plusieurs

étapes est lancée dans laquelle deux S-caractères sont appariés en utilisant une approche de comparaison élastique *DTW* alors que deux mots sont appariés en utilisant différents algorithmes de comparaison de chaînes. Pour mettre en correspondance les mots, nous comparons des chaînes de S-caractères qui sont eux même comparés en employant le *DTW*. Les coûts des opérations d'appariement pour le *DTW* sont des distances euclidiennes alors que les coûts des opérations d'appariement pour la comparaison de mots sont les distances entre S-caractères calculées par *DTW*. L'avantage d'employer le *DTW* pour la comparaison des caractères est qu'il est en mesure de tenir compte de l'étirement et de la compression non-linéaires des caractères. De cette manière, deux caractères, et par suite deux mots, identiques qui diffèrent par leurs tailles seront mis en correspondance correctement, à la différence de la corrélation où les mots doivent être de la même taille pour être appariés.

Pour appairer deux S-caractères, nous traitons les vecteurs de caractéristiques comme deux suites $X = (x_1 \dots x_m)$ et $Y = (y_1 \dots y_n)$. Pour déterminer la distance du *DTW* entre ces deux suites, nous calculons une matrice D d'ordre $m \times n$. Des opérations d'appariement sont définies pour pouvoir remplir la matrice D par un algorithme récursif, elles sont associées à des coûts calculés en utilisant la distance euclidienne. Une fois que toutes les valeurs de D sont calculées, le chemin d'appariement est déterminé comme le chemin de coût minimum entre la position $(1, 1)$ et la position (m, n) dans la matrice. Le coût d'appariement final est le coût $D(m, n)$ divisé par le nombre d'opérations effectué pour obtenir le chemin d'appariement. Deux S-caractères sont semblables si ce coût final est inférieur à un seuil (qui a été fixé après plusieurs expérimentations).

Pour comparer les deux mots, on a développé quatre méthodes différentes. La première est appelée « *Relative Position Correspondance (RPC)* ». Chaque S-caractère du mot requête est mis en correspondance avec un nombre différent de S-caractères voisins dans le mot inspecté. Ce nombre est fonction de la taille du mot de la requête. Pour chaque S-caractère de la requête, nous trouvons son meilleur appariement dans les S-caractères du mot inspecté et sommions les coûts de mise en correspondance des S-caractères. Après appariement des S-caractères des deux mots, nous normalisons le coût associé au mot en divisant par le nombre de S-caractères mis en correspondance. Si pour deux mots, ce coût normalisé est inférieur à un seuil, nous indiquons que les deux mots sont les mêmes. La deuxième méthode utilise le distance d'édition classique pour

comparer les deux mots. Les trois opérations d'édition (insertion, suppression et substitution) sont utilisées pour calculer la matrice d'édition W . Le coût d'appariement final pour les deux mots de longueur s et t est égal à la valeur $W(s, t)$ divisée par le nombre d'étapes du chemin d'appariement. Le problème avec ces deux méthodes est que l'on n'arrive pas à corriger toutes les erreurs de segmentation des mots en caractères, il reste des S-caractères qui ne sont pas des T-caractères.

