

# Arbitrary Keyword Spotting in Handwritten Documents

Mehdi Haji

A Thesis  
In The Department  
of  
Computer Science and Software Engineering

Presented in Partial Fulfillment of the Requirements  
For the Degree of  
Doctor of Philosophy (Computer Science) at  
Concordia University  
Montréal, Québec, Canada

March 2012

© Mehdi Haji, 2012

**CONCORDIA UNIVERSITY  
SCHOOL OF GRADUATE STUDIES**

This is to certify that the thesis prepared

By: Mehdi Haji

Entitled: Arbitrary Keyword Spotting in Handwritten Documents

and submitted in partial fulfillment of the requirements for the degree of

DOCTOR OF PHILOSOPHY (Computer Science)

complies with the regulations of the University and meets the accepted standards with respect to originality and quality.

Signed by the final examining committee:

\_\_\_\_\_ Chair  
Dr. A. Hamou-Lhadj

\_\_\_\_\_ External Examiner  
Dr. R. Plamondon

\_\_\_\_\_ External to Program  
Dr. N. Kharma

\_\_\_\_\_ Examiner  
Dr. S. Bergler

\_\_\_\_\_ Examiner  
Dr. L. Kosseim

\_\_\_\_\_ Examiner  
Dr. D. Ponson

\_\_\_\_\_ Thesis Co-Supervisor  
Dr. T.D. Bui

\_\_\_\_\_ Thesis Co-Supervisor  
Dr. C.Y. Suen

Approved by \_\_\_\_\_  
Dr. V. Haarslev, Graduate Program Director

April 17, 2012

\_\_\_\_\_  
Dr. Robin A.L. Drew, Dean  
Faculty of Engineering & Computer Science

## **ABSTRACT**

### **Arbitrary Keyword Spotting in Handwritten Documents**

**Mehdi Haji, Ph.D. (Computer Science)**

**Concordia University, 2012**

Despite the existence of electronic media in today's world, a considerable amount of written communications is in paper form such as books, bank cheques, contracts, etc. There is an increasing demand for the automation of information extraction, classification, search, and retrieval of documents. The goal of this research is to develop a complete methodology for the spotting of arbitrary keywords in handwritten document images.

We propose a top-down approach to the spotting of keywords in document images. Our approach is composed of two major steps: segmentation and decision. In the former, we generate the word hypotheses. In the latter, we decide whether a generated word hypothesis is a specific keyword or not. We carry out the decision step through a two-level classification where first, we assign an input image to a keyword or non-keyword class; and then transcribe the image if it is passed as a keyword. By reducing the problem from the image domain to the text domain, we do not only address the search problem in handwritten documents, but also the classification and retrieval, without the need for the transcription of the whole document image.

The main contribution of this thesis is the development of a generalized minimum edit distance for handwritten words, and to prove that this distance is equivalent to an Ergodic

Hidden Markov Model (EHMM). To the best of our knowledge, this work is the first to present an exact 2D model for the temporal information in handwriting while satisfying practical constraints.

Some other contributions of this research include: 1) removal of page margins based on corner detection in projection profiles; 2) removal of noise patterns in handwritten images using expectation maximization and fuzzy inference systems; 3) extraction of text lines based on fast Fourier-based steerable filtering; 4) segmentation of characters based on skeletal graphs; and 5) merging of broken characters based on graph partitioning.

Our experiments with a benchmark database of handwritten English documents and a real-world collection of handwritten French documents indicate that, even without any word/document-level training, our results are comparable with two state-of-the-art word spotting systems for English and French documents.

## **Acknowledgments**

I would like to thank my supervisors Dr. Tien D. Bui and Dr. Ching Y. Suen from whom I learnt invaluable lessons during my doctoral program, both in pattern recognition and image processing classes, and in our numerous research discussions in the past five years.

I would like to thank Dr. Dominique Ponson, the Vice President, Research and Development, of IMDS Software, the industrial partner and sponsor of this research work. Thank you Dominique for believing in my work, and for all your encouraging words and brilliant ideas.

I am grateful to the MITACS and NSERC of Canada for financial support of this research through the MITACS Accelerate Award and the CRD grant.

I would like to thank my thesis committee members, Dr. Nematollaah Shiri, Dr. Leila Kosseim, Dr. Sabine Bergler, Dr. Nawwaf Kharma and Dr. Rejean Plamondon for their constructive feedback and invaluable comments on my thesis.

I am indebted to all the CENPARMI and Computer Science and Software Engineering Department staff for their constant support and assistance during my doctoral studies at Concordia University, especially Mr. Nicola Nobile, Ms. Marleah Blom, Ms. Halina Monkiewicz and Ms. Massy Joulani. I will never forget your kindness.

I would like to thank my dear friends Dr. Omid Jahromi, Dr. Ali Fall and Mr. Amin Sharifi for their invaluable insights and encouragements in the past five years.

Finally, I would like to pay tribute to my family. Thank you mom and dad. You deserve endless thanks for all you have done for me. I would also like to say a special thank you

to my brother Dr. Amir Haji whose keen theoretical insights have always helped me to broaden my ways of approaching new problems.

# Table of Content

Chapter 1 .....	1
Introduction .....	1
1.1 Problem Statement .....	1
1.1.1 Characteristics of Unconstrained Handwritten Document .....	3
1.2 Overview of Document Classification Methodology .....	4
1.2.1 Background .....	5
1.2.1.1 Pre-processing .....	7
1.2.1.2 Page Segmentation .....	9
1.2.1.3 Word Spotting .....	10
1.2.1.3.1 Proposed Approach .....	15
Chapter 2 .....	17
Margin Removal and Global Skew Correction .....	17
2.1 Introduction .....	17
2.2 Document Margin Removal and Skew Correction Using Projection Profiles ...	20
2.2.1 Margin Removal for Straight Pages .....	21
2.2.2 Margin Removal for Skewed Pages .....	24
2.3 Experimental Results .....	30
Chapter 3 .....	32
Structural Noise Removal Using Fuzzy Inference Systems .....	32
3.1 Introduction .....	32
3.2 Review of Fuzzy Logic .....	33
3.2.1 Fuzzy Sets .....	34
3.2.2 Fuzzy Operators .....	35
3.2.3 Fuzzy Rules .....	36
3.2.4 Fuzzy Inference System .....	36
3.3 Structural Noise Removal Using FIS .....	38
3.3.1 Feature Extraction .....	39
3.3.2 Specification of FIS .....	40
3.3.2.1 Rule Base for Detection of Dots and Small Noises .....	43
3.3.2.2 Rule Base for Detection of Dashes .....	45
3.4 Experimental Results .....	46
Chapter 4 .....	48
Line and Word Segmentation .....	48
4.1 Introduction .....	48
4.1.1 Background .....	49
4.2 Line Extraction Based on Fast Fourier-Based Steerable Filtering .....	52
4.2.1 Fast Fourier-based Steerable Filtering .....	52
4.2.1.1 Computation of FFT in $\phi$ -direction using Linear Interpolation .....	55
4.2.1.2 Computation of FFT in $\phi$ -direction using Nearest-Neighbor Interpolation .....	57
4.2.2 Computing Line Maps by Fast Oriented Anisotropic Gaussian Filtering ...	60
4.3 Word Segmentation Based on Fast Oriented Anisotropic Gaussian Filtering .....	63
Chapter 5 .....	67

Character Segmentation .....	67
5.1 Introduction .....	67
5.2 Character Segmentation Based on Background Skeletal Graphs .....	69
5.2.1. Terminology of the Character Segmentation Algorithm .....	69
5.2.2 Description of the Character Segmentation Algorithm .....	70
5.3 Handling Over-segmentation and Under-Segmentation .....	72
5.3.1 Character Merging Based on Graph Partitioning .....	73
5.3.1.1 Neighbourhood relation .....	75
5.3.1.2 Graph Partitioning .....	76
5.3.2 Detection of Under-segmented Pairs of Handwritten Characters Using Fuzzy Inference System .....	81
5.3.2.1 Average Number of Transition (ANT) Features .....	83
5.3.2.2 Fuzzy Inference System (FIS) .....	85
5.3.2.3 Experimental Results .....	88
Chapter 6 .....	90
Cursive Character Recognition .....	90
6.1 Introduction .....	90
6.2 Artificial Neural Network for Handwritten Character Recognition .....	95
6.2.1. Feature Extraction .....	95
6.2.2. Training .....	96
6.3 Perturbation Method for Character Recognition .....	97
6.3.1 Transformation Operators .....	98
6.3.2 Combination of Classifiers .....	99
6.3.2.1 Modified Borda Count .....	100
6.4 Experimental Results .....	102
Chapter 7 .....	104
Generalized Minimum Edit Distance for Handwritten Words .....	104
7.1 Introduction .....	104
7.2 Classical Minimum Edit Distance .....	105
7.3 Generalized Minimum Edit Distance .....	106
7.3.1 Default Cost Functions .....	108
7.4 Modeling Generalized Minimum Edit Distance Using HMMs .....	109
7.4.1 Hidden Markov Models .....	110
7.4.1.1 Three Fundamental Problems for HMMs .....	111
7.4.1.2 Topologies of HMMs .....	112
7.4.2 Modeling Generalized Minimum Edit Distance Using GEHMMs .....	115
7.4.2.1 Initial and Transition Probabilities .....	116
7.4.2.2 Observation Probabilities .....	117
7.4.2.3 Decoding: Recognition of Handwritten Words Using the GEHMM Model .....	121
7.4.3 Incorporating A Priori Knowledge to GEHMMs for Handwritten Word Recognition .....	121
7.4.3.1 Adding Knowledge about the Lexicon .....	122
Chapter 8 .....	126
Experimental Results, Future Work and Conclusion .....	126
8.1 Outline of Keyword Spotting System and Design of Experiments .....	126



8.1.1 Separation of Keywords from Non-keywords .....	127
8.2 Training Data .....	133
8.3 Test Data .....	136
8.4 Experiments .....	138
8.4.1 Word-Level Experiments.....	138
8.4.1.1. Word Recognition Experiments.....	139
8.4.1.2. Word Spotting Experiments.....	145
8.4.1.2.1 Closed-Lexicon Word Spotting Experiments .....	149
8.4.1.2.2 Open-Lexicon Word Spotting Experiments.....	152
8.4.2 Document-Level Experiments .....	152
8.5 Future Work .....	157
8.6 Conclusion .....	158
List of Publications .....	161
Appendix.....	162
A1. Rule Base for Detection of Dots and Small Noises .....	162
A2. Rule Base for Detection of Dashes .....	165
References.....	170

## List of Figures

Figure 1.1 Samples of unconstrained handwritten documents with simple to moderate layouts. ....	2
Figure 1.2 Samples of unconstrained handwritten documents with complex layouts. ....	4
Figure 1.3 High-level block diagram of document classification methodology. ....	5
Figure 2.1 Examples of documents images with margins. ....	19
Figure 2.2 A document image with margin and the corresponding vertical and horizontal projection profiles. ....	20
Figure 2.3 A document image with margin and the corresponding AKC2 curves and the result of margin removal algorithm. ....	23
Figure 2.4 A skewed document page with margin and the corresponding vertical and horizontal projection profiles. ....	25
Figure 2.5 Trapezoids corresponding to vertical and horizontal projection profiles of a skewed document page with margin. ....	25
Figure 2.6 An axis-aligned rectangle tilted to the left and to the right by the same angle. ....	26
Figure 3.1 Examples of membership functions defined on variable orientation. ....	34
Figure 3.2 Fuzzy sets defined on variables Normalized Y-COG and Aspect Ratio. ....	40
Figure 3.3 Fuzzy sets defined on variables normalized height and normalized width. ....	42
Figure 3.4 Fuzzy sets defined on variables compactness and orientation. ....	43
Figure 3.5 Result of applying the FIS-based noise removal filter to a handwritten word. ....	45
Figure 3.6 Samples of handwritten text with guideline noise. ....	47
Figure 4.1 Linear interpolation for the computation of the FFT in a certain direction. Here the size of the filter is 7, and the orientation is $30^\circ$ . Green pixels correspond to the coordinates that are rounded down to the closest column index, and blue pixels correspond to the coordinates that are rounded up to the closest column index. ....	56
Figure 4.2 Down-sampling corresponding to nearest-neighbor interpolation when the line angle is less than $45^\circ$ (a), and when it is more than $45^\circ$ (b). ....	58
Figure 4.3 Some diagonals of an image corresponding to $\varphi = 30^\circ$ and $D_f = 2$ . ....	59
Figure 4.4 Line segmentation algorithm based on FFS filtering. ....	62
Figure 4.5 Result of applying the line extraction algorithm using FFS filters to a handwritten document with multiple skewed lines. (a) input document image. (b) image (a) after margin removal and global skew correction. (c) Line map obtained by applying FFS filters to the image and adding the outputs together. (d) image (c) after binarization. (e) image (d) after post-processing. ....	62
Figure 4.6 Robust line fitting versus least square line fitting in presence of outliers. ....	63

Figure 4.7 Example of a handwritten line where the space between characters of the same word is wider than the space between two neighbouring words. ....	64
Figure 4.8 Output of line and word segmentation algorithms for a handwritten French document. ....	65
Figure 4.9 Output of line and word segmentation algorithms for a handwritten English document. ....	66
Figure 5.1 Samples of handwritten words with a lot of overlapping between characters. 68	
Figure 5.2 Results of applying main steps of character segmentation algorithm to a handwritten word. ....	71
Figure 5.3 Explicit character segmentation algorithm using background skeletal graph. 72	
Figure 5.4 Samples of handwritten words that can have more than one transcription. ....	74
Figure 5.5 Example of a neighbourhood graph corresponding to a handwritten word. ...	76
Figure 5.6 Merging algorithm for sequence of connected components based on graph partitioning. ....	81
Figure 5.7 Samples of handwritten pairs of characters without deep enough skeletal branches on segmentation paths. ....	82
Figure 5.8 Three basic membership functions for the definition of fuzzy sets. ....	85
Figure 5.9 Samples from database of handwritten characters for evaluation of under-segmented detection method. ....	88
Figure 5.10 Samples of under-segmented pairs of handwritten characters that are correctly classified by under-segmented detection method. ....	89
Figure 5.11 Samples of handwritten characters that are misclassified by under-segmented detection method. ....	89
Figure 6.1 Fuzziness in handwriting. Examples of letters from NIST SD19 database which may be confused with each other. ....	91
Figure 6.2 Samples of handwritten words from the IAM database with letters that are difficult to recognize correctly in isolation. ....	92
Figure 6.3 Block diagram of perturbation-based classification versus standard classification. ....	97
Figure 6.4 Block diagram of general perturbation-based classification. ....	99
Figure 7.1 Examples of HMMs with (a and b) and without (c) topological constraints. 113	
Figure 7.2 GEHMM corresponding to the generalized minimum edit distance defined by Equation (7.2) ....	116
Figure 7.3 159-state enhanced GEHMM model for word recognition. ....	124
Figure 7.4 Decomposition of the character substitution state based on the character trigram model. ....	125
Figure 8.1 Two major modules of keyword spotting system and levels of experiments. 126	
Figure 8.2 Two major approaches to recognition-based keyword spotting. ....	128
Figure 8.3 Examples of local minima/maxima contour points of handwritten words. ...	130
Figure 8.4 Samples of synthesized images for two words ‘adhesion’ and ‘resiliation’. 134	
Figure 8.5 Samples of handwritten words from the IAM database. ....	135
Figure 8.6 Samples of handwritten forms from the IAM database. ....	136

Figure 8.7 Samples of handwritten mails from our French documents database. ....	137
Figure 8.8 Average keyword spotting performance in terms of precision-recall curves.	155
Figure 8.9 Sample output of proposed word spotting system for a handwritten document from the database of incoming mails. ....	156

# Chapter 1

## *Introduction*

### **1.1 Problem Statement**

Despite the existence of electronic media in today's world, a considerable amount of written communications are in paper form such as books, advertisements, bank cheques, contracts, etc. The quantity of paper documents that must be processed by human is growing every day. There is an increasing demand for document image processing and understanding such as automation of information extraction, classification, search, and retrieval of documents.

The goal of our research is to develop a complete methodology for automatic retrieval/classification of collections of images of unconstrained documents based on the presence of one or several keywords which can be specified by the user. Keyword spotting is the core problem in search/retrieval /classification applications, and as such it has attracted considerable interest by academia and industry in recent years. It should be mentioned that for clean printed documents the problem could be considered solved at least in theory. However, the difficulty lies in dealing with documents that are noisy and contain unstructured and handwritten material [MGB09]. We will propose a keyword spotting algorithm that enables the user to automatically retrieve/classify the documents

that contain, for example, a person/company's name or any other arbitrary word from a collection of scanned handwritten documents.

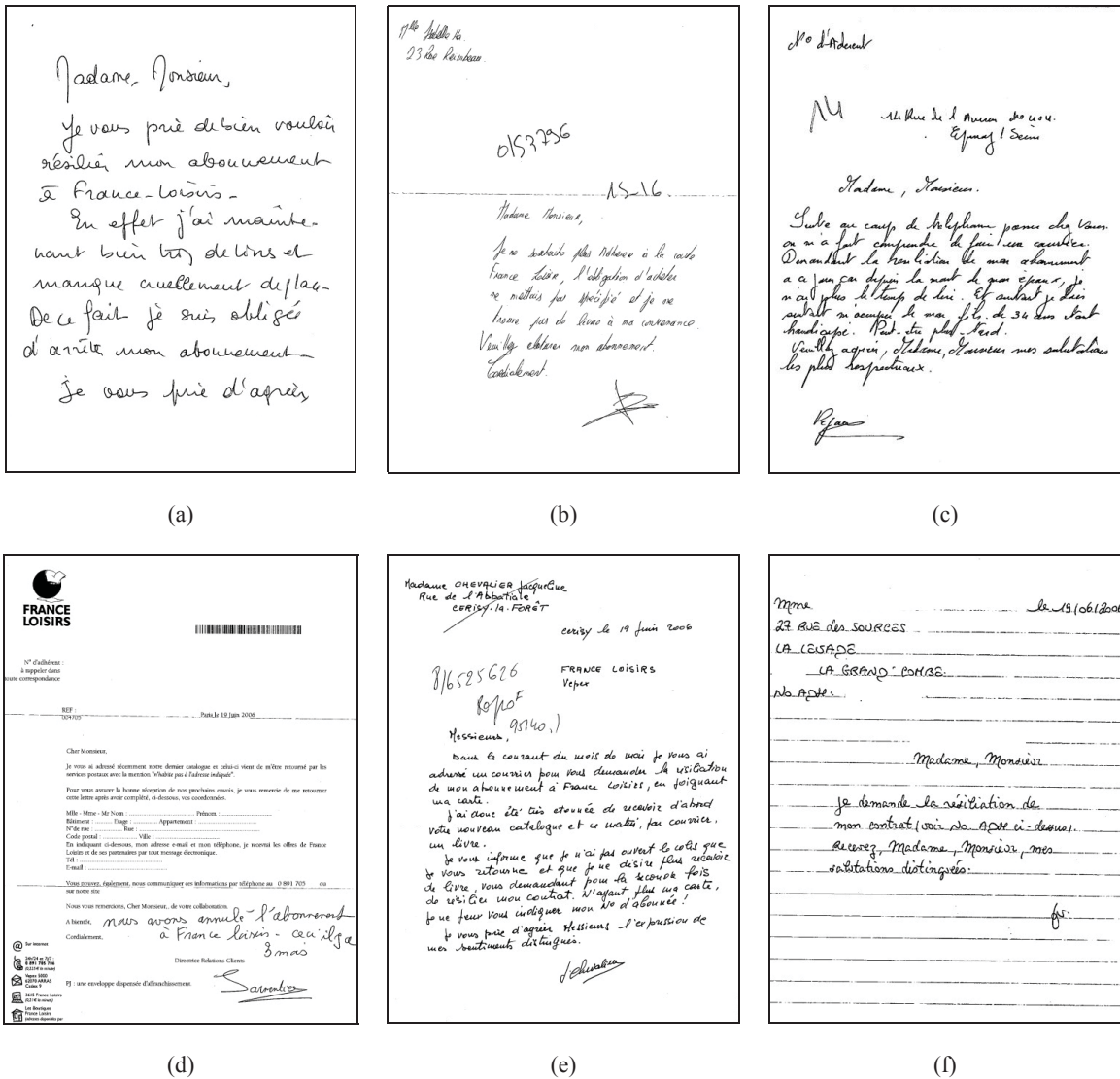


Figure 1.1 Samples of unconstrained handwritten documents with simple to moderate layouts.