Pour résoudre ce problème, on a introduit deux nouvelles méthodes de comparaison de mots qui vont permettre de traiter les problèmes posés par les caractères fragmentés et la fusion de plusieurs caractères. On définira donc une comparaison de séquences basée sur la distance d'édition augmentée d'opérations de fusion (*Merge-Split Edit distance*) et sa version rapide mais non optimale (*Linear Displacement Matching*). Dans *Merge-Split Edit distance*, nous avons introduit les opérations *Merge-Q* et *Merge-T* (basées sur la fusion de caractères) qui permettent de s'affranchir des problèmes dus à des segmentations imparfaites des mots en caractères. L'opération *Merge-Q* permet à un S-caractère du mot requête d'être comparé à deux S-caractères du mot courant à comparer, tandis que l'opération *Merge-T* permet à un S-caractère du mot courant d'être comparés à deux S-caractères du mot requête. En utilisant la fusion des S-caractères sur le mot courant et sur le mot requête, on peut ainsi comparer des caractères fusionnés en un seul S-caractère à plusieurs caractères sans avoir à segmenter le S-caractère pour retrouver les T-caractères, sachant que la segmentation impliquerait le choix, toujours difficile, du point de segmentation. Cette distance d'édition augmentée pour comparer les mots, couplée avec *DTW* pour comparer les S-caractères, améliore beaucoup les résultats comme le montre les évaluations présentées par la suite. L'inconvénient majeur, dans ce cas est le temps nécessaire pour la comparaison. Surtout pour les mots longs, le temps augmente considérablement à mesure que la taille de la matrice augmente avec le nombre de comparaisons entre les S-caractères. Pour surmonter ce problème de coût de calcul, une autre méthode est proposée où toute la matrice W n'a pas besoin d'être calculée et seules les comparaisons les plus pertinentes sont faites. Nous appelons cette méthode le « *Linear Displacement Matching* ». Le principal avantage que cette méthode a sur *Merge-Split Edit distance* est le temps de calcul, elle est plus rapide que le *Merge-Split Edit distance* mais cet avantage est compensé par une légère perte de performances en reconnaissance dûe à la construction partielle de la matrice des coûts de comparaison.

Résultats

Il n'existe pas de base standard de documents anciens imprimés disponible pour le développement et la validation des méthodes d'appariement de mots. Afin de comparer notre méthode à d'autres, nous les avons testées sur une base réalisée pour cette étude à partir des documents de la BIUM. Un ensemble de données des *XVIII* et *XIX*ème siècle, qui se compose de 48 images de documents, extraites de 12 livres différents et ayant un total de 17010 mots, a été utilisé pour évaluer les méthodes proposées. Un ensemble de 60 mots différents, ayant 435 instances au total, a été choisi pour constituer les mots des requêtes. Nous avons choisi ces mots dans différents contextes (titre, légende de figure, texte principal ...) avec des longueurs et des styles typographiques variés. Les résultats sont présentés dans les tableaux suivants.

Tableau 1 : Comparaisons des différentes méthodes proposées

	RPC	Distance d'édition	Merge-Split distance	Linear Matching
#query word instances	435	435	435	435
#words detected perfectly	401	406	427	420
#words missed	34	29	8	15
#relevant words detected	99	53	39	33
#false positives	51	16	4	3

Tableau 2 : Résultats sur la méthode de Rath et l'OCR commercialisé

	Rath et al. 2007	ABBYY Fine Reader
#query word instances	435	435
#words detected perfectly	335	422
#words missed	100	13
#relevant words	66	0
#false positives	54	0

Sur le tableau 1, on voit que le meilleur taux de reconnaissance a été réalisé par la méthode *Merge-Split* où nous avons correctement détecté les 427 instances du mot requête ainsi que 39 autres mots pertinents (ayant des racines communes comme : rhinoscopie et rhinoscopique). Le nombre de faux positifs dans ce cas n'est que de quatre,

ce qui montre l'efficacité et la robustesse de la méthode. Par rapport à la méthode *Merge-Split*, la méthode *Linear Displacement Matching* détecte 420 instances de mot. La méthode de Rath *et al.* a correctement identifié seulement 335 instances du mot requête et raté 100 mots. Le nombre de faux positifs est très élevé aussi, ce qui montre que la représentation globale du mot (fonction correspondant au niveau de mot) ne donne pas des résultats aussi bons que les représentations du mot par une suite de S-caractères caractéristiques au niveau du S-caractère.

Sur cette base, l'OCR obtient aussi bons résultats. Mais quand on fait l'expérimentation sur les document plus anciens du *XVIème* siècle, le performance de l'OCR diminue beaucoup. Sur un ensemble de 12 pages de trois livres différents du *XVIème* siècle, les résultats de l'OCR sont extrêmement pauvres, avec un taux de rappel de 52,04% seulement. Un taux de rappel semblable (51,46%) est observé pour la méthode de distance d'Édition. D'autre part, le *Merge-Split Edit distance* et le *Linear Displacement Matching* obtiennent des résultats très raisonnables avec des taux de rappel de 87,13% et 85,38% respectivement, et une précision de 86,63% et 88,48% respectivement. Les résultats expérimentaux confirment le fait que pour des images de documents anciens de mauvaise qualité, les méthodes de *Word Spotting* proposées permettent de retrouver des mots qui n'ont pas été reconnus par un OCR commercialisé. Les limites actuelles de nos méthodes pour des documents anciens sont causées par une segmentation en lignes de texte qui, bien que meilleure que celle produite par les OCRs du commerce, est entachée d'erreur.

Ce travail fournit un examen approfondi de plusieurs méthodes de recherche de mots dans des images de documents et la réalisation de nouveaux algorithmes de *Word Spotting* avec lesquels on obtient les meilleures performances dans des évaluations sur des documents historiques. Le système réalisé permet de visualiser les résultats des recherches d'information et de formuler des requêtes sous forme d'image de mot ou de texte ASCII.

Les bonnes performances de nos méthodes s'expliquent par le couplage d'une comparaison locale des caractères (en fait, des S-caractères) par *DTW* avec une comparaison des mots basée sur la distance d'édition. Les erreurs de segmentation des mots en caractères affectent peu les résultats dans la mesure où la segmentation est en partie corrigée avant la recherche de mot et que la méthode *Merge-Split Edit distance*,

ainsi que sa version sous-optimale rapide, sont insensibles à des erreurs de fusion ou de fission de caractères. Il faut noter que les documents historiques présentent de nombreux caractères scindés ou fusionnés. La qualité des résultats de reconnaissance obtenus par le système permet d'être assuré de son potentiel pour une utilisation dans différentes applications liées à des documents historiques.

Bibliography

- [ABBYY] ABBYY. Abby finereader, <http://finereader.abbyy.com/>.
- [Adamek *et al.* 2007] Adamek, T., O'Connor, N. E., and Smeaton, A. F. (2007). Word matching using single closed contours for indexing handwritten historical documents. *IJDAR*, 9:153 – 165.
- [Ambauen *et al.* 2003] Ambauen, R., Fischer, S., and Bunke, H. (2003). Graph edit distance with node splitting and merging and its application to diatom identification. In *Research and Advanced Technology for Digital Libraries, Lecture Notes in Computer Science 1923*, pages 259–264.
- [Andreev and Kirov2009] Andreev, A. and Kirov, N. (2009). Word image matching based on hausdorff distances. In *10th International Conference on Document Analysis and Recognition*.
- [Antonacopoulos and Karatzas2005] Antonacopoulos, A. and Karatzas, D. (2005). Semantics-based content extraction in typewritten historical documents. In *8th International Conference on Document Analysis and Recognition*, pages 48–53.
- [Antonacopoulos *et al.* 2004] Antonacopoulos, A., Karatzas, D., Krawczyk, H., and Wiszniewski, B. (2004). The lifecycle of a digital historical document: Structure and content. In *ACM Symposium on Document Engineering*, pages 147 –154.
- [Ar and Karsligil2007] Ar, I. and Karsligil, M. E. (2007). Text area detection in digital documents images using textural features. In *Computer Analysis of Images and Patterns (CAIP), Lecture Notes in Computer Science 4673*, volume 4673, pages 555–562.
- [Bai *et al.* 2009] Bai, S., Li, L., and Tan, C. L. (2009). Keyword spotting in document images through word shape coding. In *10th International Conference on Document Analysis and Recognition*.
- [Baird2004] Baird, H. S. (2004). Difficult and urgent open problems in document image analysis for libraries. In *1st International workshop on Document Image Analysis for Libraries*.
- [Bertolami *et al.* 2008] Bertolami, R., Gutmann, C., and Bunke, H. (2008). Shape code based lexicon reduction for offline handwritten word recognition. In *The Eighth IAPR Workshop on Document Analysis Systems*.