To the best of our knowledge, to date there is no system capable of classifying unconstrained document images based on arbitrary text keywords with an acceptable performance for real-world applications. Recently there has been some success in limited domains such as mail processing or searching in handwritten medical forms [MGB09].

However in such applications, the structure of the document is fixed and the lexicon is usually limited and small. The emphasis of our research is to search in unconstrained documents with arbitrary keywords.

### **1.1.1 Characteristics of Unconstrained Handwritten Document**

In general, an unconstrained handwritten document can have the following characteristics:

a) the text is often densely written and the text items are not well-separated. Adjacent lines may be connected and the distance between neighbouring words may be less than the gap between characters of the same word; b) aside from text, usually there are other types of items present in the document such as underlines, signatures, barcodes, graphics, logos etc. c) the document may contain a combination of handwritten and machine-printed materials which need different types of processing; d) the text lines may not be always straight and they do not always have a single global skew angle; e) different text areas may have different font sizes; f) the text items may be connected to each other or to non-text items by noise, scratches, tables, rule/margin lines or background textures; g) the document may have non-uniform illumination. This is especially true for aged and historical documents; h) Characters may be broken due to noise, poor contrast, non-uniform ink, and/or scanning artefacts; i) The words may be slanted especially in handwriting (i.e. vertical strokes of the text may deviate from the vertical direction), and the slant is not uniform across the text and/or for the same word. Figure 1.1 shows samples of unconstrained documents with simple to complex layouts, and Figure 1.2 shows samples with complex layouts.

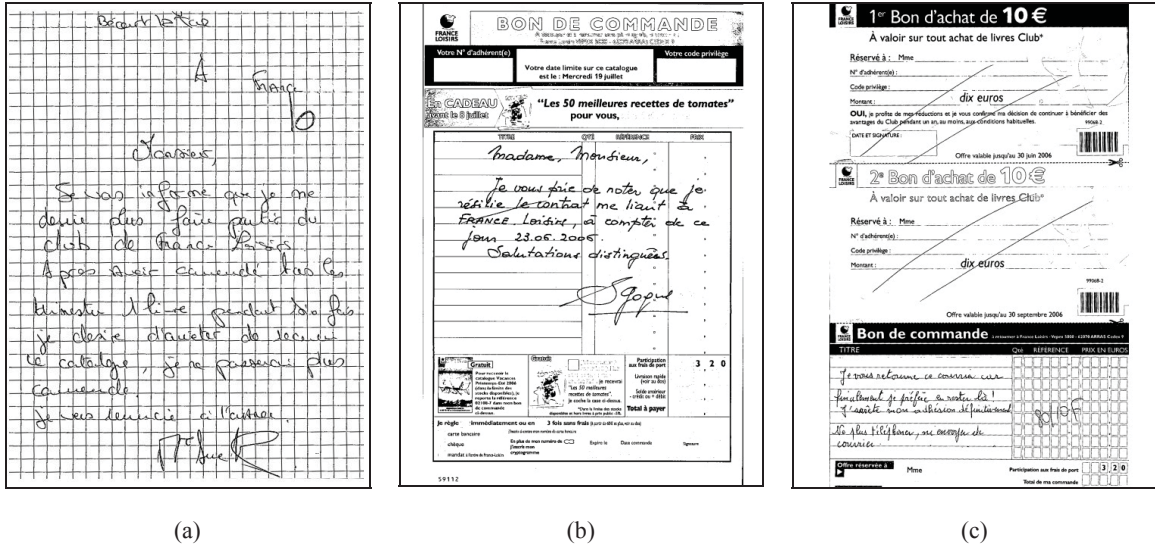


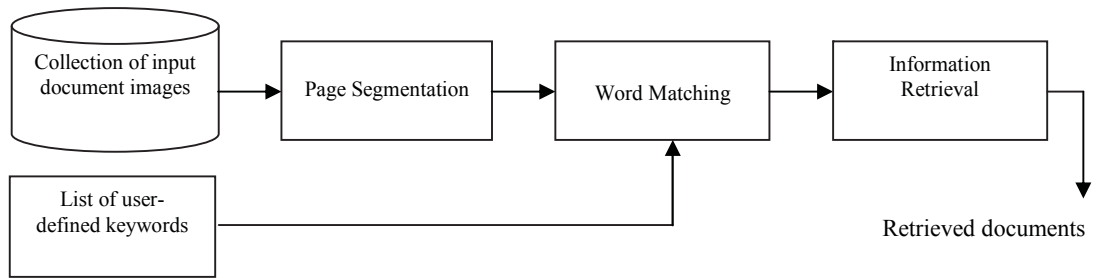
Figure 1.2 Samples of unconstrained handwritten documents with complex layouts.

## 1.2 Overview of Document Classification Methodology

The three major components of a document image classification system are document segmentation, word matching, and information retrieval (Figure 1.3).

Document segmentation algorithms are concerned with dividing a document image into its constituent parts. In general, a document may be composed of various types of items other than text such as margins, graphics, logos, barcodes, signatures, tables etc. In our application, we process the textual contents of the documents. Therefore, it is logical to view the document segmentation module as being composed of a pre-processing step which is responsible for removing any items but text, followed by line extraction and word segmentation steps. After generating a set of word candidates, we must compare each one with the set of template keywords and if there is a match, we will spot the location of the word in the document image, and perhaps assign the whole document image to a certain category based on the posterior probabilities of the detected keywords.





**Figure 1.3 High-level block diagram of document classification methodology.**

In the following, we will explore in greater depth the three major modules of document classification systems, namely page segmentation, word matching, and information retrieval. We will also present a literature survey of each module and we will describe the proposed research and methodology that we will use to tackle the challenges therein.

### **1.2.1 Background**

Handwritten document retrieval strategies can be broadly divided into two categories: template-based and recognition-based. The recognition-based approaches can be further divided into algorithms based on OCR correction and algorithms based on modified information retrieval models [GCB09]. Template-based approaches aim at solving the retrieval problem by comparing the image data with a set of template images corresponding to the keywords. On the other hand, recognition-based approaches aim at solving the retrieval problem by partially or fully transcribing the document image and then doing the retrieval in the text domain. Any approach has its advantages and

disadvantages. Generally speaking, template-based approaches are more suited for document images with complex layouts. However, there are two major downsides. Firstly, the matching process is slow. Secondly and more importantly we need to have a set of representative sample images for any keyword that may be searched for, and this restricts the area of applicability of the system. On the other hand, the main advantage of recognition-based approaches is that they obviate the need for the collection of training samples for keywords. However, the disadvantage of recognition-based approaches is that they require the segmentation of the document image into its constituent lines, words and characters which involve challenging problems as we will see in subsequent chapters.

In our application, we are interested in the retrieval of documents that may contain any arbitrary keywords, such as a person/company's name. Therefore, our proposed methodology will be based on recognition-based approaches. There are two main approaches to word recognition: holistic [vdZSH08, LRM04] and non-holistic (a.k.a. analytic) [EYSSG99, KSS05]. In the former, which is more straightforward, a database of training samples is needed for each word (keyword). Therefore, we are faced with the same problem as in the template-based approaches, that it is not always possible to compile a large enough training database for all possible keywords. Consequently, the main idea behind our proposed methodology is to use non-holistic methods. We extract the text lines, segmented them into their constituent words and then letters. We dynamically build models for keywords based upon trained models of handwritten characters. Therefore, in our proposed general keyword spotting methodology, the word matching consists of character segmentation and character recognition.

### ***1.2.1.1 Pre-processing***

The pre-processing step is composed of two procedures: margin removal and noise removal. We will begin with margin removal. Document images which are obtained from scanners or photocopiers usually have a black margin which interferes with subsequent stages of document layout analysis and page segmentation algorithms. Therefore, it is necessary to remove these margins before any subsequent stages in a document processing application.

There are a few works which have addressed the problem of document margin removal. Manmatha and Rothfeder in [MR05] have proposed a novel method using scale spaces for segmenting words in handwritten documents, where in a pre-processing stage, they have used the basic technique of projection profiles for the detection of document margins. It is quite straightforward to find the margins from the projection profiles when the document is not tilted and the page is a perfectly straight rectangle. But this is not always the case. The page may be skewed, and it may not be a perfect rectangle, meaning the corners may not be right angles and even the page sides may not be perfect straight. cuts. Also, any of the four margins may be present or not. The basic technique discussed in [MR05], is not able to handle these cases. A more advanced algorithm is presented in [FWL02], which is based on a top-down approach. The image is first split by finding possible boundaries between connected blocks, and then the regions corresponding to marginal noise are identified and removed based upon shape length and location of the split blocks. This algorithm is able to remove marginal noise from skewed pages, but it cannot correct the page skew. Moreover, it does not find the page borders, i.e. it just removes the marginal

noise and if portions of a neighboring page are present in the image, they will be left untouched.

In our application, we need to find the page borders and correct the skew we eventually want to display the location of the detected keywords on the original document image. We have devised an algorithm for document margin removal based on the detection of document corners from projection profiles [HBS09]. This algorithm does not make any restrictive assumptions regarding the input document image to be processed. It neither needs all four margins to be present nor requires the corners to be right angles. In the case of the tilted documents, it is able to detect and correct the skew. We will discuss margin removal in more detail in Chapter 2.

Noise removal is the first step of any image processing and computer vision system, and we have given particular attention to it because the presence of noise complicates all the subsequent steps of a document processing system. We present an effective method based on fuzzy inference systems for removal of structural noise from document images. Structural noise is a type of noise that is not an artifact but rather a part of the data that is undesirable, for example when we want to recognize a handwritten word in a text line, the comma that separates the word from the following word is considered as structural noise. Structural noise is application-dependent and usually defined by some linguistic rules and qualitative terms which are imprecise in nature. Therefore, we utilized fuzzy logic which is a tool for handling imprecision and qualitative knowledge. We will talk about structural noise removal using fuzzy inference systems in Chapter 3.

### **1.2.1.2 Page Segmentation**

After removal noise, we have to detect and separate the text lines. Due to intrinsic challenges of unconstrained documents, this problem has remained unresolved and thus, many different approaches to segmenting lines and words have been proposed so far [LGPH09]. These segmentation approaches usually make two key assumptions: 1) the gap between neighbouring text lines is significant; and 2) the text lines are reasonably straight, or else they have a single global skew angle. These assumptions are not always valid for unconstrained handwritten documents. According to our experiments, the method of steerable directional local profile [SSG09] produces the best results for our database of unconstrained document images. However, this method is based on the Adaptive Local Connectivity Map (ALCM) filtering [SSG05] which is computed by convolution in time-domain, and consequently it is slow. In order to overcome the slowness problem, we computed the connectivity map by using anisotropic Gaussian filters [ASW03].

Once the text lines are extracted, we have to segment words on the same text line. The difficulty here is rooted in the fact that in handwritten documents, inter-word-spacing is sometimes wider than the intra-word-spacing and thus it is not always possible to segment the document at the word level perfectly using geometrical information only. Many different approaches to segmenting words are proposed so far. We may categorize word segmentation algorithms to either top-down, bottom-up or hybrid ones. According to our experiments, the scale-space algorithm [MR05] gives promising for unconstrained handwritten documents. As in the line extraction, we compute the scale space by using anisotropic Gaussian filters. Therefore, we devise a unified approach to line extraction and

word segmentation based upon anisotropic Gaussian filters. We will explore these algorithms in greater depth in Chapter 4.

### **1.2.1.3 Word Spotting**

Having generated a set of candidates for image segments that represent words, the next task is to decide whether or not each one corresponds to a given text keyword. This is the core problem in document classification and retrieval. The detection of keywords fulfills two purposes in documents. It determines by means of a computer program whether or not a scanned document image contains a text keyword and optionally, spots the instances of the keyword in the document image. Many different approaches to word recognition and word spotting have been proposed so far.

A handwritten word spotting method based on biologically inspired features is proposed in [vdZSH08]. The authors use a holistic recognition approach based on simple  $k$ -Nearest Neighbor ( $k$ -NN) classifiers. The theory behind this word recognition model is based on a model of the visual cortex proposed by Serre et al. [SWB+07]. This model follows the organization of visual cortex in primates which is hierarchical, aiming to build invariance to position and scale first and then to viewpoint and other transformations. The advantage of the biologically inspired features is that they alleviate the need for large number of training samples for each word. According to the experiments reported in [vdZSH08], an accuracy rate of around 53% can be achieved for a lexicon of size 2099 for the words with 10 or less training samples. However, in order to achieve a higher accuracy, more training samples are needed. The accuracy of the system rises to 89% for the words with 50 or more training samples. To collect this amount of data, especially in real-time applications,

is cumbersome and not practical. As the authors point out, the intrinsic disadvantage of this method, as any holistic classification is the large number of classes that need to be trained. Moreover, there is no reuse of shape knowledge such as is the case in concatenated character-based HMMs. The authors believe that a considerable improvement in recognition performance is expected by applying character or syllable-based HMM models in conjunction with their locally invariant features.

Another holistic approach to word spotting is proposed in [KAA+00]. This system uses a line-oriented search strategy where each document image is considered as a sequence of text lines, each of which is represented by an ordered sequence of columns. Using this approach, the problem of segmenting the text into individual words is avoided. This system uses a template-based matching, meaning that there is no training. However, the problem of collecting template models still remains, that is we have to have enough number of representative samples for each class. The template matching is based on profile features and Dynamic Time Warping (DTW). Given that enough number of training samples is available, this system achieves a moderate performance of 40% at low false positive rates.

A template-free spotting keyword spotting system is proposed in [FFB10]. This system is derived from a novel unconstrained handwritten word recognition engine which is based on Bidirectional Long Short-Term Memory (BLSTM) neural networks [GLF+09]. The BLSTM neural networks are a type of recurrent neural networks specifically designed for sequence labeling tasks where it is difficult to find the boundaries between the constituent parts of the data and there are long-range contextual dependencies between the data. Using the BLSTM neural networks it is possible to have access to past and future context, which

is not available using ordinary left-to-right HMMs. According to the experiments carried out on large lexica containing 20,000 to 30,000 words, this neural-network-based recognition system achieves a recognition rate of around 74%, significantly outperforming some state-of-the-art HMM-based recognition systems. The word spotting system based on BLSTM neural networks uses a token passing algorithm in order to compute the matching score between a sequence of letter probabilities and sequence of text characters. This word spotting system achieves an average precision rate of 82.8% and a high precision rate of 95% at 50% recall on a lexicon of size 4,000 words.

A segmentation-free word spotting method is proposed in [LLE07]. Segmentation free methods are particularly useful for processing historical manuscripts where the document is degraded or has a complex layout, and the text is densely written. In this method the authors have proposed differential features that are compared using a cohesive elastic matching method. This matching method is based on Zones of Interest (ZOIs) in order to match only the informative parts of the words. Feature selection based on ZOIs overcomes the incompetence of correlation-based methods when directly applied on the grey levels. Aside from providing a better matching capability for handwriting, a main advantage of this method is less computation time compared to naïve matching methods. Furthermore, there is no need to gather a training database. However, the user has to provide one image of the word which is going to be spotted, and the image must be selected from the same set of documents in order for the matching algorithm to achieve a good performance. In this regard, this word-spotting system resembles a Content-Based Image Retrieval (CBIR) system. The authors have tested their system on two small databases of Latin and Semitic



manuscripts. The system achieves a precision and recall rate of around 60-80% depending on the level noise and degradation in the document and variations in the writing styles.

A probabilistic method for keyword retrieval in handwritten document images is proposed in [CBG09]. This method addresses the problem of imperfect word segmentation by modeling segmentation errors as probabilities and integrating these probabilities into the word spotting algorithm. The word segmentation probabilities are obtained by modeling the conditional distribution of distance features of word gaps. The word recognition probabilities are obtained from the distances returned by a lexicon-driven word recognition engine [KG97]. Then, the segmentation and the recognition probabilities are combined in a probabilistic model of word spotting. The lexicon-driven word recognition engine segments a word image into character hypotheses by finding all possible locations of the ligatures connecting any two characters. This is done using a contour-based analysis. Then, the distance between the word image and an entry in the lexicon is computed by enumerating all possible segmentations of the word image and finding the one that has the overall minimum distance. Finally, the distances from the recognition engine are converted into probabilities using the Universal Background Model (UBM) [RQD00]. The authors have tested their method on database of medical forms. Automatic processing of medical forms is a challenging task due to the poor image quality and wide range of keywords used (in order of 50,000 words). By comparing their method to two state-of-the-art word spotting systems based on template matching, the authors concluded that an improvement of 2.5% to 3.8% (in terms of mean average precision) is obtained by using word segmentation probabilities in the similarity measurement. However, the word spotting

performance on this challenging database is still very low. The system achieves a precision rate of less than 15% at 10% recall, and a mean average precision of 4.7%.

A retrieval system for machine-printed documents is proposed in [ZEP10]. This method falls in the category of template-based techniques that use word images as queries. However, there is no need for collecting training samples because for machine-printed documents, a query image can be automatically generated from the input query by using a font that has characteristics similar to the estimated characteristics of the characters in the document image. The matching is performed at the word-level using a set of shape features; consequently there is no need for recognition of individual characters. The distances between images is simply computed in the feature space using the Minkowski distance of order 1 (a.k.a.  $L_1$  distance). The authors have tested their system on a collection of 100 document images. These document images are artificially created by rendering various texts as images and then adding different amounts of noise to them. The retrieval system achieves a mean precision rate of 87.8% at a 99.26% recall. Although the artificially generated collection of documents does not reflect the challenges of real-world documents, it can still show the effectiveness of the proposed system. The authors have utilized a state-of-the-art OCR software in order to transcribe the document images so as to do the retrieval in the text domain. Compared to the spotting-based retrieval system, the performance of the transcription-based retrieval system is considerably lower with a precision rate of 76.7% at a 58.4% recall.

Another OCR-free retrieval method for machine-printed documents is proposed in [MDES09]. Based on extensive experiments carried out on two real-world databases of English and Arabic documents, the authors show the capability of their matching-based

technique in language-independent document retrieval. However, the performance of the system is lower compared to corresponding transcription-based retrieval systems. The matching-based retrieval system achieves a mean average precision of 23% on the Arabic documents, compared to 38% achieved by the transcription-based retrieval system. This result is contrary to the result of [ZEP10] that suggests matching-based retrieval is more effective than OCR-based retrieval. Therefore in this sense, we can say that there is “no free lunch” in document retrieval. Whether OCR-based retrieval is better than OCR-free retrieval depends on the set of documents, features, classification methods etc.

#### **1.2.1.3.1 Proposed Approach**

Our approach to spotting arbitrary keywords in handwritten documents is based on a generalized minimum edit distance. This distance computes the cost of the conversion of a sequence of images (of characters) to any arbitrary sequence of characters. Therefore, we developed a character segmentation algorithm for cursive text, which is based on the analysis of background skeletal graphs. The main function of the algorithm is to obtain the branches that correspond to possible segmentation paths from the graph corresponding to the skeleton of the word background. Over-segmentation and under-segmentation errors are two inherent sources of performance degradation in any character segmentation algorithm. In order to handle these problems, we developed a merging algorithm for over-segmented and broken characters, and a fuzzy inference system for detection of under-segmented pairs of handwritten characters. We will present the character segmentation algorithm in greater depth in Chapter 5.