-
- [BIUM] BIUM. Digital library of bium (bibliothèque interuniversitaire de médecine, paris), <http://www.bium.univ-paris5.fr/histmed/medica.htm>.
- [Bukhari *et al.* 2008] Bukhari, S. S., Shafait, F., and Breuel, T. M. (2008). Segmentation of curled textlines using active contours. In *The Eighth IAPR Workshop on Document Analysis Systems*.
- [Burges *et al.* 1992] Burges, C., Be, J., and Nohl, C. (1992). Recognition of handwritten cursive postal words using neural networks. In *USPS Fifth Advanced Technology Conference*.
- [Casey and Nagy1982] Casey, R. and Nagy, G. (1982). Recursive segmentation and classification of composite patterns. In *Sixth Int'l Conference of Pattern Recognition*.
- [Casey and Lecolinet1996] Casey, R. G. and Lecolinet, E. (1996). A survey of methods and strategies in character segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 18:690 – 706.
- [Cesar and Shinghal1990] Cesar, M. and Shinghal, R. (1990). An algorithm for segmenting handwritten postal codes. *International Journal of Man-Machine Studies*, 33:63 – 80.
- [Chhikara and Folks1989] Chhikara, R. S. and Folks, L. (1989). *The inverse Gaussian distribution: theory, methodology, and applications*. CRC Press.
- [Christodoulakis and Brey2008] Christodoulakis, M. and Brey, G. (2008). Edit distance with single-symbol combinations and splits. In *Proceedings of the Prague Stringology Conference*.
- [Costa and Jr2001] Costa, L. D. F. and Jr, R. M. C. (2001). *Shape Analysis and Classification: Theory and Practice*. CRC Press.
- [de Waard1995]de Waard, W. P. (1995). An optimised minimal edit distance for hand-written word recognition. *Pattern Recognition Letters*, 16:1091–1096.
- [Doulgeri and Kavallieratou2009] Doulgeri, N. and Kavallieratou, E. (2009). Retrieval of historical documents by word spotting. In *16th Document Recognition and Retrieval Conference, DRR-09, USA*.
- [Duong *et al.* 2001] Duong, J., Ct, M., Emptoz, H., and Suen, C. (2001). Extraction of text areas in printed document images. In *ACM Symposium on Document Engineering ,DocEng'01*, pages 157–165, Atlanta (USA).
- [Faure and Vincent2009] Faure, C. and Vincent, N. (2009). Simultaneous detection of vertical and horizontal text lines based on perceptual organization. In *16th Document Recognition and Retrieval Conference, DRR 2009, USA*.

-
- [Feng and Tan2004] Feng, M.-L. and Tan, Y.-P. (2004). Contrast adaptive binarization of low quality document images. *IEICE Electron. Express*, 1:501–506.
- [Friedman and Kandel1999] Friedman, M. and Kandel, A. (1999). *Introduction to Pattern Recognition : Statistical, Structural, Neural and Fuzzy Logic Approaches*. World Scientific Publishing Company.
- [Gatos et al. 2009] Gatos, B., Ntirogiannis, K., and Pratikakis, I. (2009). Icdar 2009 document image binarization contest (dibco 2009). In *10th International Conference on Document Analysis and Recognition*.
- [Gatos and Pratikakis2009] Gatos, B. and Pratikakis, I. (2009). Segmentation-free word spotting in historical printed documents. In *10th International Conference on Document Analysis and Recognition*.
- [Gatos et al. 2006] Gatos, B., Pratikakis, I., and Perantonis, S. (2006). Adaptive degraded document image binarization. *Pattern Recognition*, 39:317 – 327.
- [Helmers and Bunke2003] Helmers, M. and Bunke, H. (2003). Generation and use of synthetic training data in cursive handwriting recognition. In *Pattern Recognition and Image Analysis, Lecture notes in computer science 2652*, pages 336–345.
- [Heutte et al. 1998] Heutte, L., Paquet, T., Moreau, J. V., Lecourtier, Y., and Olivier, C. (1998). A structural/statistical feature based vector for handwritten character recognition. *Pattern Recognition Letters*, 19:629–641.
- [Hoffman and McCullough1971] Hoffman, R. and McCullough, J. (1971). Segmentation methods for recognition of machine-printed characters. *IBM J. Research and Development*, pages 153–165.
- [IDP] IDP. The international dunhuang project : The silk road online - <http://idp.bl.uk/>.
- [Øivind D. Trier and Taxt1995] Øivind D. Trier and Taxt, T. (1995). Evaluation of binarization methods for document images. 17:312 – 315.
- [Jain1989] Jain, A. K. (1989). *Fundamentals of digital image processing*. Prentice Hall.
- [Jameson2004] Jameson, M. (2004). Promises and challenges of digital libraries and document image analysis: a humanist’s perspective. In *First International Workshop on Document Image Analysis for Libraries, DIAL04*.
- [Journet et al. 2005] Journet, N., Eglin, V., Ramel, J.-Y., and Mullet, R. (2005). Ancient printed documents indexation: a new approach. In *Pattern Recognition and Data Mining, Lectures Notes in Computer Science 3686*, pages 513 – 522.

-
- [Journet *et al.* 2006] Journet, N., Mullot, R., Eglin, V., and Ramel, R. J.-Y. (2006). Analyse d'images de documents anciens:catégorisation de contenus par approche texture. In *CIFED, Colloque International sur l'Ecrit et le Document*.
- [Jung *et al.* 1999] Jung, M.-C., Shin, Y.-C., and Srihari, S. N. (1999). Machine printed character segmentation method using side profiles. In *IEEE conference on Systems, Man, and Cybernetics*.
- [Kaygin and Bulut2002]Kaygin, S. and Bulut, M. M. (2002). Shape recognition using attributed string matching with polygon vertices as the primitives. *Pattern Recognition Letters*, 23:287 – 294.
- [Keogh and Pazzani2001] Keogh, E. and Pazzani, M. (2001). Derivative dynamic time warping. In *First SIAM International Conference on Data Mining, (Chicago, IL, 2001)*.
- [Khurshid *et al.* 2009a] Khurshid, K., claudie faure, and nicole vincent (2009a). Fusion of word spotting and spatial information for figure caption retrieval in historical document images. In *10th International Conference on Document Analysis and Recognition*.
- [Khurshid *et al.* 2008a] Khurshid, K., Faure, C., and Vincent, N. (2008a). Feature based word spotting in ancient printed documents. In *8th edition of PRIS in 10th Int'l conference on Enterprise Information Systems, ICEIS 2008, Spain*.
- [Khurshid *et al.* 2008b] Khurshid, K., Faure, C., and Vincent, N. (2008b). Recherche de mots dans des images de documents par appariement de caractères. In *Colloque International Francophone sur l'Ecrit et le Document, CIFED 2008, France*.
- [Khurshid *et al.* 2009b] Khurshid, K., Faure, C., and Vincent, N. (2009b). A novel approach for word spotting using merge-split edit distance. In *13th International Conference on Computer Analysis of Images and Patterns, CAIP*.
- [Khurshid *et al.* 2009c] Khurshid, K., Siddiq, I., Faure, C., and Vincent, N. (2009c). Comparison of niblack inspired binarization methods for ancient documents. In *16th Document Recognition and Retrieval Conference, DRR-09, USA*.
- [Kluzner *et al.* 2009] Kluzner, V., Tzadok, A., Shimony, Y., Walach, E., and Antonacopoulos, A. (2009). Word-based adaptive ocr for historical books. In *10th International Conference on Document Analysis and Recognition*.
- [Kolcz *et al.* 2000] Kolcz, A., Alspector, J., Augusteijn, M., Carlson, R., and Popescu, G. V. (2000). A line-oriented approach to word spotting in handwritten documents. *Pattern Analysis & Applications*, 3:153–168.