Having segmented a word image into its constituent characters (or sub-characters), we need to decide whether or not the word image represent a keyword. For this purpose, we developed a generalized minimum edit distance. Furthermore, we proved that this distance is equivalent to an Ergodic Hidden Markov Model (EHMM) therefore we were able to optimize the free parameters of the distance using the well-established HMM training algorithms. This generalized distance enables us to assign the whole document image to a certain category, or sort all the document images in the collection in order of relevance to the input keywords. The main contribution of our approach is that it provides an exact model for the temporal information present in the handwriting with a feasible number of states. To the best of our knowledge, this is the first work to present an exact 2D model for handwritten words while satisfying practical constraints.

We will discuss our cursive character recognition approach in Chapter 6, followed by the generalized edit distance in Chapter 7. Finally, we will present our experimental results in Chapter 8.

## Chapter 2

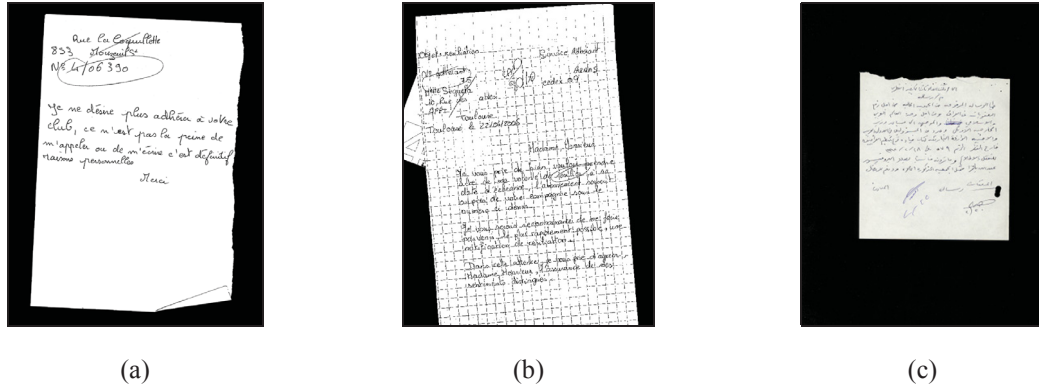
### *Margin Removal and Global Skew Correction*

#### **2.1 Introduction**

Document images obtained from scanners or photocopiers usually have a black margin which interferes with subsequent stages of page segmentation algorithms. Thus, the margins must be removed at the initial stage of a document processing application. This chapter presents an algorithm which we have developed for document margin removal based upon the detection of document corners from projection profiles. The algorithm does not make any restrictive assumptions regarding the input document image to be processed. It neither needs all four margins to be present nor needs the corners to be right angles. In the case of the tilted documents, it is able to detect and correct the skew. In our experiments, the algorithm was successfully applied to all document images in our databases of French document images which contain more than six hundred images with different types of layouts, noise, and intensity levels.

Document processing technologies are concerned with the use of computers for automatic processing of different kinds of media containing text data. Examples of the applications are Optical Character Recognition (OCR), digital searchable libraries, document image retrieval, postal address recognition, bank cheque processing and so on. In most of these applications, the source of data is an image of a document coming from a

scanner or a photocopier. During the process of scanning or photocopying, an artifact, which we simply refer to as margin, is added to the image. These black margins are not only a useless piece of data and unpleasant when the page is reproduced (reprinted), but also can interfere with the subsequent stages of document layout analysis and page segmentation algorithms. Therefore, it is desirable or necessary to remove these margins before any subsequent stages in a document processing application. Despite its practical significance, this problem is often overlooked or not discussed thoroughly in papers. There are only a few studies which have addressed the problem of document margin removal. Manmatha and Rothfeder in [MR05] have proposed a novel method using scale spaces for segmenting words in handwritten documents wherein they have used the basic technique of projection profiles for the detection of document margins. It is easy to obtain the margins from the projection profiles when the document is not tilted and the page is a perfectly straight rectangle. But as shown in Fig. 2.1, this is not always the case. The page may be skewed, and it may not be a perfect rectangle. Also, any of the four margins may be present or not. The basic technique discussed in [MR05], is not able to handle these cases. Peerawit and Kawtrakul in [PK04] have proposed a marginal noise removal method based upon edge detection. They have used the edge density property of the noise and text areas to detect the border between them. This method is designed to remove left and right margins only, and is incapable of handling skewed pages.



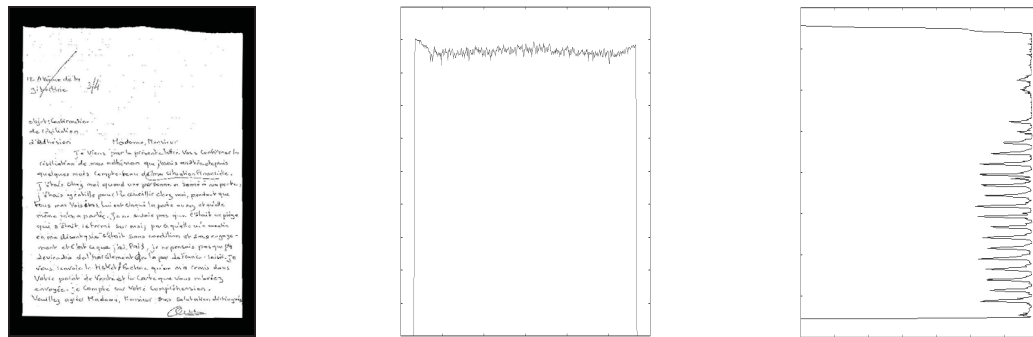
**Figure 2.1** Examples of documents images with margins.

In [FWL02], Fan et al. have proposed a top-down approach to margin removal. Firstly, the image is divided by locating possible boundaries between connected blocks. Next, the regions corresponding to marginal noise are identified by applying some heuristics based upon shape length and location of the split blocks and finally these regions are removed. Fan et al.'s algorithm is able to remove marginal noise from skewed pages, but it cannot correct the page skew. Moreover, it does not find the page borders, i.e. it only removes the marginal noise and if portions of a neighboring page are present in the image, they will not be removed.

In [SvBKB07, SvBKB08], Shafait et al. have used a geometric matching algorithm to find the optimal page frame. Their method is based on extraction and labeling of connected components at the first stage. Text lines and text zones must be identified prior to margin detection. However, extracting text lines from a page is a challenging task, especially for unconstrained handwritten types of documents [DPB09, LZD+08b]. In fact, Shafait et al.'s algorithm is designed for machine printed documents. Moreover, it assumes the page frame is an axis-aligned rectangle (i.e. again, it cannot handle skewed pages).

In [NSK07], Stamatopoulos et al. have proposed a border detection algorithm for camera document images. Their method is based upon projection profiles combined with a connected component labeling process. But again, it needs the document skew to be corrected prior to margin removal.

There are several other published works concerning the problem of margin removal [LTW96, CLLT02, ZT01], but to the best of our knowledge, the algorithm we present here is the first to address the problem of margin removal in presence of document skew.



(a) Input Document Image      (b) Vertical Projection Profile      (c) Horizontal Projection Profile

**Figure 2.2 A document image with margin and the corresponding vertical and horizontal projection profiles.**

## 2.2 Document Margin Removal and Skew Correction Using Projection Profiles

In this section we explain the margin removal algorithm, starting with the case of straight pages and then generalizing it to handle skewed pages.



### 2.2.1 Margin Removal for Straight Pages

The basic function of the algorithm is to find the corners which correspond to the page margins from the projection profiles of the input image. For a straight page, the left-most and right-most sharp corners in the horizontal profile of the image correspond to the left and right margins, and the left-most and right-most sharp corners in the vertical profile of the image correspond to the upper and lower margins (Fig. 2.2). Carrying out this task may appear simple, however the difficulty of implementation lies in corner detection, which is one of the most studied and open problems in computer vision. But in our case, by searching for the corners in 1-D projection profiles, rather than a 2-D image, we encounter a problem which can be easily solved.

Much research has been conducted upon the subject of corner detection in computer vision literature. This research can be broadly classified into two categories: grey-level and boundary-based [SLYT07]. In the first category, corners are found by using corner templates or computing the gradient at edge pixels. In the second category, corners are found by analyzing the properties of boundary pixels. For our case, we have chosen a boundary-based approach because we want to obtain corners from 1-D profiles which correspond to the document boundaries. We use a modification of the  $K$ -Cosine measure presented in [SLYT07] which is a new and robust algorithm for position, orientation, and scale invariant boundary-based corner detection for 2-D images.

The  $K$ -Cosine measure for a set of boundary points  $S = \{ P_i \mid i = 1, 2, \dots, m \}$  is defined for each point  $i$  as follows:

$$c_i(K) = \cos \theta_i = \frac{\vec{a}_i(K) \cdot \vec{b}_i(K)}{\|\vec{a}_i(K)\| \cdot \|\vec{b}_i(K)\|} \quad (2.1)$$

Where  $\bar{a}_i(K) = \bar{P}_{i+K} - \bar{P}_i$  and  $\bar{b}_i(K) = \bar{P}_{i-K} - \bar{P}_i$  are the two vectors connecting the point  $i$  to the  $K^{\text{th}}$  point before and after it, and  $\theta_i$  denotes the angle between these two vectors. Therefore,  $K$ -cosine provides a measure of the curvature of boundary points over a region of support specified by  $K$ .

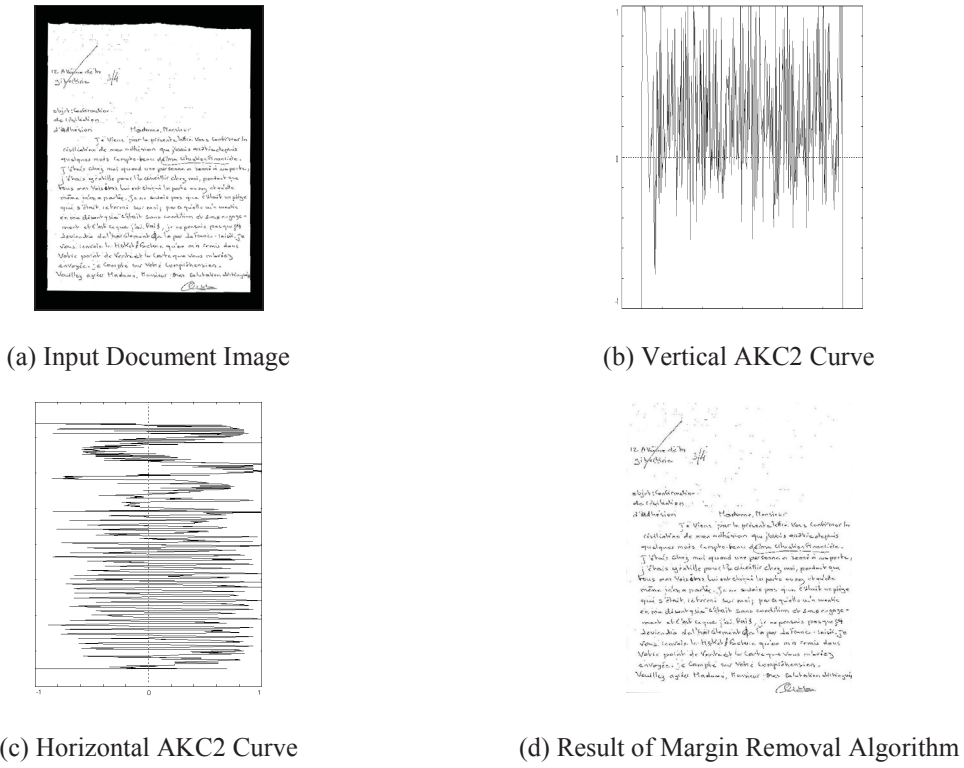
The overall performance of the 2-D corner detection algorithm based on the  $K$ -Cosine measure greatly depends on  $K$ . In [SLYT07], a careful analysis and a method of choosing a proper value for  $K$  is discussed, which is based on some geometric properties of the input set of boundary points. But, in our simplified 1-D version of the problem, where we are looking for corners in 1-D profiles, even a fixed value of  $K$  will work fine. Because, firstly, the corners of interest are almost right angles, secondly, they are located near the left and right ends of the boundary (i.e. projection profile), thirdly, there is only zero or one corners at each end (depending on whether or not the margin is present).

As the value of  $K$  is fixed in our application, we modify the definition of the  $K$ -Cosine measure in order to make sure that the corner detection scheme is robust against profile noise. We simply use a low-pass filtering which can be implemented as an averaging operation. More precisely, for each point of a projection profile, now we take the average of the  $K$ -Cosine measure over a local neighborhood of  $K$ . This new curvature measure is defined as follows:

$$C_i(K) = \frac{1}{K} \sum_{k=K/2}^{3K/2} c_i(k) \quad (2.2)$$

Having defined the curvature measure, we apply it to all points of the projection profile to obtain the corresponding Averaged  $K$ -Cosine Curvature Curve (AKC2). Now, the first

zero-crossings of AKC2, scanning from left to right and right to left, correspond to the left and right corners of the projection profile. This is due to the fact that  $K$ -Cosine values vary between  $-1 = \cos(\pi)$  and  $1 = \cos(0)$ , and thus the AKC2 curve has to cross the axis at the left and right rising edges of the corresponding profile. Please note that, even if the projection profile is not an exact rectangle function (i.e. it does not have 90-degree corners), the AKC2 curve still has two zero crossings which correspond to the left and right (or top and bottom) margins. Fig. 2.3 shows the document image of Fig. 2.2 with the corresponding AKC2 curves which determine the four margins of the image and the final result of margin removal.



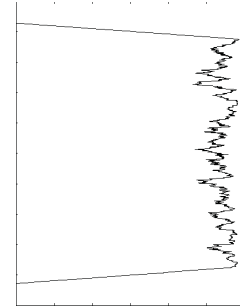
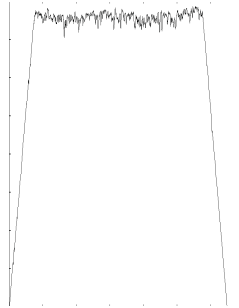
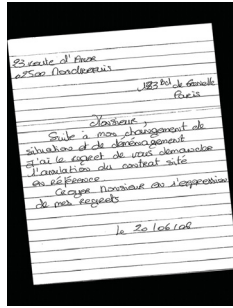
**Figure 2.3 A document image with margin and the corresponding AKC2 curves and the result of margin removal algorithm.**

### 2.2.2 Margin Removal for Skewed Pages

For skewed pages we observe that horizontal and vertical projection profiles have an isosceles trapezoidal shape as shown in Fig. 2.4. In this case, we need to estimate the base angle of the corresponding trapezoid to be able to correct the page skew. Let  $T_{vpp}(I)$  and  $T_{hpp}(I)$  denote the trapezoids corresponding to the vertical and horizontal projection profiles of the input document image  $I$  respectively. The base angle of  $T_{vpp}$  which is the angle that the two non-parallel sides of it make with vertical axis, or equivalently, the base angle of  $T_{hpp}$  which is the angle that the two non-parallel sides of it make with horizontal axis is equal to the page skew angle.

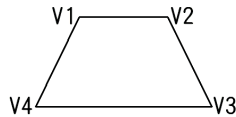
In order to estimate the base angle, we use the same technique discussed in the previous section for finding corners in projection profiles. However in this case, we need all the four corners (i.e. the four vertices of the corresponding trapezoid).

Let  $V_1, V_2, V_3$  and  $V_4$  denote the four vertices of  $T_{vpp}$ , and  $H_1, H_2, H_3$  and  $H_4$  denote the four vertices of  $T_{hpp}$  as shown in Fig. 2.5.  $V_2$  and  $V_3$ , and  $H_2$  and  $H_3$  can be found from the corresponding AKC2 curves, exactly the same way we did in the previous section. However, for  $H_1$  and  $H_4$ , and  $V_1$  and  $V_4$ , it should be noted that these corners may be very close to, or exactly lie on, the two ends (boundaries) of the corresponding profiles. Therefore, the AKC2 may not provide an appropriate measure of curvature to find them. We can easily handle this boundary problem by padding the profiles with enough ( $> K$ ) number of zeros, corresponding to fictitious black margins on the four sides of the input document image.

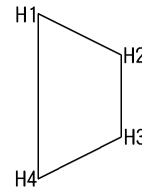


(a) Input Document Image (b) Vertical Projection Profile (c) Horizontal Proj. Profile

**Figure 2.4 A skewed document page with margin and the corresponding vertical and horizontal projection profiles.**



(a)  $T_{vpp}$



(b)  $T_{hpp}$

**Figure 2.5 Trapezoids corresponding to vertical and horizontal projection profiles of a skewed document page with margin.**

Having obtained the coordinates of the vertices of  $T_{hpp}$  and  $T_{vpp}$ , we can calculate the absolute value of the page skew angle, but not the sign of it. As shown in Fig. 2.6, an axis-aligned rectangle when tilted to the left and to the right by the same skew angle  $\theta$ , result in the same horizontal and the same vertical projection profiles. The proof is trivial by noting that the areas of the triangles  $L_1L_2T'_1$  and  $R_1R_2S'_2$ ;  $L_3L_4T'_3$  and  $R_3R_4S'_4$ ;  $L_1L_4T'_4$  and  $R_1R_4S'_1$ ; and  $L_2L_3T'_2$  and  $R_2R_3S'_3$  are equal by symmetry; and so are the areas of the parallelograms  $T'_4L_2T'_2L_4$  and  $R_1S'_3R_3S'_1$ ; and  $L_1T'_1L_3T'_3$  and  $S'_2R_2S'_4R_4$ . Where  $T'_1$  is the

intersection of the line segments  $L_1T_1$  and  $L_2L_3$ ;  $T'_2$  is the intersection of the line segments  $L_2T_2$  and  $L_3L_4$ ;  $T'_3$  is the intersection of the line segments  $L_3T_3$  and  $L_1L_4$ ; and  $T'_4$  is the intersection of the line segments  $L_4T_4$  and  $L_1L_2$ ; and similarly for  $S'_1, S'_2, S'_3$  and  $S'_4$ .



**Figure 2.6 An axis-aligned rectangle tilted to the left and to the right by the same angle.**

In Fig. 2.6, the inner rectangle  $P_1P_2P_3P_4$  can correspond to the bounding box of a page of document without skew and margin. Then, the rectangles  $L_1L_2L_3L_4$  and  $R_1R_2R_3R_4$  are the skewed versions of it, and the triangles  $I_1L_1L_2, I_2L_2L_3, I_3L_3L_4, I_4L_4L_1, I_1R_1R_4, I_2R_2R_1, I_3R_3R_2$  and  $I_4R_4R_3$  correspond to the black (dark) margins around the page.

In our problem, given the horizontal and vertical projection profiles, we want to find the page corners (i.e. the coordinates of the rectangle  $P_1P_2P_3P_4$ ). We do this by first obtaining the coordinates of  $L_1L_2L_3L_4$  and  $R_1R_2R_3R_4$  and then determining the sign of the skew angle. Let  $V_{1x}, V_{2x}, V_{3x}$  and  $V_{4x}$  be the indices of the four columns of the image corresponding to the four corners of the vertical projection profile as shown in Fig. 2.5(a). Let  $H_{1y}, H_{2y}, H_{3y}$  and  $H_{4y}$  be the indices of the four rows of the image corresponding to the four corners of the horizontal projection profile as shown in Fig. 2.5(b). Now, when  $L_1L_2L_3L_4$  and  $R_1R_2R_3R_4$  correspond to the left-skewed and right-skewed versions of the image, from Fig. 2.6, we can easily see:

$$\begin{aligned}
L_{1x} &= R_{4x} = V_{4x} \\
L_{4x} &= R_{1x} = V_{1x} \\
L_{2x} &= R_{3x} = V_{2x} \\
L_{3x} &= R_{2x} = V_{3x} \\
L_{2y} &= R_{1y} = H_{1y} \\
L_{1y} &= R_{2y} = H_{2y} \\
L_{3y} &= R_{4y} = H_{3y} \\
L_{4y} &= R_{3y} = H_{4y}
\end{aligned} \tag{2.3}$$

Or,

$$\begin{aligned}
L_1 &= (V_{4x}, H_{2y}) \\
L_2 &= (V_{2x}, H_{1y}) \\
L_3 &= (V_{3x}, H_{3y}) \\
L_4 &= (V_{1x}, H_{4y}) \\
R_1 &= (V_{1x}, H_{1y}) \\
R_2 &= (V_{3x}, H_{2y}) \\
R_3 &= (V_{2x}, H_{4y}) \\
R_4 &= (V_{4x}, H_{3y})
\end{aligned} \tag{2.4}$$

Therefore we have obtained the coordinates of the skewed versions of the page from the projection profiles of it. Now, it is straightforward to calculate the absolute value of the skew angle  $\theta$ . From Fig. 2.6, obviously we can obtain the absolute value of  $\theta$ , by computing the slope of any of the eight sides of the rectangles  $L_1L_2L_3L_4$  and  $R_1R_2R_3R_4$ . But as we pointed out earlier, the projection profiles are noisy and the page may not be a

perfect rectangle; consequently, the coordinates of the skewed rectangles that we obtain from the above set of equations are estimates and not exact. Therefore, we make use of all the eight sides of the two rectangles to obtain the Maximum Likelihood (ML) estimate for the absolute value of  $\theta$ :

$$|\theta| = \frac{1}{8} \left\{ \left| \tan^{-1} \left( \frac{L_{2y} - L_{1y}}{L_{2x} - L_{1x}} \right) \right| + \left| \tan^{-1} \left( \frac{L_{3x} - L_{2x}}{L_{3y} - L_{2y}} \right) \right| + \left| \tan^{-1} \left( \frac{L_{4y} - L_{3y}}{L_{4x} - L_{3x}} \right) \right| + \left| \tan^{-1} \left( \frac{L_{1x} - L_{4x}}{L_{1y} - L_{4y}} \right) \right| + \left| \tan^{-1} \left( \frac{R_{2y} - R_{1y}}{R_{2x} - R_{1x}} \right) \right| + \left| \tan^{-1} \left( \frac{R_{3x} - R_{2x}}{R_{3y} - R_{2y}} \right) \right| + \left| \tan^{-1} \left( \frac{R_{4y} - R_{3y}}{R_{4x} - R_{3x}} \right) \right| + \left| \tan^{-1} \left( \frac{R_{1x} - R_{4x}}{R_{1y} - R_{4y}} \right) \right| \right\} \quad (2.5)$$

As we mentioned earlier, from the projection profiles we cannot determine the sign of the skew angle. Therefore, we need another source of information to resolve the ambiguity of whether  $L_1L_2L_3L_4$  or  $R_1R_2R_3R_4$  corresponds to the true bounding box of the page. We use the fact that the local deviation of image pixels along the two sides (left and right, or up and down) of any of the four borders of the page is “high”, and any border of the page corresponds to one side of the true bounding box. More precisely, the deviation of image pixels along the two sides of a line segment belonging to the true bounding box is “higher” than the other candidate line segment belonging to the other bounding box. Let  $ALD_w(I, L)$  be the Average Local Deviation function which maps an area of the image  $I$  specified by the line segment  $L$  and thickness  $w$  to an integer in  $[0, 255]$ , assuming the input image is an 8-bit grayscale one. The output of the function is the average of the absolute differences of the sum of  $w$  image pixels on the left and right, or top and bottom, along the line segment. If the line slope is greater than 1, meaning the line segment is more vertical than horizontal, we look at the left and right side of it for computing the local deviation. Otherwise, the line slope is less than 1, meaning the line segment is more



horizontal than vertical, we look at the top and bottom of it for computing the local deviation. Let  $\{ L_i \mid i = 1, 2, \dots, n \}$  be the set of coordinates of the image pixels corresponding to the line segment  $L$ . We obtain these coordinates by using the Bresenham's line algorithm [Bre65]. Now, the function  $ALD_w(I, L)$  can be formally defined as follows:

$$ALD_w(I, L) = \frac{1}{w.n} \left( H \left( \frac{|L_{ny} - L_{1y}|}{|L_{nx} - L_{1x}|} - 1 \right) \left| \sum_{i=1}^n \sum_{t=1}^w I(L_{iy}, L_{ix} - t) - \sum_{i=1}^n \sum_{t=1}^w I(L_{iy}, L_{ix} + t) \right| + H \left( \frac{|L_{nx} - L_{1x}|}{|L_{ny} - L_{1y}|} - 1 \right) \left| \sum_{i=1}^n \sum_{t=1}^w I(L_{iy} - t, L_{ix}) - \sum_{i=1}^n \sum_{t=1}^w I(L_{iy} + t, L_{ix}) \right| \right) \quad (2.6)$$

Where  $H(x)$  is the Heaviside step function.

Having defined the ALD function, we can check which of the rectangles  $L_1L_2L_3L_4$  or  $R_1R_2R_3R_4$  corresponds to the true bounding box of the page. If  $L_1L_2L_3L_4$  is the true bounding box, then  $ALD_w(I, L_1L_2)$  is higher than  $ALD_w(I, R_1R_2)$ , and vice versa. The same proposition holds true for the other three pairs of sides:  $L_2L_3$  and  $R_2R_3$ ,  $L_3L_4$  and  $R_3R_4$ , and  $L_4L_1$  and  $R_4R_1$ . As we do not assume the document page must have perfectly straight borders (look at the top border of the document page of Fig. 2.1(c) for example), we use all the four propositions to calculate the sign of the skew angle by taking a simple majority vote. We never encountered a case of a draw in our experiments. But if it happens, for example when the ALD function for two sides of  $L_1L_2L_3L_4$  is higher than the two corresponding sides of  $R_1R_2R_3R_4$  and is lower for the other two sides, it is either because 1) the skew angle is too small, and so we do not need to correct the skew at all,

or 2) the page borders are very jagged, in which case we can try a larger value for  $w$ , for example we can multiply it by 2, and then calculate the ALD propositions again.

Having obtained the absolute value of the skew angle and the sign of it, we can correct the page skew by rotating the image by  $-\theta$  around the center of the page which is the intersection of the diagonals of the bounding box.

The coordinates of the bounding box after skew correction,  $P_1$ ,  $P_2$ ,  $P_3$  and  $P_4$  (according to the naming convention of Fig. 2.6), determine the page margins. We again use the ML estimates:

$$\text{left margin} = (P_{1x} + P_{4x})/2 \quad (2.7)$$

$$\text{right margin} = (P_{2x} + P_{3x})/2 \quad (2.8)$$

$$\text{top margin} = (P_{1y} + P_{2y})/2 \quad (2.9)$$

$$\text{bottom margin} = (P_{3y} + P_{4y})/2 \quad (2.10)$$

### 2.3 Experimental Results

We tested our proposed algorithm on a database containing 156 French document images with different types of margin noise, layouts and background/foreground intensity levels. As only a small percentage of the documents were skewed (21 documents in total), we added some artificially generated skewed document images to the database by randomly selecting a set of the real documents and rotating each one by a random angle within  $-\pi/6$  to  $\pi/6$ . There were 57 of these artificially skewed samples so we obtained an equal

number of straight and skewed document images. With  $K = 30$  and  $w = 10$  fixed throughout all the experiments, our proposed algorithm successfully estimated the skew angle (with a standard deviation of less than 0.25 degrees) and removed margins in all cases. It should be mentioned that the algorithm performance is not very sensitive to the values of  $K$  and  $w$ . We expect the algorithm to have the same performance for a wide range of values for these two parameters.

In summary, in this chapter, we proposed a document margin removal algorithm based on corner detection in projection profiles. The algorithm does not need the input page to be a perfect and axis-aligned rectangle; meaning that it can handle skewed, non-right-angled corners, or jagged page borders which are the cases that we may encounter in the processing of real-world documents.

## Chapter 3

### *Structural Noise Removal Using Fuzzy Inference Systems*

#### **3.1 Introduction**

There are two types of noise that we have to handle when working with handwritten documents: statistical and structural. Low-level noise is a statistical artifact that is introduced by the involved equipment, for example during the scanning process. Structural noise is not an artifact but rather a part of the data that is undesirable, for example when we want to recognize a handwritten word in a text line, the comma that separates the word from the following word is considered as structural noise. There are a lot of different approaches to reducing (or removing) low-level statistical noise from images [CB05, ZJ10]. However, structural noise removal depends on the specific application, and obviously the inherent constraints and settings of each problem may call for different treatments. What is structural noise and needs to be removed is usually defined by some linguistic rules and qualitative terms which are imprecise in nature. For example, if we want to remove the separator dots (‘.’) from a text line but keep the dots that belong to the characters (‘i’ and ‘j’), we decide based on a rule that uses a piece of knowledge that a separator dot should appear *near* the baseline.

Fuzzy logic is a form of logic derived from fuzzy set theory to deal with variables and reasoning that are approximate. Fuzzy inference systems (FISs) which are rule-based

systems based on fuzzy variables have been successfully applied to many fields such as expert systems, data classification, decision making, computer vision and automatic control [JS97, OC02]. One main advantage of fuzzy variables and fuzzy rules is that they facilitate the expression of rules and facts that are easily understandable for humans. Furthermore, it is easy to modify a fuzzy inference system by inserting and deleting rules, meaning that there is no need to create a new system from scratch. In order to train a fuzzy inference system, it is possible to start with a few rules that are designed by human expert and then fine-tune the parameters of the FIS over a set of training (validation) data.

In this chapter, we will present the process of designing a FIS for removal of structural noise from images. We will start by building an FIS which can distinguish small noises from character dots and then will show how to extend the system for other types of structural noise such as background line noise versus dashes etc. Finally, we will show the effectiveness of the rule-based noise removal system by some experimental results carried out on real-world images.

## **3.2 Review of Fuzzy Logic**

In this section we present a brief review of fuzzy logic. However, we encourage the reader to refer to a textbook on the subject [JS97, Neg04] for further information. Fuzzy logic is an extension of classical (binary) logic that uses a continuous range of truth degrees in the real interval  $[0, 1]$ , rather than the strict values of 0 and 1. In order to introduce fuzzy logic, first we define the concept of fuzzy sets.

### 3.2.1 Fuzzy Sets

A fuzzy set is a set whose elements have degrees of membership in the real interval  $[0, 1]$ . In classical set theory, an element either belongs to a set or not. The membership of an element  $x$  in a set  $A$ , in classical logic, is defined by an indicator function (a.k.a. characteristic function). The value of the indicator function is 1 when  $x \in A$ , and 0 when  $x \notin A$ . In fuzzy logic, the degree of membership of an element in a set is indicated by a value in the real interval  $[0, 1]$ . This extension allows the gradual assessment of the membership of elements in a set.

The function that defines the degree of membership of an element  $x$  in a set  $A$  is  $m_A(x)$ , and therefore we denote the fuzzy set by the pair  $(A, m_A(x))$ , or  $A(x)$  for short.

**Example.** Let  $x$  be the orientation (in degrees) of a 2D shape  $S$ . We can define the fuzzy sets HORIZONTAL and VERTICAL on  $x$  by the triangular membership functions as given in Figure 3.1.



(a) Membership function for the set horizontal      (b) Membership function for the set vertical

**Figure 3.1** Examples of membership functions defined on variable orientation.

When the orientation  $x$  is  $0^\circ$  or  $180^\circ$ , it is fully included in the fuzzy set HORIZONTAL, and it is not included in the set VERTICAL. When  $x$  is  $90^\circ$ , it is fully included in the set

VERTICAL, and not included in the set HORIZONTAL. For these three values ( $0^\circ$ ,  $90^\circ$ ,  $180^\circ$ ), the memberships can be defined by the classical notion of set as well. However, when  $x$  is  $22.5^\circ$  for example, then its degree of membership to the set HORIZONTAL is 0.5, which can be interpreted as *somewhat* horizontal in linguistic terms.

### 3.2.2 Fuzzy Operators

The basic operations defined on crisp sets, namely intersection (AND), union (OR) and complement (NOT), can be generalized to fuzzy sets. The generalization to fuzzy sets can be achieved in more than one possible way. The most widely used fuzzy set operations that we will use in this work are called standard operations. The three standard fuzzy operations are standard fuzzy intersection, standard fuzzy union, and standard fuzzy complement.

Let  $A(x)$  and  $B(x)$  denote two fuzzy sets, that is the degree to which  $x$  belongs to  $A$  is  $m_A(x)$ , and the degree to which  $x$  belongs to  $B$  is  $m_B(x)$ .

The standard fuzzy complement for set  $A(x)$  denoted by  $cA(x)$  is defined as  $1 - m_A(x)$ .

The standard fuzzy intersection for two set  $A(x)$  and  $B(x)$  denoted by  $(A \cap B)(x)$  is defined as  $\min[ m_A(x), m_B(x) ]$ .

The standard fuzzy union for two set  $A(x)$  and  $B(x)$  denoted by  $(A \cup B)(x)$  is defined as  $\max[ m_A(x), m_B(x) ]$ .

### 3.2.3 Fuzzy Rules

In fuzzy logic, we represent logic rules by a collection of IF-THEN statements. Each statement has the general form of IF  $P$  THEN  $Q$ , where the antecedent  $P$  is a single or compound fuzzy assignment statement, and so is  $Q$ . A single assignment statement has the general form of “ $x$  is  $A_i$ ” and a compound assignment statement is constructed from single assignments and set operations for example “*orientation* is HORIZONTAL AND *height* is HIGH”. As can be seen, fuzzy rules facilitate the representation of linguistic rules. In order to make the representation of such rules even easier, we use fuzzy hedges, which are equivalent of the adverbs in natural languages. The most common types of fuzzy hedges are “very” and “somewhat” which are defined as follows. Let  $(A, m_A(x))$  denote a fuzzy set defined on the universe of discourse  $x$ , then:

$$\text{very } A(x) \equiv (\text{very } A, m_{\text{very } A}(x)) \text{ where } m_{\text{very } A}(x) = [m_A(x)]^2$$

$$\text{somewhat } A(x) \equiv (\text{somewhat } A, m_{\text{somewhat } A}(x)) \text{ where } m_{\text{somewhat } A}(x) = [m_A(x)]^{1/2}$$

Of course there is more than one possible way to define these hedges. The purpose of “very” is to concentrate the membership function, the purpose of “somewhat” is to dilate the membership function.

### 3.2.4 Fuzzy Inference System

The process of definition of the mapping from a given set of inputs to a set of outputs using fuzzy logic is called fuzzy inference. The relation between the set of inputs and



outputs is defined by fuzzy IF-THEN rules as explained in the previous section. The set of fuzzy rules combined with a method of fuzzy inference is called Fuzzy Inference System (FIS). There are two major types of FIS systems: Mamdani-type and Sugeno-type, of which the former one is the most commonly used and the one that we use in this work.

Mamdani's inference method is based on "MIN-MAX" operations, therefore sometimes Mamdani's inference method is referred to as MIN-MAX inference.

The first step in Mamdani's inference is to compute the degree of membership of each input variable  $x_i$  to all fuzzy sets that are defined on it. This step is called input fuzzification. Next, we compute the *truth degree* or the *value of antecedent* of each rule in the rule base. When  $P$  is a single assignment (i.e. *orientation* is HORIZONTAL), the value of antecedent is simply the value of the corresponding membership function. When  $P$  is a compound assignment statement (i.e. *orientation* is HORIZONTAL AND *height* is SHORT), the value of antecedent is obtained by applying the MIN (for AND) and MAX (for OR) operators to the truth degrees of each part of  $P$ . For example, if the truth degree (i.e. membership value) of "*orientation* is HORIZONTAL" is 0.7, and the truth degree of "*height* is SHORT" is 0.9, then the antecedent value of the rule "IF *orientation* is HORIZONTAL AND *height* is SHORT THEN ..." is  $\min(0.7, 0.9) = 0.7$ .

After obtaining the value of antecedent, we compute the consequent membership function for each rule. This process is called fuzzy implication. In Mamdani's inference, the implication operator is MIN. The MIN operator limits the membership function of the consequent to the value of antecedent. Formally, let  $P$  be the antecedent,  $v_{P(x)}$  be the value of antecedent, and  $Q(x) \equiv (Q, m_Q(x))$  be the consequent. Then, the membership function of the consequent  $Q$  is defined to be  $\min(v_{P(x)}, m_Q(x))$ .

The next step is to aggregate the conclusions, that are the membership functions of the consequents of all rules in the rule base. In Mamdani's inference, the aggregation operator is MAX, which is the standard fuzzy union operator. When we have more than one rule defining the relation between the input variables and an output variable, in fuzzy logic, all rules are fired (with different degrees of strength) and hence they collaborate to define the value of the output. As the rules are independent, and they are all equally important, the combination of them is defined as the union. As we mentioned in the previous section, in standard fuzzy, union can be obtained by the MAX operator.

The last step in Mamdani's inference is *defuzzification*. Defuzzification is the process of transforming a fuzzy set into a single crisp value. In function approximation or decision problems, the output typically has to be expressed by a single value. For example, in fuzzy-based denoising, we want to eventually decide whether or not a connected component in image is a small noise that needs to be removed. There are many different methods of defuzzification including Center of Gravity (COG), Center of Area (COA), Middle of Maximum (MOM) etc. In this work, we use the COG which is one of the most popular defuzzification methods. Formally, let  $(A, m_A(x))$  be a fuzzy set defined on the universe of discourse  $x$ , then the defuzzified value of the set  $A$ , using the COG method, is defined to be  $\left[ \int x m_A(x) \right] / \left[ \int m_A(x) \right]$ , which is the  $x$ -coordinate of the center of gravity of the membership function.

### 3.3 Structural Noise Removal Using FIS

We have devised a fuzzy inference system for the detection of structural noise from binary images. We define the structural noise as any type of noise that is not an artifact, but

actually a part of the image. Consequently, we cannot expect to remove the structural noise by a general purpose image denoising operator such as a Gaussian filter. The structural noise is subjective and usually defined by some linguistic rules. What is noise in one application may be an important part of data in another application. For example, commas may be considered as structural noise in a word spotting application, however they may be important in a text-to-speech application.

In this section, we firstly present the FIS that we have developed for distinguishing the small noises from character dots, and then we will show how to extend the system for other types of structural noises such as commas, dashes, etc.

### **3.3.1 Feature Extraction**

In order to decide whether a connected component in a binary image is a separator or noise, we have to extract some properties (features) from the connected component. Then we construct the FIS systems so that they compute the degree of truth of a connected component being a dot, small noise, dash etc. based on the values of the features.

The features that we need to extract from a connected component in order to decide whether it is a dot or small noise could be as simple as: height, width, aspect ratio (defined as the ratio of height to width) and  $y$ -coordinate of the center of gravity (which can measure how close the connected component is to the upper baseline). However, for the detection of the other separators from more complex shapes, we add three more features: orientation, eccentricity, and compactness. Eccentricity is an indication of elongation [GW07], and compactness is an indication of solidness and convexity which is defined as follows.

Let  $B$  be a binary shape, for an arbitrary axis  $L$ , the compactness of  $B$  is defined as the average of density of shape pixels over all lines along the axis. The density of a shape for a given line is defined as the number of shape pixels lying on the line over the distance between the two farthest boundary-points (i.e. intersections of the line and the shape). We define the compactness of a shape as the average of compactness for horizontal and vertical axes.

Therefore we have 7 features in total. In order to facilitate the definition of the fuzzy sets, we want the values of the feature to be independent from the size and coordinate system of the image. Therefore, we normalize the height, width and  $y$ -coordinate of the center of gravity by the height of the image (i.e. number of rows when the image is represented by a raster data structure).