-
- [Konidaris *et al.* 2007] Konidaris, T., Gatos, B., Ntzios, K., Pratikakis, I., Theodoridis, S., and Perantonis, S. J. (2007). Keyword-guided word spotting in historical printed documents using synthetic data and user feedback. *IJDAR*, 9:167 – 177.
- [Konidaris *et al.* 2008] Konidaris, T., Gatos, B., Perantonis, S., and Kesidis, A. (2008). Keyword matching in historical machine-printed documents using synthetic data, word portions and dynamic timewarping. In *The Eighth IAPR Workshop on Document Analysis Systems*.
- [Lee *et al.* 2000] Lee, K.-H., Choy, Y.-C., and Cho, S.-B. (2000). Geometric structure analysis of document images: A knowledge-based approach. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22:1224 – 1240.
- [Leedham *et al.* 2003] Leedham, G., Yan, C., Takru, K., Tan, J. H. N., and Mian, L. (2003). Comparison of some thresholding algorithms for text/background segmentation in difficult document images. In *Seventh International Conference on Document Analysis and Recognition (ICDAR)*.
- [Levenshtein1966] Levenshtein, V. I. (1966). Binary codes capable of correcting deletions, insertions, and reversals. *Soviet Physics Doklady*, 10:707–710.
- [Lewis1995] Lewis, J. P. (1995). Fast template matching. In *Vision Interface*, pages 120 – 123.
- [Leydier2006] Leydier, Y. (2006). *Numérisation et exploration des manuscrits médiévaux*. PhD thesis, L’Institut National des Sciences appliquées, Lyon.
- [Leydier *et al.* 2005] Leydier, Y., LeBourgeois, F., and Emptoz, H. (2005). Textual indexation of ancient documents. In *Proceedings of the ACM symposium on Document engineering*, pages 111 – 117.
- [Li *et al.* 2009] Li, J., Fan, Z.-G., Wu, Y., and Le, N. (2009). Document image retrieval with local feature sequences. In *10th International Conference on Document Analysis and Recognition*.
- [Lu and Shridhar1996] Lu, Y. and Shridhar, M. (1996). Character segmentation in handwritten words — an overview. *Pattern Recognition*, pages 77–96.
- [Madhvanath and Govindaraju2001] Madhvanath, S. and Govindaraju, V. (2001). The role of holistic paradigms in handwritten word recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 23:149 – 164.
- [Manmatha *et al.* 1996a] Manmatha, R., Han, C., and Riseman, E. M. (1996a). Word spotting: A new approach to indexing handwriting. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, page 631.

-
- [Manmatha *et al.* 1996b] Manmatha, R., Han, C., Riseman, E. M., and Croft, W. B. (1996b). Indexing handwriting using word matching. In *1st ACM International Conference on Digital Libraries*.
- [Marinai *et al.* 2006] Marinai, S., Faini, S., Marino, E., and Soda, G. (2006). Efficient word retrieval by means of som clustering and pca. In *Workshop on Document Analysis Systems VII, Lecture Notes in Computer Science 3872*, pages 336–347.
- [Marinai *et al.* 2007] Marinai, S., Marino, E., and Soda, G. (2007). Exploring digital libraries with document image retrieval. In *Research and Advanced Technology for Digital Libraries, Lecture Notes in Computer Science 4675*, volume 4675, pages 368–379.
- [Marinai *et al.* 2008] Marinai, S., Marino, E., and Soda, G. (2008). A comparison of clustering methods for word image indexing. In *The Eighth IAPR International Workshop on Document Analysis Systems*.
- [Marti and Bunke2001] Marti, U. V. and Bunke, H. (2001). Using a statistical model to improve the performance of an hmm based cursive handwriting system. *International Journal of Pattern Recognition and Artificial Intelligence*.
- [Mitchell and Yan2001] Mitchell, P. E. and Yan, H. (2001). Newspaper document analysis featuring connected line segmentation. In *Sixth International Conference on Document Analysis and Recognition*, pages 1181 – 1185.
- [Moghaddam and Cheriet2009] Moghaddam, R. F. and Cheriet, M. (2009). Application of multi-level classifiers and clustering for automatic word spotting in historical document images. In *10th International Conference on Document Analysis and Recognition*.
- [NAVIDOMASS] NAVIDOMASS. <http://l3iexp.univ-lr.fr/navidomass/index.html>.
- [Niblack1986] Niblack, W. (1986). *An Introduction to Digital Image Processing*. Prentice Hall.
- [Nosary *et al.* 1999] Nosary, A., Heutte, L., Paquet, T., and Lecourtier, Y. (1999). Defining writer invariants to adapt the recognition task. In *5th edition of ICDAR*.
- [Ogier and Tombre2006] Ogier, J. M. and Tombre, K. (2006). Madonne: Document image analysis techniques for cultural heritage documents. In *Digital Cultural Heritage, Proceedings of 1st EVA Conference*, pages 107 – 114, Vienna, Austria.
- [Okun *et al.* 1999] Okun, O., Doermann, D., and Pietikainen, M. (1999). Page segmentation and zone classification: The state of the art. Technical report, University of Maryland.
- [Otsu1979] Otsu, N. (1979). A threshold selection method from grey level histogram. *IEEE Transactions on Systems, Man, and Cybernetics*, 9:62–66.