### 3.3.2 Specification of FIS

In this section, first we present the fuzzy sets that we define on each feature, and then we give the rule base for the detection of separators and small noises.



(a) Fuzzy sets defined on Normalized Y-COG      (b) Fuzzy set defined on Aspect Ratio

**Figure 3.2 Fuzzy sets defined on variables Normalized Y-COG and Aspect Ratio.**

The number of fuzzy sets that we define on an input variable depends on the context knowledge and how we are going to define the rules. This number is usually between 1

and 4. For example, in order to determine whether a small dot belongs to a character, a human expert uses a linguistic rule such as: “if the dot is near the top of the image then it most likely belongs to a character”. Therefore, in this case, only one or two fuzzy sets will be enough: TOP  $\equiv$  near the top of the image, and BOTTOM  $\equiv$  near the bottom of the image. The fuzzy sets that we define on each shape feature are given in Table 3.1.

**Table 3.1 Fuzzy sets defined on shape features.**

Feature	Fuzzy sets
Normalized Y-coordinate of Center of Gravity	TOP, BOTTOM
Aspect Ratio	AROUND_1
Normalized Height	SMALL_COMPARED_TO_NASW, EQUAL_TO_NASW, LARGE_COMPARED_TO_NASW, SMALL, MEDIUM, HIGH
Normalized Width	SMALL_COMPARED_TO_NASW, EQUAL_TO_NASW, LARGE_COMPARED_TO_NASW, SMALL, MEDIUM, HIGH
Orientation	HORIZONTAL, VERTICAL, DIAGONAL_LEFT, DIAGONAL_RIGHT
Eccentricity	SMALL, MEDIUM, HIGH
Compactness	SMALL, MEDIUM, HIGH

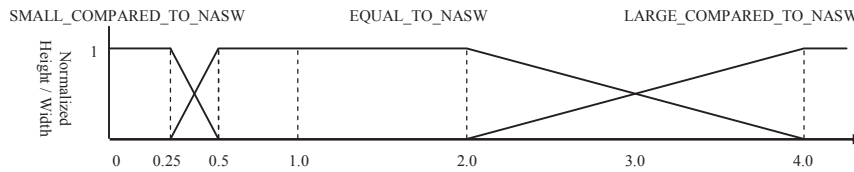
Fig. 3.2(a) shows the fuzzy sets TOP and BOTTOM that we define on the feature  $y$ -coordinate of the center of gravity (YCOG). On the feature Aspect Ratio (AR), we only define one fuzzy set: ARONUD\_1, which defines how close the aspect ratio is to unity. The membership function  $m_{\text{AROUND}_1}(x)$  is shown in Fig. 3.2(b). It is a triangular with the

value of 1 at  $x = 1$  which linearly goes to 0 at  $x = 0.5$  and  $x = 2$ , which means that the aspect ratio is not around 1 when the height is two times larger than the width, or the width is two times larger than the height.

In order to decide whether a small connected component is noise or part of the text, we have to have an estimate for the Average Stroke Width (ASW). For a binary image  $B$ , we take the median of run-lengths of black (text) pixels in all rows and all columns of the input image as an estimate for ASW:

$$ASW_B = \text{median}(\text{length}(R_H) \cup \text{length}(R_V)).$$

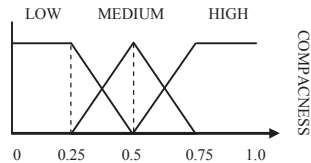
where  $R_H = \{\text{black runs in all rows of } B\}$  and  $R_V = \{\text{black runs in all columns of } B\}$ .



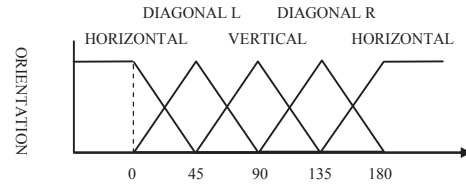
**Figure 3.3 Fuzzy sets defined on variables normalized height and normalized width.**

The size (height and width) of a dot that is part of the text is close to the stroke width. Therefore, we define three fuzzy sets on the normalized height and normalized width of a connected component to specify how small, equal or large these features are compared to the Normalized ASW (NASW). These fuzzy sets are called SMALL COMPARED TO NASW, EQUAL TO NASW and LARGE COMPARED TO NASW as shown in Fig. 3.3. Aside from these three fuzzy sets, we also define the three fuzzy sets of SMALL, MEDIM and LARGE as shown in Fig. 3.4(a). In fuzzy applications, these are the most typical fuzzy sets that we define on a real variable in the interval  $[0, 1]$ . In our

application, we define the same fuzzy sets on the input variables eccentricity and compactness. Finally, we define the four fuzzy sets of HORIZONTAL, VERTICAL, DIAGONAL LEFT and DIAGONAL RIGHT on orientation as shown in Fig. 3.4(b).



(a) Fuzzy set defined on Compactness



(b) Fuzzy set defined on Orientation

**Figure 3.4 Fuzzy sets defined on variables compactness and orientation.**

The rule base for the detection of each separator consists of a set of intuitively-designed linguistic rules. We start with the rule base for the detection of dots and small noises.

### ***3.3.2.1 Rule Base for Detection of Dots and Small Noises***

We define the rule base for the detection of dots and small noises to be composed of rules of the following form:

```

IF (Normalized Height is ...) AND (Normalized Width is ...) AND
   (Normalized YCOG is ...) AND (Aspect Ratio is ...) AND
   (Eccentricity is ...) AND (Compactness is ...) AND
   (Orientation is ...) THEN
   (Dot is ...) AND (Small Noise is ...);

```

Of course, the antecedent of a rule of this form does not need to contain all parts of the conjunction. We start by defining the two basic cases where 1) small noises are likely and character dots are unlikely; and 2) character dots are likely and small noises are unlikely.

The fuzzy rules corresponding to these two basic cases are as follows:

```
Rule 1 := IF (Normalized Height is SMALL_COMPARED_TO_NASW) AND
(Normalized Width is SMALL_COMPARED_TO_NASW) THEN (Dot is LOW) AND
(Small Noise is HIGH);
```

```
Rule 2 := IF (Normalized Height is EQUAL_TO_NASW) AND (Normalized Width
is EQUAL_COMPARED_TO_NASW) THEN (Dot is HIGH) AND (Small Noise is LOW);
```

Now, we can refine these rules by adding more knowledge about the location of the connected component. We know that if a small connected component appears near the bottom of the image, it is less likely to be a character dot, compared to when it appears near the top of the image. Therefore, based on the location of the connected component, we can decompose Rule 1 into two rules and modify Rule 2 as follows:

```
Rule 1-1 := IF (Normalized Height is SMALL_COMPARED_TO_NASW) AND
(Normalized Width is SMALL_COMPARED_TO_NASW) AND (Normalized YCOG is
BOTTOM) THEN (Dot is very LOW) AND (Small Noise is very HIGH);
```

```
Rule 1-2 := IF (Normalized Height is SMALL_COMPARED_TO_NASW) AND
(Normalized Width is SMALL_COMPARED_TO_NASW) AND (Normalized YCOG is not
BOTTOM) THEN (Dot is somewhat LOW) AND (Small Noise is somewhat HIGH);
```



Rule 2 := IF (Normalized Height is EQUAL\_TO\_NASW) AND (Normalized Width is EQUAL\_COMPARED\_TO\_NASW) AND (Normalized YCOG is not BOTTOM) THEN (Dot is very HIGH) AND (Small Noise is very LOW);

Where we have used the fuzzy hedges “very”/“somewhat” to increase/decrease the emphasis on their corresponding fuzzy sets. We can further refine these rules using more features such as aspect ratio and compactness. The complete rule base for the detection of dots and small noises is given in Appendix A1.

### 3.3.2.2 Rule Base for Detection of Dashes

A character dash (‘-’) is intuitively defined as an elongated shape that is almost horizontal, whose height is small, and whose width is medium (compared to average width of characters). The process of the definition of the rule base for the detection of dashes is similar to that of dots and small noises. We start with a few basic rules and then gradually refine them by adding more knowledge. The complete rule base for the detection of dashes is given in Appendix A2.

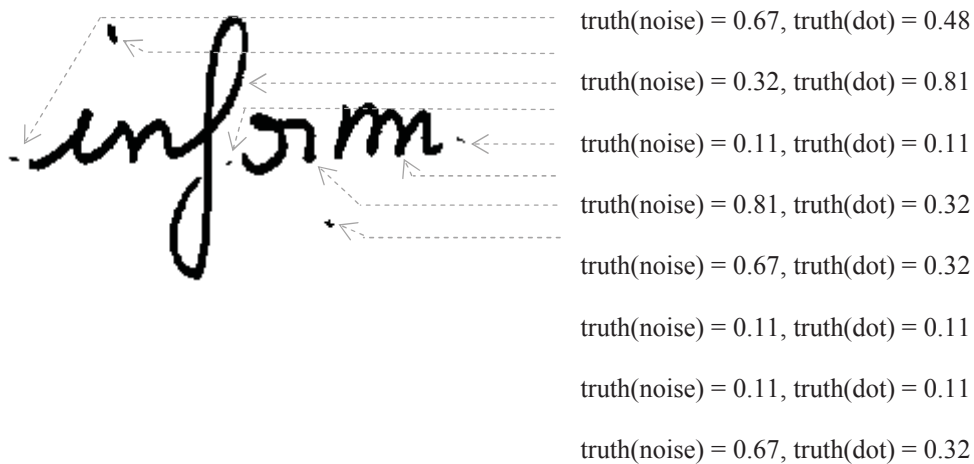


Figure 3.5 Result of applying the FIS-based noise removal filter to a handwritten word.

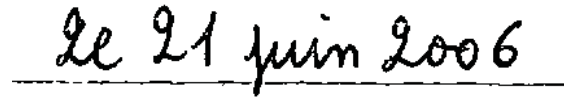
### 3.4 Experimental Results

In this section, we show the result of applying the FIS-based noise removal filter to some images of handwritten words. For an input document image, first we remove the margins. Next, we extract lines and binarize each line separately. Next, we estimate the average stroke width (ASW) locally, and extract the feature set for each connected component. Finally, we apply the FIS for detection of dots and small noises to each connected component  $C$  and we defuzzify the output to obtain the degree of truth of the connected component being a dot ( $T_{dot}$ ) and a small noise ( $T_{noise}$ ). We remove the connected component if it satisfies two conditions: 1) it is noise and 2) it is more noise than dot. In order to decide if a connected component is noise, we look at the degree of truth  $T_{noise}$ , which is a value between 0 and 1. Therefore, in the absence of any further information, if  $T_{noise}$  is higher than 0.5, we should take the connected component as noise.

Figure 3.5 shows the degrees of truth of dot and noise for each connected component of a handwritten word. Using the FIS-based noise removal filter, we are able to keep the dot that belongs to the word and remove all other noises. Of course, this filter is only designed for small noises and dots; therefore we cannot use it to remove other types of structural noise such as background lines. Figure 3.6 shows samples of handwritten text with guideline noise. We refer to background lines on ruled papers as guideline noise. These lines are typically used as guidelines to help the user keep their writing consistent.

A sample of handwritten text "Monsieur" written in black ink on a light blue horizontal guideline. The text is slightly above the line, and the guideline is visible as a thin blue line.

(a)

A sample of handwritten text "Le 21 juin 2006" written in black ink on a light blue horizontal guideline. The text is slightly above the line, and the guideline is visible as a thin blue line.

(b)

**Figure 3.6 Samples of handwritten text with guideline noise.**

The guidelines are usually printed in light colors, i.e. lighter than the ink that is used in pens. Therefore, in most cases we are able to remove the guidelines with proper binarization. However, in certain situations the binarization algorithm may not be able to remove the guidelines, for example when we apply a global binarization operator to the whole document. In such cases, we may still be able to remove the guidelines by a FIS-based noise removal filter. The rule-base for the FIS for the removal of guidelines is similar to the rule base for the detection of dashes that we discussed in section 3.3.2.2.

## Chapter 4

### *Line and Word Segmentation*

#### **4.1 Introduction**

Lines and words are building blocks of text. In document processing applications, we often need to divide a document into its constituent lines, and sometimes we need to further divide each line into its constituent words. Although both line segmentation and word segmentation in unconstrained documents can be considered as open-problems, generally speaking extracting lines from a document is more straightforward. The reason is that the text lines are almost well-defined based on geometrical information. However, words are not as well-defined. In handwritten documents, inter-word-spacing is sometimes wider than the intra-word-spacing and thus it is not always possible to segment the document at the word level perfectly using geometrical information only. Fortunately, perfect word segmentation is not always necessary. The level of details we have to divide a document into depends on the specific application and method. In some applications such as template-based approaches to word spotting, there is even no need to segment the document at line level. We devise a top-down approach to word spotting, where we need to extract the text lines and words or sub-words. In the following, first we present a literature survey on line and word segmentation algorithms. Then we describe the theory of fast Fourier-based steerable filtering that our line and word segmentation

methods are based on. Finally we present our line and word segmentation algorithm along with some experimental results.

#### **4.1.1 Background**

Due to intrinsic challenges of unconstrained documents, this problem has remained unresolved and thus, many different approaches to segmenting lines and words have been proposed so far [LSZT07, Kav10, DPB09, LGPH09]. Line segmentation approaches usually make two key assumptions: 1) the gap between neighboring text lines is significant; and 2) the text lines are reasonably straight, or else they have a single global skew angle. Word segmentation approaches usually make two key assumptions: 1) the gap between neighboring words is wider than the gap between characters belonging to the same word; 2) neighboring words are not connected together; in other words, any connected component of text belongs to only one word.

These assumptions are not always valid for handwritten documents. However, in most cases they are valid for documents with simple to moderate types of layouts.

It must be noted that a majority of the classical algorithms in the document processing literature are specifically developed for machine-printed types of documents [EDC97, SPJ97, NSV92], and not surprisingly they cannot provide an acceptable performance for unconstrained handwritten documents. Previously, there was a greater interest in processing machine-printed documents, but recently a new trend has developed to move beyond traditional machine-printed methodologies to deal with unconstrained documents as well. The basic ideas upon which these new algorithms are built are more or less the same considering the fact that several simplifying assumptions such as parallel lines of

text are not valid anymore. Moreover, this calls for more sophisticated pre-processing and post-processing techniques.

In [NSV92], a top-down segmentation method based on projection profiles is proposed. This method can only handle machine-printed documents because it is based on the assumptions of parallel text lines and large intra-line gaps. In a more recent study [PD03], the basic projection profiles technique is extended to deal with slightly curved lines of text. The idea is to form areas of text wherein the lines are parallel and then segment them using horizontal projection profile technique. The document spectrum method [O’G93] is a classic example of a bottom-up segmentation algorithm. It works by connecting neighboring connected components based on the geometric relationship between a fixed number of nearest neighbors. Docspectrum achieves good results for machine-printed as well as handwritten documents with slightly curved lines. Another bottom-up algorithm is [LSF94] based upon the three Gestalt criteria of proximity, similarity and direction continuity for perceptual grouping of connected components to text lines. General curve extraction techniques based on the Hough transform have also been used for text line detection [LGH07], [LSHF95]. Recently an almost real-time implementation of Hough transform has been developed [FO08]. However, sophisticated post-processing techniques are still needed for extraction of text lines after computing the transform. A partial-contour-following-based method to detect the separating lines is proposed in [ZTMR01]. The text slant is first detected and text line numbers are evaluated using partial projection. Next, a partial contour following for each line is performed in two opposite directions and finally, the adjacent lines are separated. Smearing is a common technique used in page segmentation algorithms. In [SSG05], the

authors have used the so-called Adaptive Local Connectivity Map (ALCM) which is a transformation operator replacing each pixel by the value of the sum of neighboring pixels within a horizontal distance. After smearing the image with this operator and then binarizing it, the connected components of the resultant image will correspond to candidate lines of text. A more elaborate algorithm using the ALCM technique is [KB06], where the authors have used pre-processing and post-processing steps to remove rule/margin lines and break connected text regions containing more than one line.

The general image segmentation method of level sets has recently been utilized in the realm of document image processing. In the algorithm proposed in [LZD+08a], after estimating a probability map for text lines, the level set method is applied to determine the boundary of neighboring text lines. This method is script-independent and doesn't require the neighboring lines to be absolutely parallel and straight. However, it is computationally demanding and perhaps not suitable for applications where speed is a major concern. Moreover, the segmentation results depend on the number of boundary evolution steps as pointed out in [DPB09]. The authors in [DPB09] have proposed the use of the Mumford-Shah (MS) model for text line segmentation because the text area only consists of two uniform regions, wherein the piecewise constant approximation of the MS model well suits the segmentation task. An advantage of the MS-based model over [LZD+08a] is that it segments the lines by minimizing the MS energy functional and thus unlike [LZD+08a] the results do not depend on the number of evolution steps.

## **4.2 Line Extraction Based on Fast Fourier-Based Steerable Filtering**

We developed a new line extraction method based on Fast Fourier-based Steerable (FFS) filtering. The algorithm is composed of two stages: fast filtering and local skew correction. In the following, first we present the theory of FFS filtering. Then, we describe the line map computation and post-processing steps of our line segmentation algorithm.

### **4.2.1 Fast Fourier-based Steerable Filtering**

For the extraction of the text lines based on filtering the two obvious choices for the kernel are box and Gaussian. If we use a box kernel, the output of the filtering is, by definition, the so called Adaptive Local Connectivity Map (ALCM) which is proposed by Shi et al. [SSG05]. According to the ICDAR 2009 Handwriting Segmentation Contest [GSL09], the ALCM-based algorithm is the best line segmentation algorithm for handwritten documents. However, in the original paper [SSG05], the authors implemented the ALCM by convolution in spatial domain. Here, we compute the map using FFS filtering which is based on the decomposition of the filter and Fast Fourier Transform (FFT) operations, resulting in significant speedup over the conventional convolution in spatial domain.

Another possible choice for the kernel is a Gaussian. Gaussian kernels are among the most commonly used kernels in image processing due to their desirable properties from both theoretical and computational point of view. The general case of an anisotropic Gaussian filter in two dimensions is defined by:



$$G(x, y; \sigma_x, \sigma_y) = \frac{1}{2\pi \cdot \sigma_x \cdot \sigma_y} e^{-\left(\frac{x^2}{2\sigma_x^2} + \frac{y^2}{2\sigma_y^2}\right)} \quad (4.1)$$

Where  $\sigma_x$  is the standard deviation along the  $x$ -axis, and  $\sigma_y$  is the standard deviation along the  $y$ -axis. This filter is axis-aligned, and thus can be used to analyze fairly horizontal or vertical structures. In order to analyze structures with arbitrary orientations, we have to “steer” (orient) the filter at arbitrary orientations. In [FA91] an efficient architecture is proposed to synthesize filters of arbitrary orientations from linear combinations of basis filters. However, according to this framework, no exact basis exists for rotating an anisotropic Gaussian. The existence of basis filters is important from a computing perspective. It is well known that direct implementation of filtering by convolution in spatial domain is slow, particularly in higher dimensions. If we can decompose a 2D filter as a linear combination of a set of 1D filters, we can compute the result of the filtering with much less calculation time. In [ASW03], the authors showed the decomposition of an oriented anisotropic Gaussian filter in two Gaussian line filters in non-orthogonal directions.

The general case of an oriented anisotropic Gaussian filter in two dimensions is obtained by rotating the basic filter defined in (4.1) by the desired angle  $\theta$ . Let’s denote the oriented anisotropic Gaussian filter by  $G_\theta(u, v, \sigma_u, \sigma_v, \theta)$ . We can define  $G_\theta$  as follows:

$$G_\theta(u, v; \sigma_u, \sigma_v, \theta) = \frac{1}{\sqrt{2\pi} \cdot \sigma_u} e^{-\frac{1}{2} \cdot \frac{u^2}{\sigma_u^2}} * \frac{1}{\sqrt{2\pi} \cdot \sigma_v} e^{-\frac{1}{2} \cdot \frac{v^2}{\sigma_v^2}} \quad (4.2)$$

Where “\*” denotes convolution, and the relation between the two coordinate systems  $x$ - $y$  and  $u$ - $v$  is given as follows:

$$\begin{pmatrix} u \\ v \end{pmatrix} = \begin{bmatrix} \cos \theta & \sin \theta \\ -\sin \theta & \cos \theta \end{bmatrix} \begin{pmatrix} x \\ y \end{pmatrix} \quad (4.3)$$

As can be seen, the filter is separated in  $u$  and  $v$  directions. However, this separation does not form a convenient basis from a computational perspective. We need to decompose the filter along the horizontal or vertical direction. The solution proposed in [ASW03] decomposes the filter along the  $x$ -direction and another direction as follows:

$$G_\theta(x, y; \sigma_x, \sigma_\varphi, \theta) = \frac{1}{\sqrt{2\pi} \cdot \sigma_x} e^{-\frac{1}{2} \frac{x^2}{\sigma_x^2}} * \frac{1}{\sqrt{2\pi} \cdot \sigma_\varphi} e^{-\frac{1}{2} \frac{t^2}{\sigma_\varphi^2}} \quad (4.4)$$

This equation represents a Gaussian filtering along the  $x$ -direction, followed by a Gaussian filtering along a line  $t = x \cos \varphi + y \sin \varphi$ . It can be shown that the standard deviations  $\sigma_x$  and  $\sigma_\varphi$ , and the intercept of the line  $\tan \varphi$  are computed as follows:

$$\sigma_x = \frac{\sigma_u \cdot \sigma_v}{\sqrt{\sigma_u^2 \sin^2 \theta + \sigma_v^2 \cos^2 \theta}} \quad (4.5)$$

$$\sigma_\varphi = \frac{1}{\sin \varphi} \sqrt{\sigma_u^2 \sin^2 \theta + \sigma_v^2 \cos^2 \theta} \quad (4.6)$$

$$\tan \varphi = \frac{\sigma_u^2 \sin^2 \theta + \sigma_v^2 \cos^2 \theta}{(\sigma_u^2 - \sigma_v^2) \cos \theta \sin \theta} \quad (4.7)$$

In the original paper [ASW03], the authors propose two implementations of Equation (4.4): one based on conventional convolution, and the other one based on recursive filters [JVVYV98]. In our work, we perform the filtering using FFT.

The computation of the FFT in the  $x$ -direction is straightforward. However, for the computation of the FFT in the  $\varphi$ -direction, we need to do interpolation because a point on the line may not necessarily lie on an image pixel. The authors in [ASW03] used linear interpolation. However, we will use nearest-neighbor interpolation in our approach because it facilitates the computation of the FFT as explained in the following.

#### **4.2.1.1 Computation of FFT in $\varphi$ -direction using Linear Interpolation**

In spatial domain, filtering along the line  $t$  with intercept  $\mu = \tan\varphi$  is achieved by [ASW03]:

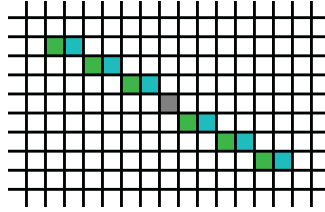
$$g_\theta[x, y] = w_0 g_x[x, y] + \sum_{j=1}^{\lfloor M/2 \rfloor} w_j (g_x[x - j/\mu, y - j] + g_x[x + j/\mu, y + j]) \quad (4.8)$$

Where  $g_x[x, y]$  is the input image filtered with the  $x$ -filter, and  $w_j$  is the filter kernel for half the sampled Gaussian from 0 to  $\lfloor M/2 \rfloor$ .

The coordinates  $y \pm j$  exactly lie on an image pixel, however the coordinates  $x \pm j/\mu$  coordinate may fall between two image pixels. In order to solve this problem, the authors in [ASW03] compute the value of the pixel of interest by the linear interpolation of the two neighboring pixels. Therefore, Equation (4.8) becomes:

$$\begin{aligned}
g_\theta[x, y] &= w_0 g_x[x, y] + \sum_{j=1}^{\lfloor M/2 \rfloor} w_j \{ a \cdot g_x[\lfloor x - j/\mu \rfloor, y - j] + a \cdot g_x[\lfloor x + j/\mu \rfloor, y + j] + \\
&\quad (1-a) \cdot g_x[\lceil x - j/\mu \rceil, y - j] + (1-a) \cdot g_x[\lceil x + j/\mu \rceil, y + j] \} \\
&= w_0 g_x[x, y] + a \cdot \sum_{j=1}^{\lfloor M/2 \rfloor} w_j \{ g_x[\lfloor x - j/\mu \rfloor, y - j] + g_x[\lfloor x + j/\mu \rfloor, y + j] \} + \\
&\quad (1-a) \cdot \sum_{j=1}^{\lfloor M/2 \rfloor} w_j \{ g_x[\lceil x - j/\mu \rceil, y - j] + g_x[\lceil x + j/\mu \rceil, y + j] \}.
\end{aligned} \tag{4.9}$$

where  $a$  is the interpolation factor.



**Figure 4.1** Linear interpolation for the computation of the FFT in a certain direction. Here the size of the filter is 7, and the orientation is  $30^\circ$ . Green pixels correspond to the coordinates that are rounded down to the closest column index, and blue pixels correspond to the coordinates that are rounded up to the closest column index.

According to Equation (4.9), filtering in the  $\varphi$ -direction can be achieved by two FFT operations, where each one is computed for a sequence of gray values at integer coordinates (Fig. 4.1). However, this formulation requires us to compute the coordinates of the integer pixels for each pixel of the image separately. In other words, it is desirable to compute the FFT along every diagonal (in the  $\varphi$ -direction) of the image only once and then use the FFT coefficients for the computation of the filtering (in the  $\varphi$ -direction). As we will show in the next section we can achieve this purpose by using nearest-neighbor interpolation rather than linear interpolation.

#### 4.2.1.2 Computation of FFT in $\varphi$ -direction using Nearest-Neighbor Interpolation

In Fig. 4.1, it is easy to see that an approximation to the line the  $\varphi$ -direction can be achieved by starting from the left-most pixel of the line and skipping every other pixel until the other end of the line. Therefore, using nearest-neighbor interpolation rather than linear interpolation, Equation (4.9) reduces to:

$$g_o[x, y] = w_0 g_x[x, y] + \sum_{j=1}^{\lfloor M/2 \rfloor} w_j \{g_x[\lfloor x - j/\mu \rfloor, y - j] + g_x[\lceil x + j/\mu \rceil, y + j]\} \quad (4.10)$$

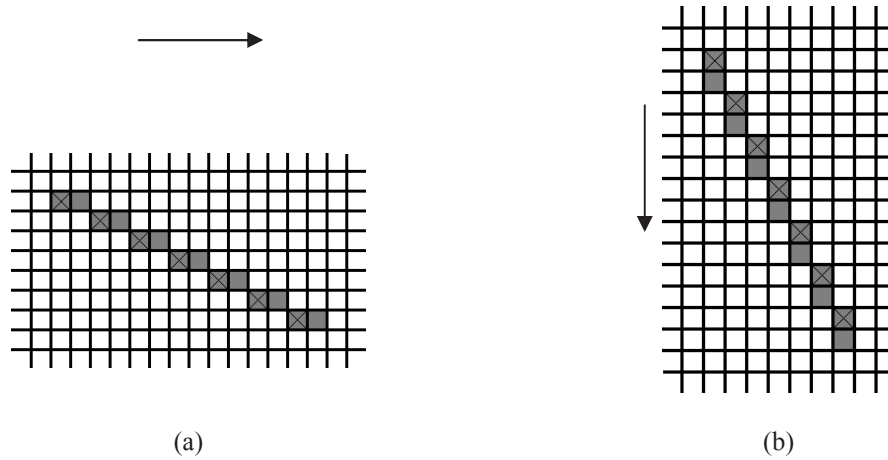
The advantage of Equation (4.10) lies in the fact that the pixels approximating the line are symmetric around the central pixel. Therefore, the filtering along the line in the  $\varphi$ -direction can be computed by down-sampling followed by the FFT along the corresponding diagonal of the image. In the example shown in Fig. 4.1, we picked out every other pixel, therefore the down-sampling factor is 2. In general, the down-sampling factor  $D_f$  is defined by the following equation:

$$D_f = \begin{cases} \text{round}(1/\mu) & \text{if } \mu < 1, \text{ or } \varphi < 45^\circ \\ \text{round}(\mu) & \text{if } \mu \geq 1, \text{ or } \varphi \geq 45^\circ \end{cases} \quad (4.11)$$

This equation simply states that if the line is more horizontal than vertical (i.e.  $\varphi < 45^\circ$ ) we down-sample along the horizontal direction, and similarly, if the line is more vertical than horizontal (i.e.  $\varphi \geq 45^\circ$ ) we down-sample along the vertical direction (Fig. 4.2).

Another advantage of using nearest-neighbor interpolation is now clear. The down-sampling factor is an integer which further reduces the complexity of the computations.

In general, if the down-sampling factor is not an integer but rather a rational fraction, the down-sampling operation can be implemented by two sampling operations: an integer up-sampling followed by an integer down-sampling.



**Figure 4.2** Down-sampling corresponding to nearest-neighbor interpolation when the line angle is less than  $45^\circ$  (a), and when it is more than  $45^\circ$  (b).

Having described the down-sampling operation, we present the procedure to perform the convolution in  $\varphi$ -direction using the FFT as follows. First, we define the  $\mu$ -diagonals of an image. A  $\varphi$ -diagonal of an image is a diagonal corresponding to the filter angle  $\varphi$  and the down-sampling factor  $D_f$ . For  $\varphi < 45^\circ$ , we obtain a  $\varphi$ -diagonal by starting from a pixel on the left-most column or the top-most row and then going  $D_f$  pixels to the right and 1 pixel to the bottom until we reach the right-most column or the bottom-most row of the image (Fig. 4.3). Similarly, for  $\varphi \geq 45^\circ$ , we obtain a  $\varphi$ -diagonal by starting from a pixel on the left-most column or the top-most row and then going  $D_f$  pixels to the bottom and 1 pixel to the right until we reach the right-most column or the bottom-most row of the image.

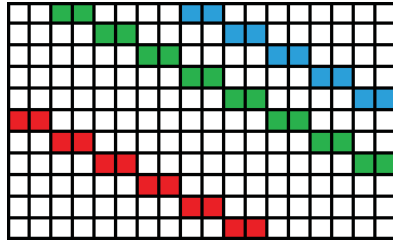


Figure 4.3 Some diagonals of an image corresponding to  $\varphi = 30^\circ$  and  $D_f = 2$ .

Now we use the following property of the down-sampling theorem in the discrete Fourier domain [Bra99]. Let  $x(n)$  be a discrete signal of length  $N$  in time domain, let  $C(\omega)$  be the Discrete Time Fourier Transform (DTFT) of  $x(n)$ . Let  $x(Mn)$  be the down-sampled version of  $x(n)$  corresponding to a down-sampling factor of  $M$ . Then the DTFT of  $x(Mn)$  denoted by  $C_d(\omega)$  have the following relation with  $C(\omega)$ :

$$C_d(k) = \frac{1}{M} \sum_{l=0}^{M-1} C(k + Ll) \quad (4.12)$$

Where  $L = N / M$ , and  $k = 0, 1, \dots, L - 1$ .

A numerical example is given below.

Let  $x = [ 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12 ]$  be the original signal. Let the down-sampling factor be 3. The down-sampled version of the signal denoted by  $x_d$  is obtained by picking out every 3 sample:  $x_d = [ 1, 4, 7, 10 ]$ .

The discrete Fourier transform of the original signal  $x$  denoted by  $c_x$  is computed as follows:

$$c_x(0) = 78, c_x(1) = -6 + 22.3923i, c_x(2) = -6 + 10.3923i, c_x(3) = -6 + 6i, c_x(4) = -6 + 3.4641i, c_x(5) = -6 + 1.6077i, c_x(6) = -6, c_x(7) = -6 - 1.6077i, c_x(8) = -6 - 3.4641i, c_x(9) = -6 - 6i, c_x(10) = -6 - 10.3923i, c_x(11) = -6 - 22.3923i.$$

The discrete Fourier transform of the down-sampled version of the signal  $x_d$  denoted by  $c_{xd}$  is computed as follows:

$$c_{xd}(0) = 22, c_{xd}(1) = -6 + 6i, c_{xd}(2) = -6, c_{xd}(3) = -6 - 6i.$$

Now, we can see that the following relations hold:

$$c_{xd}(0) = 1/3 \times \{ c_x(0) + c_x(4) + c_x(8) \} = 1/3 \times \{ 78 - 6 + 3.4641i - 6 - 3.4641i \} = 22.$$

$$c_{xd}(1) = 1/3 \times \{ c_x(1) + c_x(5) + c_x(9) \} = 1/3 \times \{ -6 + 22.3923i - 6 + 1.6077i - 6 - 6i \} = -6 + 6i.$$

$$c_{xd}(2) = 1/3 \times \{ c_x(2) + c_x(6) + c_x(10) \} = 1/3 \times \{ -6 + 10.3923i - 6 - 6 - 10.3923i \} = -6.$$

$$c_{xd}(3) = 1/3 \times \{ c_x(3) + c_x(7) + c_x(11) \} = 1/3 \times \{ -6 + 6i - 6 - 1.6077i - 6 - 22.3923i \} = -6 - 6i.$$

## 4.2.2 Computing Line Maps by Fast Oriented Anisotropic Gaussian Filtering

Having defined the FFS filtering, we compute the line map as follows. Firstly, we pre-process the input image. The pre-processing step involves 1) removing the margins from the page, and 2) correcting the global skew. Remember from the previous chapter that our proposed margin removal algorithm also corrects the global skew of the document. Secondly, we apply a set of FFS filters to the image and add the outputs together. The



reason why we use a set of filters rather than only one horizontal filter is that the text lines in handwritten documents may have multiple skew angles. Therefore, we steer the filters at all possible orientations that the text lines may exist. Thirdly, we binarize the resultant filtered image from the previous step order to obtain the binarized line map. Fourthly, we post-process the binarized line map; and fifthly and finally we obtain the locations of the text lines. The post-processing step involves 1) removing thin connected components in the binarized map that correspond to background noise, and 2) filling the remaining connected components vertically. The vertical filling operation is defined as finding the upper and lower profile of a connected component and then filling all the background pixels within any point on the upper profile and its corresponding point on the lower profile. The formal description of the line extraction algorithm is given in Fig. 4.4.

Note that Step 2 of the algorithm we could use the distributive property of convolution over addition. However, then the obtained kernel is not necessarily an oriented anisotropic Gaussian (the set of anisotropic Gaussians at different orientations is not closed under addition).

In our experiments, we only used 3 anisotropic Gaussian filters ( $N=3$ ), at  $\theta = -10.0, 0.0$  and  $10.0$  degrees, because the orientations of skewed text lines in our document images always fall within this range. Increasing the angular resolution (i.e. number of filters) did not affect the performance of the algorithm in terms of the localization of the text lines. We set  $\sigma_v$  to 15, which is around half the average height of the text lines in our database. We take the aspect ratio (i.e.  $\sigma_v / \sigma_u$ ) of the anisotropic Gaussians to be  $1/8$ , which is

around half the average aspect ratio of the text lines in our database. Thereby,  $\sigma_u$  is set to 120.

**Algorithm** LINESEGMENTATION( $I$ )  
*Input.* A document image  $I$ .  
*Output.* A set of bounding boxes  $B$  indicating the locations of the text lines in  $I$ .

Step 1. Pre-processing:  
    Step 1.1 Remove margins from  $I$ .  
    Step 1.2 Correct global skew of  $I$ .  
Step 2.  $F \leftarrow G_{\theta_1} * I + G_{\theta_2} * I + \dots + G_{\theta_N} * I$   
    where “ $*$ ” denotes convolution and,  
     $G_{\theta_1}, G_{\theta_2}, \dots, G_{\theta_N}$  are a set of anisotropic Gaussian filters oriented at  $\theta_1, \theta_2, \dots, \theta_N$ .  
Step 3. Binarize  $F$ .  
Step 4. Post-processing:  
    Step 4.1. Remove thin connected components in  $F$ .  
    Step 4.2. Vertically fill the connected components in  $F$ .  
Step 5.  $B \leftarrow$  Bounding boxes of the connected components in  $F$ .

Figure 4.4 Line segmentation algorithm based on FFS filtering.

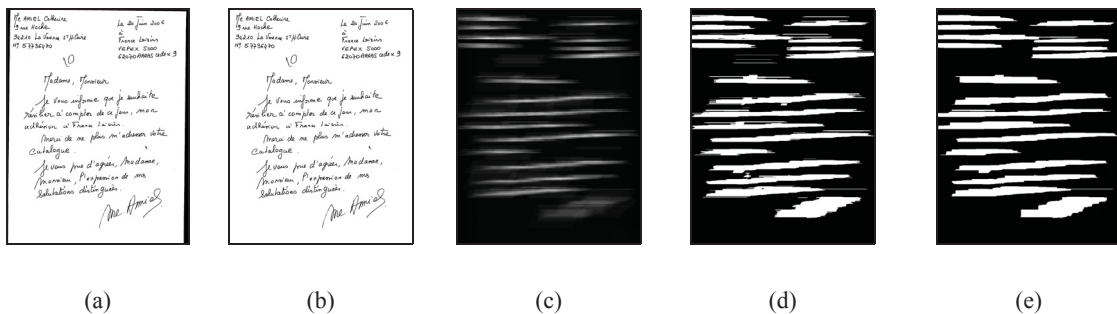
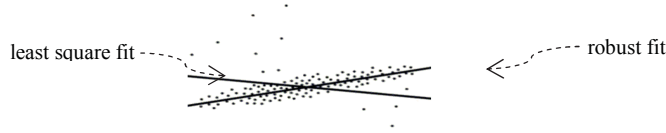


Figure 4.5 Result of applying the line extraction algorithm using FFS filters to a handwritten document with multiple skewed lines. (a) input document image. (b) image (a) after margin removal and global skew correction. (c) Line map obtained by applying FFS filters to the image and adding the outputs together. (d) image (c) after binarization. (e) image (d) after post-processing.

Fig. 4.5 shows the result of applying the line extraction algorithm to a handwritten document. As can be seen, the connected components in the binarized FFS map correspond to the text lines.



**Figure 4.6 Robust line fitting versus least square line fitting in presence of outliers.**

In order to facilitate the processing for the subsequent steps of the word spotting system, we perform a local skew correction inside the bounding box corresponding to each text line. We correct the local skew by the robust line fitting technique [PTVF07]. In robust line fitting, we maximize the probability of the data given the model rather than minimizing the squared sum of errors that is done in least-square fitting. For straight line fitting, the probability of the data given the model is defined as follows:

$$P(\text{data} \mid \text{model}) \propto \prod_{i=0}^{N-1} \left\{ \exp \left[ -\frac{1}{2} \left( \frac{y_i - \mathcal{Y}(x_i)}{\sigma} \right)^2 \right] \Delta y \right\} \quad (4.13)$$

When the data points are noisy, the robust line fitting gives a better fit than the least square because it is tailored to be less sensitive to outliers (Fig. 4.6).

### **4.3 Word Segmentation Based on Fast Oriented Anisotropic Gaussian**

#### **Filtering**

Once the text lines are extracted, we have to segment words on the same text line. Word segmentation in handwritten document is a difficult task because inter-word-spacing is

sometimes wider than the intra-word-spacing (Fig. 4.7). Thus, it is not always possible to segment the document at the word level perfectly using geometrical information only.