-
- [Pankow2005] Pankow, D. (2005). *The printer's manual: an illustrated history : classical and unusual texts on printing from the seventeenth, eighteenth, and nineteenth centuries*. RIT Cary Graphic Arts Press.
- [Pavladis and Zhou1991] Pavladis, T. and Zhou, J. (1991). Page segmentation by white streams. In *International conference on document analysis and retrieval*.
- [Ramel and Leriche2005] Ramel, J. and Leriche, S. (2005). Segmentation et analyse interactive de documents anciens imprimés. In *Traitement du Signal (TS)*, pages 209 – 222.
- [Randen and Husøy1994] Randen, T. and Husøy, J. H. (1994). Segmentation of text/image documents using texture approaches. In *Proc. NOBIM-Konferansen-94*, pages 60 – 67, Norway.
- [Rath2005] Rath, T. M. (2005). *Retrieval of handwritten historical document images*. PhD thesis, Graduate School of the University of Massachusetts Amherst.
- [Rath et al. 2002] Rath, T. M., Kane, S., Lehman, A., Partridge, E., and Manmatha, R. (2002). Indexing for a digital library of george washington's manuscripts: A study of word matching techniques. Technical report, University of Massachusetts Amherst.
- [Rath and Manmatha2003] Rath, T. M. and Manmatha, R. (2003). Features for word spotting in historical manuscripts. In *Seventh International Conference on Document Analysis and Recognition (ICDAR)*, page 218.
- [Rath and Manmatha2007] Rath, T. M. and Manmatha, R. (2007). Word spotting for historical documents. *IJDAR*, 9:139 – 152.
- [Reicher1969] Reicher, G. M. (1969). Perceptual recognition as a function of meaningfulness of stimulus material. *Journal of Experimental Psychology*, pages 275–280.
- [Rodriguez-Serrano and Perronnin2009] Rodriguez-Serrano, J. A. and Perronnin, F. (2009). Handwritten word-image retrieval with synthesized typed queries. In *10th International Conference on Document Analysis and Recognition*.
- [Rothfeder et al. 2003] Rothfeder, J. L., Feng, S., and Rath, T. M. (2003). Using corner features correspondences to rank word images by similarity. In *Conference on Computer vision and pattern recognition*, pages 30 – 35, USA.
- [Rui et al. 1998] Rui, Y., Huang, T. S., Ortega, M., and Mehrotra, S. (1998). Relevance feedback: A power tool for interactive content-based image retrieval. *IEEE Transactions on Circuits and Systems for Video Technology*, 8:644 – 655.