**Figure 4.7 Example of a handwritten line where the space between characters of the same word is wider than the space between two neighbouring words.**

Many different approaches to segmenting words are proposed so far. We may categorize word segmentation algorithms to either top-down, bottom-up or hybrid ones. We experimented with well-known algorithms from each category, and we concluded that the scale-space algorithm proposed by Manmatha and Rothfeder [MR05] gives the best results for our collection of unconstrained handwritten documents. We carry out the word segmentation task by an enhanced version of the scale-space algorithm. We obtain the scale-space using derivatives of fast anisotropic Gaussian filters implemented in the Fourier domain. Therefore, our approach to word segmentation is based on the same theory that we introduced for the extraction of lines. There are only two minor differences here. First, we do not need to steer the Gaussians at different orientations because words within a skew corrected line are reasonably straight, and moreover the aspect ratio of a word (ratio between its width to its height) is much less than that of a text line. Second, we have to use two Gaussian filtering operations in order to compute the Laplacian of Gaussian (LoG) operator. This is explained in more details in the following.

The scale-space is computed by convolving the image with a kernel that is the sum unmixed second partial derivatives of a Gaussian (in the  $x$  and  $y$  directions) [MR05]:

$$L(x, y; \sigma_x, \sigma_y) = G_{xx}(x, y; \sigma_x, \sigma_y) + G_{yy}(x, y; \sigma_x, \sigma_y) \quad (4.14)$$

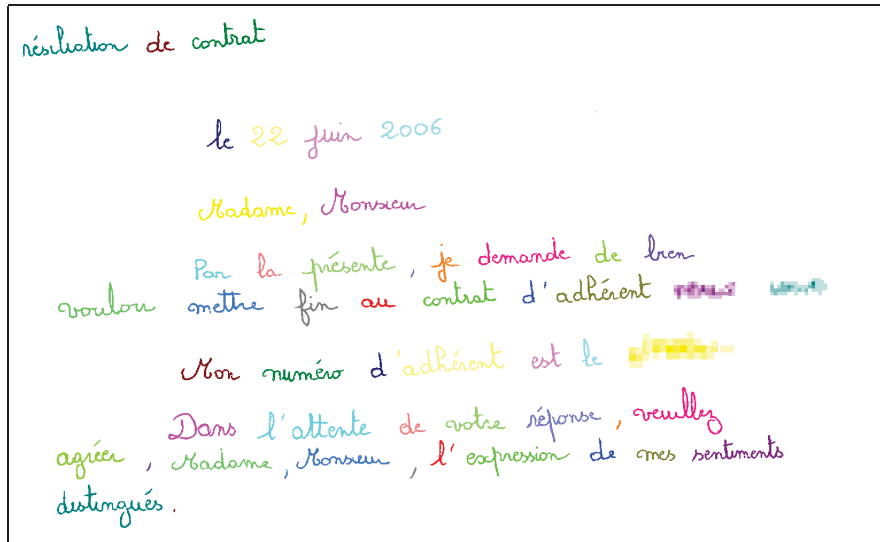
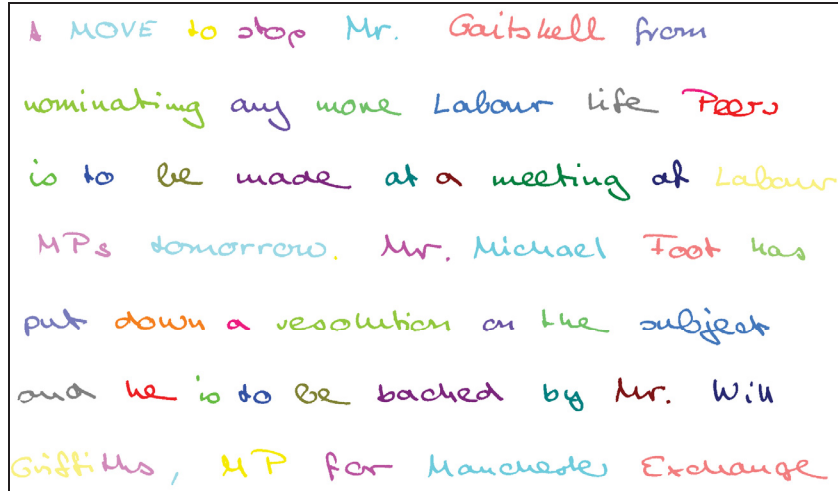


Figure 4.8 Output of line and word segmentation algorithms for a handwritten French document.

This operator is called Laplacian of Gaussian (LoG) filtering. It can be shown that the LoG operator can be approximated by the difference of two standard Gaussian filtering:

$$\begin{aligned}
 L(x, y; \sigma_x, \sigma_y) &\approx -\frac{1}{\pi \sigma_x^2 \sigma_y^2} \left[ 1 - \left( \frac{x^2}{2\sigma_x^2} + \frac{y^2}{2\sigma_y^2} \right) \right] e^{-\left( \frac{x^2}{2\sigma_x^2} + \frac{y^2}{2\sigma_y^2} \right)} \\
 &\approx \frac{1}{\pi \sigma_x \sigma_y} e^{-\left( \frac{x^2}{2\sigma_x^2} + \frac{y^2}{2\sigma_y^2} \right)} - \frac{1}{\pi K^2 \sigma_x \sigma_y} e^{-\left( \frac{x^2}{2K^2 \sigma_x^2} + \frac{y^2}{2K^2 \sigma_y^2} \right)}
 \end{aligned} \quad (4.15)$$

This equation actually subtracts a wide Gaussian from a narrow Gaussian in order to approximate the second partial derivative.



A MOVE to stop Mr. Gaitskell from  
nominating any more Labour life Peers  
is to be made at a meeting of Labour  
MPs tomorrow. Mr. Michael Foot has  
put down a resolution on the subject  
and he is to be backed by Mr. Will  
Griffiths, MP for Manchester Exchange

Figure 4.9 Output of line and word segmentation algorithms for a handwritten English document.

The output of the word segmentation algorithm for a handwritten French document, as applied to each line of text separately, is shown in Figure 4.8, where the word hypotheses are represented with different colors. The text lines are extracted by the FFS filtering that we described in the previous section. Another sample output is shown in Figure 4.9 for a handwritten English document from the IAM database.

## Chapter 5

### *Character Segmentation*

#### 5.1 Introduction

As we mentioned earlier, the main goal of this research is to develop a methodology for spotting *arbitrary* keywords. Therefore, we cannot rely on holistic word recognition approaches, because it is not possible to compile a large enough training database for all possible keywords. Consequently, our main approach is to use non-holistic (analytical) recognition methods, and so for general keyword detection, we need to either implicitly or explicitly divide each word into its constituent letters. This task is done by a character segmentation algorithm.

Most of the conventional character segmentation methods in the literature are based on the analysis of projection profiles or candidate segmentation points, where in either case the 2D information in the image is not taken advantage of effectively [RMKI09, HAI07]. The segmentation paths generated are usually obtained without taking into account the constraints on character shapes and neighboring characters. One fundamental assumption in these algorithms is that characters are separable by vertical lines (after slant correction). This assumption is correct for machine-printed and simple cursive text, but not for complicated styles of handwriting. In general, where there is considerable amount of overlapping between neighboring characters, they are not separable by straight lines. Samples of handwritten words with high overlapping are given in Fig. 5.1. In such cases,

application of a typical character segmentation algorithm would result in some damaged characters (i.e. some characters with missing parts and some characters with parts from neighboring characters).

We have developed a new character segmentation algorithm based on background skeletal graphs. Our proposed character segmentation algorithm is based on 2D data structures that correspond to arbitrary regions of the image, where any arbitrary character shapes can be circumscribed by a region or a sequence of regions. Consequently, the algorithm is capable of finding the perfect boundaries of a character no matter how much overlapping it may have with neighboring characters. Aside from the character segmentation, the character merging algorithm (which will be discussed later in this chapter) is benefited from the 2D representation. Incorporation of the context knowledge about characters to the merging algorithm is intuitive when we use data structures that correspond to characters or sub-characters.

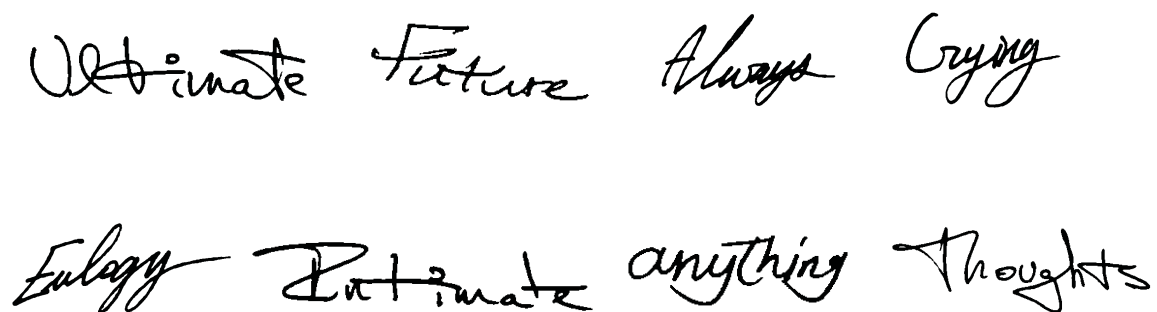


Figure 5.1 Samples of handwritten words with a lot of overlapping between characters.

Any character segmentation algorithm, be it implicit or explicit, needs more than only geometrical information in the word image in order to segment it perfectly. In other words, it is not always possible to perfectly segment a word image into its constituent



characters without knowing the corresponding transcription. The reason is that a word image may represent more than one transcription. Therefore, we have to segment the input word in all possible ways and then resolve the ambiguity using the context, which is a lexicon in the simplest form. In order to generate all valid segmentation hypotheses, we developed a new merging algorithm which is based on graph partitioning.

In the rest of this chapter, firstly we present the terminology and detailed description of the character segmentation algorithm and next the character merging algorithm. We will give illustrative examples as well as pseudo code for each algorithm.

## **5.2 Character Segmentation Based on Background Skeletal Graphs**

Our proposed approach to the segmentation of handwritten words is based on background skeletal graphs. A background skeletal graph is a geometric (location aware) graph corresponding to the skeleton of the background of the image. The main function of the algorithm is to keep the edges of the skeletal graph that correspond to possible segmentation paths. The decision whether or not an edge of the graph may correspond to a segmentation path is made based on the orientation, length and location of the edge. Before presenting the formal description of the algorithm, we will define the terminology that we are going to use.

### **5.2.1. Terminology of the Character Segmentation Algorithm**

Let  $G(V,E)$  be the skeletal graph corresponding to the background of the input word image  $I$ .  $G$  is a location-aware geometric graph where along with the neighborhood information,

we keep the coordinates of vertices and consequently the orientations of the edges. Then, we have the following definitions:

**End-point:** An *end-point* is defined as a vertex  $v \in V$  with a degree of 1.

**Junction-point:** A *junction-point* is defined as a vertex  $v \in V$  with a degree of greater than 2 (which is either 3 or 4 when the image is represented by a raster data structure).

**Branch:** A *branch* is defined as an edge  $e \in E$  starting from a *junction-point* and ending in an *end-point*.

**Curve:** A *curve* is an edge  $e \in E$  starting from an *end-point* and ending in an *end-point*.

**Downward/Upward branch:** A *downward/upward branch* is a *branch* whose start vertex lies on the upper/lower part of the graph.

**EPD:** An *EPD* denotes the *end-point* of a *downward branch*.

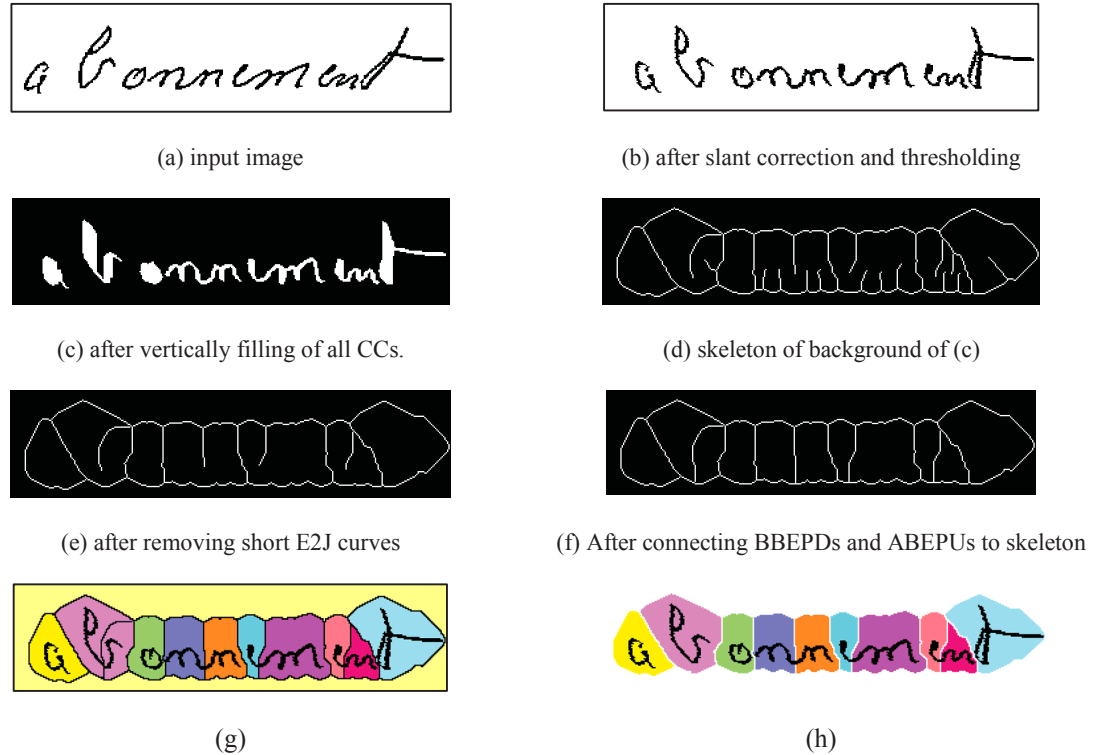
**EPU:** An *EPU* denotes the *end-point* of an *upward branch*.

**BEPD:** A *BEPD* denotes the *branch* corresponding to an *EPD* which goes below the baseline of the image.

**BEPU:** A *BEPU* denotes the *branch* corresponding to an *EPU* which goes above the baseline of the image.

### 5.2.2 Description of the Character Segmentation Algorithm

Having defined the terminology, we present the high level description of the character segmentation algorithm as follows. The first step is pre-processing which includes binarization followed by removal of isolated dots that are noise. Next, we compute the skeleton of the background of the image and the skeletal graph corresponding to it.



**Figure 5.2 Results of applying main steps of character segmentation algorithm to a handwritten word.**

As can be seen in Fig. 5.2, the branches of the skeletal graph correspond to the possible segmentation paths. Therefore, in order to form the character (or sub-characters) regions of the image from the skeletal graph, we apply the following rules in order: 1) we remove all curves and all short branches of the skeletal graph, because they do not correspond to any segmentation path; 2) for each *BEPD* of the graph, we connect it to the nearest point on the skeletal graph that is below the baseline; 3) for each *BEPU* of the graph, we connect it to the nearest point on the skeletal graph that is above the baseline; and 4) we remove all the remaining branches of the graph. The results of the main steps of the algorithm as applied

to a handwritten word is shown in Fig. 5.2. The pseudo code of the algorithm is given in Fig. 5.3.

```

Algorithm EXPLICITCHARACTERSEGMENTATION(I)
Input. A binary/grayscale image I representing a machine-printed or handwritten word.
Output. A list of regions corresponding to characters or sub-characters of I.

Step 1. Preprocessing:
    Step 1.1. Correct slant of I.
    Step 1.2. Binarize I.
    Step 1.3. Vertically fill inside each connected component of I.
    Step 1.4. Remove isolated dots in I.
Step 2. Skeletonization:
    S ← Skeleton of background of I.
Step 3. Formation of segmentation paths:
    Step 3.1.  $G(V,E) \leftarrow$  Geometric (location aware) graph corresponding to S.
    Step 3.2. Remove all short branches from G.
    Step 3.3. for each  $e \in E$ 
        do if e is a BEPD
            then connect e to the nearest skeleton point below it.
        else if e is a BEPU
            then connect e to the nearest skeleton point above it.
    Step 3.4. Remove all remaining curves from G.

```

**Figure 5.3** Explicit character segmentation algorithm using background skeletal graph.

### 5.3 Handling Over-segmentation and Under-Segmentation

The performance of a character segmentation algorithm is dropped by over-segmentation and under-segmentation errors. The output of our region-based segmentation algorithm is a list of disjoint regions corresponding to areas of image. We define over-segmentation as when there is more than one region whose union corresponds to one character. We define under-segmentation as when there is one region that corresponds to more than one character. For handling over-segmentation errors, we devise a merging method based on graph partitioning, and for detecting under-segmentation errors, we propose a

classification method based on fuzzy inference systems. In the following sections, we will present the descriptions of these two methods.

### **5.3.1 Character Merging Based on Graph Partitioning**

Over-segmentation is unavoidable without recognition. In other words, an explicit character segmentation algorithm, without knowing what a character is, may have to over-segment it. Moreover, sometimes we have intrinsic over-segmented characters which are due to noise, abrupt ink changes, binarization, or even the writing style. Indeed, certain characters are composed of more than one region: a main body and an accent or some dots. In handwriting it is not always trivial to decide to which neighboring character a dot or accent belongs to.

We devise a novel merging algorithm for handling broken characters which is based on graph partitioning with a heuristic search. This merging algorithm can be applied as a treatment step after character segmentation and before recognition. The algorithm is briefly explained in the following paragraphs.

Assume that we have an input sequence of connected components where we know each one corresponds to either a character or a piece of a character. We need to merge some pieces in order to form a sequence of characters out of the input sequence of characters and sub-characters. This problem may appear easy at a first glance, however as we will show, in general it is a NP-complete problem. Simply, the number of possible ways to form a sequence of characters out of a sequence of broken characters can be too many. Without knowing what the sequence means, we don't know how to merge the broken characters. This is a chicken-egg dilemma which one way to overcome is to generate all

the possible hypotheses in the segmentation phase and then resolve the ambiguity using the context. For handwriting recognition, the context needed to find the most likely candidate among the possible hypotheses is usually a dictionary of words. However, in general a dictionary alone is not enough and we need to employ a language model as well. Fig. 5.4 shows samples of handwritten words that may have more than one transcription. Therefore, over-segmentation is unavoidable without recognition/context, and the segmentation algorithm has to generate all the possible hypotheses.



(a) clear or dear



(b) man or won

**Figure 5.4 Samples of handwritten words that can have more than one transcription.**

The basic idea of the algorithm is to define a graph corresponding to the word image and then obtain the partitions of the graph that represent the different ways that the character pieces can be merged. Since graph partitioning is NP-complete and it is practically impossible to generate and then evaluate all the partitions, we develop a heuristic that can efficiently limit the search space to more promising partitions. Like any other heuristic search, theoretically the best solution is not guaranteed, however a good solution always is as we will show later on.

In the following, first we will define the neighborhood relation by which we obtain the neighborhood graph. Then, we will present the graph partitioning algorithm.

### 5.3.1.1 Neighbourhood relation

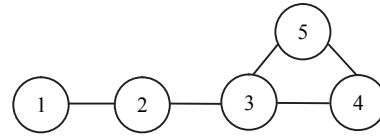
The input to the merging algorithm is a sequence of connected components  $S = \{s_0, s_1, \dots, s_{N-1}\}$  where each one corresponds to either a character or a piece of a character. We want to merge some mergeable connected components of  $S$  in order to create sequences of a certain smaller size. Therefore, we need to define a neighbourhood relation on the sequence in order to determine whether or not two connected components (in general two sequences of connected components) are mergeable. We consider two connected components to be neighbours if they are close or have enough vertical overlapping. To be more precise, two connected components  $s_i$  and  $s_j$  are neighbours if the distance between them is below a certain threshold  $D_{max}$ , or if the amount of overlapping between their projections on the x-axis is above a certain threshold  $O_{min}$ . The distance between two connected components is defined as the minimum of the Euclidean distances between any two of their respective points. The performance of the algorithm is not sensitive to the values for these thresholds. A typical value for  $D_{max}$  would be 5 pixels, and a typical value for  $O_{min}$  would be 50%. The higher the value for  $D_{max}$ , and the lower the value for  $O_{min}$ , the more flexibility the algorithm has to merge the connected components.