-
- [Rusinol and Lladós2008] Rusinol, M. and Lladós, J. (2008). Word and symbol spotting using spatial organization of local descriptors. In *The Eighth IAPR Workshop on Document Analysis Systems*.
- [Sankoff and Kruskal1999] Sankoff, D. and Kruskal, J. (1999). *Time Warps, String Edits, and Macromolecules: The Theory and Practice of Sequence Comparison*. CSLI Publications.
- [Sauvola and Pietikäinen2000] Sauvola, J. and Pietikäinen, M. (2000). Adaptive document image binarization. *Pattern Recognition*, 33:225–236.
- [Shi and Govindaraju2005] Shi, Z. and Govindaraju, V. (2005). Multi-scale techniques for document page segmentation. In *Eighth International Conference on Document Analysis and Recognition (ICDAR)*, pages 1020 – 1024.
- [Siddiqi and Vincent2008] Siddiqi, I. and Vincent, N. (2008). How to define local shape descriptors for writer identification and verification. In *PRIS'08: 8th Int'l workshop on Pattern Recognition in Information Systems*.
- [Steinherz *et al.* 1999] Steinherz, T., Rivlin, E., and Intrator, N. (1999). Offline cursive script word recognition - a survey. *International Journal on Document Analysis and Recognition*, pages 90–110.
- [Tan and Zhang2001] Tan, C. L. and Zhang, Z. (2001). Text block segmentation using pyramid structure. In *Proceedings of SPIE, the International Society for Optical Engineering*.
- [Tang *et al.* 1993] Tang, Y. Y., Yan, C. D., Cheriet, M., and Suen, C. Y. (1993). *Automatic analysis and understanding of documents, Handbook of pattern recognition & computer vision*. World Scientific Publishing.
- [Terasawa *et al.* 2009] Terasawa, K., Imura, H., and Tanaka, Y. (2009). Automatic evaluation framework for word spotting. In *10th International Conference on Document Analysis and Recognition*.
- [Vamvakas *et al.* 2008] Vamvakas, G., Gatos, B., Stamatopoulos, N., and Perantonis, S. J. (2008). A complete optical character recognition methodology for historical documents. In *The Eighth IAPR Workshop on Document Analysis Systems*.
- [Wagner and Fischer1974] Wagner, R. A. and Fischer, M. J. (1974). The string-to-string correction problem. *Journal of the Association for Computing Machinery*, 21:168–173.
- [Wilkinson1992] Wilkinson, R. A. (1992). Comparison of massively parallel segmenters. Technical report, National institute of standards and technology.

-
- [Wolf and Jolion2003] Wolf, C. and Jolion, J.-M. (2003). Extraction and recognition of artificial text in multimedia documents. *Pattern Analysis and Applications*, pages 309 – 326.
- [Wong *et al.* 1982] Wong, K. Y., Casey, R. G., and Wahi, F. M. (1982). Document analysis system. *IBM Journal of Research Development*, 26:647 – 656.
- [Zagoris *et al.* 2006] Zagoris, K., Papamarkos, N., and Chamzas, C. (2006). Web document image retrieval system based on word spotting. In *IEEE International Conference on Image Processing*.
- [Zhang and Tan2001] Zhang, Z. and Tan, C. L. (2001). Recovery of distorted document images from bound volumes. In *Sixth International Conference on Document Analysis and Recognition*, pages 429 – 433. IEEE.
- [Zramdini and Ingold1993] Zramdini, A. W. and Ingold, R. (1993). Optical font recognition from projection profiles. *Electronic Publishing*, 6:249–260.



Author's Publications

Khurram Khurshid, Claudie Faure, Nicole Vincent, “A novel approach for Word Spotting using Merge-Split Edit Distance”, *13th International Conference on Computer Analysis of Images and Patterns (CAIP), Lecture Notes in Computer Science*, September 2009, Germany.

Khurram Khurshid, Claudie Faure, Nicole Vincent, “Fusion of Word Spotting and Spatial Information for Figure Caption Retrieval in Historical Document Images”, *ICDAR 2009*, July 2009, Spain.

Khurram Khurshid, Imran Siddiqi, Claudie Faure, Nicole Vincent, “Comparison of Niblack inspired Binarization methods for ancient documents”, *16th Document Recognition and Retrieval Conference (DRR 2009)*, 21–22 Jan 2009, USA.

Claudie Faure, **Khurram Khurshid, Nicole Vincent**, « Détection des composantes implicitement associées dans les images de document », *EGC-2009*, Jan 2009, France.

Khurram Khurshid, Claudie Faure, Nicole Vincent, « Recherche de mots dans des images de documents par appariement de caractères », *Colloque International Francophone sur l'Ecrit et le Document (CIFED 2008)*, 28–31 Oct 2008, France.

Khurram Khurshid, Claudie Faure, Nicole Vincent, “Feature based word spotting in ancient printed documents”, *8th edition of PRIS in 10th Int'l conference on Enterprise Information Systems (ICEIS 2008)*, 12–13th June 2008, Spain.