Having defined the neighbourhood relation, we create the graph  $G(V,E)$  from the sequence of connected components  $S$ , where each node  $v_i \in V$  corresponds to one connected component  $s_i$ , and for each pair of neighbouring connected components  $s_i$  and  $s_j$ , there is an edge  $e_{ij} = \langle v_i, v_j \rangle$ .

Fig. 5.5 shows an example of a neighbourhood graph corresponding to a handwritten word.



(a) handwritten word



(b) neighbourhood graph corresponding to (a)

**Figure 5.5 Example of a neighbourhood graph corresponding to a handwritten word.**

The neighbourhood graph determines how the connected components in the sequence should be merged. Having created the neighbourhood graph  $G(V,E)$ , we partition it into  $k$  parts  $V^1, V^2, \dots, V^k$  where the vertices in each partition determine the corresponding connected components that will be merged. In general, the number of parts  $k$  is between 1 (in which case all connected components will be merged together) to the number of vertices  $|V| = |S|$  (in which case no merging will be performed). However, in most cases we can limit the range of  $k$ . The number of parts is equal to the number of letters/digits of the word/numeral image, which can be estimated. Let  $A_{avg}$  be the average aspect ratio (height to width ratio) for the characters, then for an word/numeral image  $I$  with  $I_h$  rows and  $I_w$  columns, the average number of characters  $n_{chars}$  is  $A_{avg} * I_w / I_h$ . In order to eliminate the estimation errors, in our experiments we set  $k = n_{chars} - 3$  to  $n_{chars} + 3$ . It should be mentioned that in some applications such as word spotting, the value of  $k$  is known exactly, because we are going to spot a specific keyword with a known length in a document.

### **5.3.1.2 Graph Partitioning**

Having defined the neighbourhood graph  $G$ , we compute the partitions of  $G$  in order to find the mergeable connected components of  $S$ . However, we cannot simply compute all



the possible partitions and then evaluate them and choose the good ones, because the number of partitions is combinatorial in the number of nodes of the graph. For a complete graph with  $n$  nodes, the number of partitions is the  $n$ 'th Bell number denoted by  $B_n$ . Even for small size problems the search space is too large to be exhausted\*. Therefore, we need a way to prune such a large space of partitions. In other words, we want to generate a small set of partitions that is guaranteed to include the good partitions.

Our solution to this problem is a bottom-up one by using a heuristic to guide the search. We start with the trivial partition of size  $n = |V|$  where each node (corresponding to a connected component) is in one and only one partition. Then, we reduce the number of partitions by 1 at each iteration by merging all mergeable partitions and then keep the good ones for the next iteration. The good partitions are those ones with the highest scores. The score of a partition is a measure of how likely the corresponding sequence of connected components can be a sequence of characters. We use two properties of text in order to define the measure. First, connected components (corresponding to letters or digits) have more or less the same width. Second, there is not much overlapping between connected components as the text is written horizontally. Therefore, we want a measure that favours sequences having with more regularity and less overlapping over sequences with less regularity and more overlapping.

The regularity measure that we define is based on the Arithmetic Mean-Geometric Mean (AM-GM) inequality which states that for any list of  $n$  non-negative real numbers  $x_0, x_1, \dots, x_{n-1}$  we have:

---

\* To get an idea, even for a small value for  $n$  such as 15, the number of partitions is as large as  $B_{15} = 1,382,958,545$ .

$$\frac{x_0 + x_1 + \dots + x_{n-1}}{n} \geq \sqrt[n]{x_0 \cdot x_1 \cdots x_{n-1}} \quad (5.1)$$

and that equality holds if and only if we have  $x_0 = x_1 = \dots = x_{n-1}$ . A geometric interpretation of the AM-GM inequality is that, for  $n = 2$ , a square has the largest area among all rectangles with equal perimeter. In general, an  $n$ -cube has the largest volume among all  $n$ -dimensional boxes with the same edge length sum.

Let  $W = \{w_0, w_1, \dots, w_{n-1}\}$  be the widths of a sequence of connected components  $S$ , if we consider these  $w_i$ 's as edge lengths of an  $n$ -dimensional box, then the sequence of connected component that is the most regular (in terms of widths) is an  $n$ -dimensional box which has the largest volume. Therefore, for a list of widths  $W$ , we define the regularity measure as follows:

$$R(W) = \frac{w_0}{\sum w_i} \cdot \frac{w_1}{\sum w_i} \cdots \frac{w_{n-1}}{\sum w_i} = \frac{\prod w_i}{(\sum w_i)^n} \quad (5.2)$$

Note that we have divided each width  $w_i$  by the sum of the widths in order to normalize the perimeter to 1. Thus, the maximum of  $R(W)$  is  $\frac{1}{n^n}$  which is reached when  $w_0 = w_1 = \dots = w_{n-1}$ . Since we want to combine  $R(w)$  with other measures for the computation of the total measure, we divide it by the maximum to derive the normalized regularity measure  $R_{Norm}$ :

$$R_{Norm}(W) = n^n \cdot \frac{\prod w_i}{(\sum w_i)^n} \quad (5.3)$$

Now, obviously we have  $0 < R_{Norm}(W) \leq 1$ . However, in practice we implement  $R_{Norm}(W)$  by taking logarithm of both sides in order to avoid overflows. Therefore, we re-write Equation (5) as follows:

$$R_{Norm}(W) = \exp(n \ln(n) + \sum \ln(w_i) - n \ln(\sum w_i)) \quad (5.4)$$

$R_{Norm}(W)$  measures how regular a sequence of connected components is in terms of their widths. In order to quantify the amount of vertical overlapping between connected components, first we define the percentage of the overlapping  $O_p$  between two line segments  $L_i$  and  $L_j$  as follows:

$$O_p(L_i, L_j) = \begin{cases} 0 & \text{if } L_i \text{ and } L_j \text{ have no overlap.} \\ \frac{2|L_{i,j}|}{|L_i| + |L_j|} & \text{if } L_i \text{ and } L_j \text{ have some overlap where } L_{i,j} \text{ is the line segment in common.} \\ 1 & \text{if } L_i \text{ is completely inside } L_j \text{ or vice versa.} \end{cases} \quad (5.5)$$

For a set of line segments  $L = \{L_0, L_1, \dots, L_{n-1}\}$ , we define the normalized total amount of overlapping as follows:

$$O_p(L) = \frac{2}{n(n-1)} \sum_{i=0}^{n-2} \sum_{j=i+1}^{n-1} O_p(L_i, L_j) \quad (5.6)$$

which is the average amount of overlapping between all pairs of line segments in  $L$ . The minimum of  $O_p(L)$  is 0 when there is no overlapping between any pairs of line segments, and the maximum is 1 when all pairs have complete overlapping.

Now, in order to define the score  $S_G$  of a neighbourhood graph  $G$ , we combine  $R_{Norm}(W)$  and  $O_p(L)$  in the following way:

$$S_G = R_{Norm}(W) \times ( 1 - O_p(L) ) \quad (5.7)$$

The maximum of  $S_G$  is 1, which is reached when the bounding boxes corresponding to the partitions have all the same width and there is no vertical overlapping between any pairs of bounding boxes.

Having defined the score for neighbourhood graphs, we describe the graph partitioning algorithm as given in Fig. 5.6. The order of the algorithm is  $O( N \times |E| \times (|V| - N_{min}) )$ , where  $|E|$  is the number of edges of the neighbourhood graph  $G_0$ , which is the number of pairs of connected components of  $S$  that are neighbours according to the neighbourhood relation;  $|V|$  is the number of vertices of  $G_0$ ;  $N_{min}$  is the desired minimum number of connected components in an output sequence of connected components; and  $N$  is the number of best graphs that are kept at each level of the search. In our experiments we set  $N = 50$ .

**Algorithm MERGECONNECTEDCOMPONENTS**( $S, N_{min}, N_{max}$ )

*Input.* A sequence of connected components  $S$  corresponding to characters or sub-characters.

*Output.* A set of output sequence of connected components  $W$  with sizes of  $\geq N_{min}$  and  $\leq N_{max}$ .

Step 1.  $Q \leftarrow \{\}$

Step 2. Define of neighbourhood relation  $R$ .

Step 3.  $G_0(V_0, E_0) \leftarrow$  the neighbourhood graph corresponding to  $S$ .

where  $|V_0| = |S|$  and  $|E_0| =$  number of pairs of connected components of  $S$  which are neighbours according to  $R$ .

Step 4.  $Q \leftarrow$  PARTITIONGRAPH(  $\{G_0\}, Q, N_{min}, N_{max}$  ).

Step 5.  $W \leftarrow$  Set of sequence of connected components corresponding to  $Q$ .

Step 6. **return**  $W$ .

**Algorithm PARTITIONGRAPH**( $P, Q, N_{min}, N_{max}$ )

*Input.* A set of partitioned graphs  $P$  where the partitions of each graph determine the connected components of  $S$  that are merged together.

*Output.* A set of partitioned graphs  $Q$  with sizes of  $\geq N_{min}$  and  $\leq N_{max}$ .

Step 1. Initialize an empty list of  $N$ -best partitioned graphs  $T_N$  (for the next level of search).

Step 2. **for** each partitioned graph  $G(V, E) \in P$

**do if**  $|V| < N_{min}$

**then continue.**

**else if**  $N_{min} \leq |V| \leq N_{max}$

**then**  $Q \leftarrow Q \cup \{G\}$ .

**if**  $N_{min} == |V|$

**then continue.** (\* no more merging is needed \*)

**for** each edge  $e \in E$

**do** merge the two end partitions (i.e. two end sets of vertices) of  $e$  to make  
        a new partitioned graph  $G_{new}$ .

        Insert  $G_{new}$  to  $T_N$ .

Step 3. Compute the score of each partitioned graph  $T_N$  using Equation (5.7) and keep the  $N$ -best ones.

Step 4. PARTITIONGRAPH(  $T_N, Q, N_{min}, N_{max}$  )

Step 5. **return**  $Q$ .

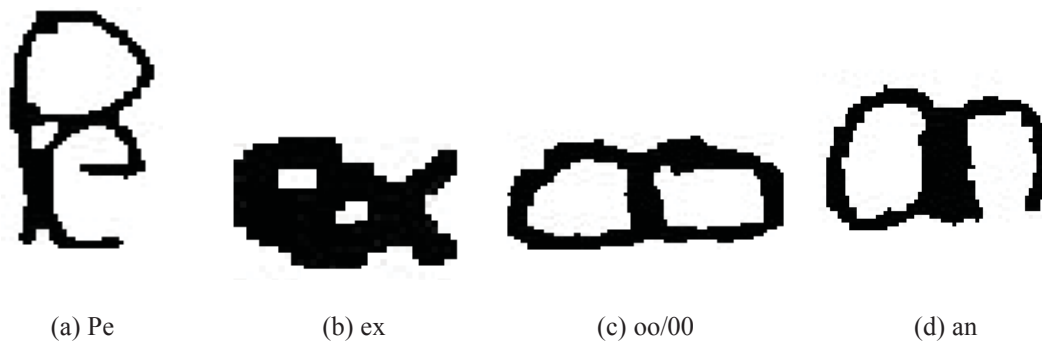
**Figure 5.6 Merging algorithm for sequence of connected components based on graph partitioning.**

### 5.3.2 Detection of Under-segmented Pairs of Handwritten Characters

#### Using Fuzzy Inference System

Under-segmented characters are the other type of error in the output of an explicit character segmentation algorithm. In our algorithm, under-segmented errors are the

results of branches of the skeletal graph that are not deep enough to form a segmentation path. This may happen where neighboring characters are too close together, due to the writing style or improper binarization. Fig. 5.7 shows samples of handwritten pairs of characters without deep enough skeletal branches on segmentation paths.



**Figure 5.7** Samples of handwritten pairs of characters without deep enough skeletal branches on segmentation paths.

It is important to detect under-segmented characters as they will adversely affect the process of word recognition and consequently spotting. This is due to the fact that the output of a character classifier for a pair of characters that it has not been trained for is unpredictable.

For the detection of under-segmented pairs of characters we devise a classifier based on a Fuzzy Inference System (FIS) using a set of features called Average Number of Transitions (ANTs) that we specifically design for this classification task. In the following, we will describe the ANT features and then the rule-base for the FIS. Finally, we will present the database that we created for the evaluation of our method and show the effectiveness of our approach.

### **5.3.2.1 Average Number of Transition (ANT) Features**

By looking at the under-segmented pairs, some of which are shown Fig. 5.7, we notice that the basic feature that distinguishes a binary image that represents more than one character from a binary image that represents one character (or part of a character) is the number of gaps in the image. The more gaps an image has, the likelier it is an under-segmented pair of characters.

The number of gaps in a row (or column) of a binary image is actually the number of transitions between black and white runs in that row (or column). Therefore, in order to estimate the average number of gaps for the whole image, we compute the average of transitions between black and white runs over all rows and columns of the image. We are able to distinguish most characters from under-segmented characters by counting the number of horizontal gaps only. However, for few characters such as ‘m’/‘M’ and ‘w’/‘W’ whose average number of horizontal gaps is 2 or more, we have to make the decision based on the number of horizontal and vertical gaps. The average number of vertical gaps for these characters is 0, which can separate them from a pair of under-segmented O’s (Fig. 5.7(c)) whose average number of vertical gaps is 1.

We formally define the ANT features as follows. Let  $I_{M \times N}$  denote a binary image with  $M$  rows and  $N$  columns that represents part of a character, a character or a sequence of characters. Let  $R_i$  denote the  $i$ ’th row, and  $C_j$  denote the  $j$ ’th column of  $I$  where  $0 \leq i \leq M - 1$ , and  $0 \leq j \leq N - 1$ .

We define a salient white run in a row (or column) of an image as a long-enough sequence of white pixels that is surrounded by two long-enough sequences of black

pixels on each side. A run is considered as long-enough if its length is greater than or equal to a threshold. We use two thresholds, one for white runs and the other one for black runs. Let  $L_{Ri}^W(T_W, T_B)$  denote the number of salient white runs in  $R_i$  where  $T_W$  is the threshold for white runs, and  $T_B$  is the threshold for black runs. We use these thresholds so as the average number of gaps is not sensitive to short runs that may correspond to noise. Assuming that in the binary image  $I$ , the background is represented by white pixels and the text is represented by black pixels, a reasonable value for  $T_B$  would be somewhere between the minimum stroke width and average stroke width. In our experiments, we obtained the best classification results with  $T_B = 2$ , which means that the classification is not too sensitive to the value that we choose for  $T_B$  as long as we make sure that the value is smaller than the average stroke width. For  $T_W$ , we choose a range of values and then compute the average of  $L_{Ri}^W(T_W, T_B)$  over this range.

Let  $T_{Wmin}$  be the minimum and  $T_{Wmax}$  be maximum in the range of values for  $T_W$ . Then, we define the average number of gaps  $G_{Ri}$  in the  $i$ 'th row of  $I$  as follows:

$$G_{Ri} = \left( \sum_{T_W=T_{Wmin}}^{T_{Wmax}} L_{Ri}^W(T_W, T_B) \right) / (T_{Wmax} - T_{Wmin} + 1) \quad (5.8)$$

In our experiments, we set  $T_{Wmin}$  to 2, and  $T_{Wmax}$  to 4.

Having defined the average number of gaps, we define the set of features as follows:

**F<sub>R01</sub>**: Normalized number of rows with 0 or 1 gaps.

**F<sub>R2+</sub>**: Normalized number of rows with 2 or more gaps.

**F<sub>R3+</sub>**: Normalized number of rows with 3 or more gaps.

**F<sub>C0</sub>**: Normalized number of columns with 0 gaps.



$F_{C1}$ : Normalized number of columns with 1 gap.

$F_{C3+}$ : Normalized number of columns with 3 or more gaps.

Where we normalize a number by dividing it by the length of the dimension that it is computed for, which is the number of rows of the image for  $F_R$  features, and the number of columns of the image for  $F_C$  features.

Besides these transition-based features, we define the Aspect Ratio (AR) of the image as the last feature:

$F_{AR}$ : ratio of the height of the image ( $M$ ) to its width ( $N$ ).

### 5.3.2.2 Fuzzy Inference System (FIS)

Having defined the features, we need to define the fuzzy sets on each feature. The three basic membership functions for the definition of the fuzzy sets are given below.

**Triangular**: a triangle defined by the  $x$ -coordinates of the three vertices as shown in Fig. 5.8(a).

**ShoulderLeft**: a trapezoid that extends to  $-\infty$ , thereby defined by the  $x$ -coordinates of the two vertices of the right boundary as shown in Fig. 5.8(b).

**ShoulderRight**: a trapezoid that extends to  $+\infty$ , thereby defined by the  $x$ -coordinates of the two vertices of the left boundary as shown in Fig. 5.8(c).

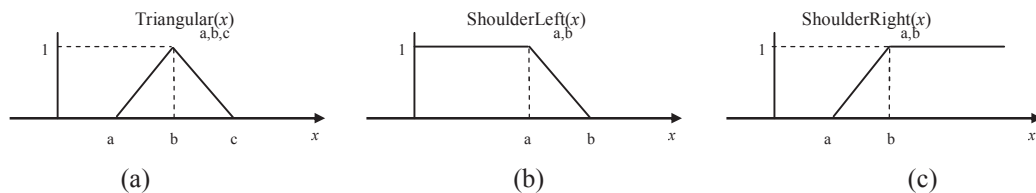


Figure 5.8 Three basic membership functions for the definition of fuzzy sets.

**Table 5.1 Fuzzy sets defined on shape features.**

<b>Variable</b>	<b>Fuzzy sets</b>
<b>F<sub>R01</sub></b>	HIGH := ShoulderRight <sub>0.95, 1.0</sub>
<b>F<sub>R2+</sub></b>	HIGH := ShoulderRight <sub>0.1, 0.2</sub>
<b>F<sub>R3+</sub></b>	HIGH := ShoulderRight <sub>0.01, 0.02</sub> TOO_HIGH := ShoulderRight <sub>0.3, 0.6</sub>
<b>F<sub>C0</sub></b>	HIGH := ShoulderRight <sub>0.95, 1.0</sub>
<b>F<sub>C1</sub></b>	HIGH := ShoulderRight <sub>0.3, 0.6</sub>
<b>F<sub>C3+</sub></b>	HIGH := ShoulderRight <sub>0.1, 0.2</sub>
<b>F<sub>AR</sub></b>	LOW := ShoulderLeft <sub>0.2, 0.33</sub> HIGH := ShoulderRight <sub>2.0, 3.5</sub>
<b>UnderSegmented</b>	LOW := ShoulderLeft <sub>0.25, 0.5</sub> MEDIUM := Triangular <sub>0.25, 0.5, 0.75</sub> HIGH := ShoulderRight <sub>0.5, 0.75</sub> TOO_HIGH := ShoulderRight <sub>0.75, 0.85</sub>

The fuzzy sets that we define on each variable (the seven features and the output variable) are given in Table 5.1.

The complete rule base for the under-segmented detection FIS is defined as follows.

Rule #1. if  $F_{C3+}$  is HIGH then Undersegmented is HIGH.

Rule #2. if  $F_{AR}$  is not HIGH and  $F_{R3+}$  is TOO\_HIGH then Undersegmented is TOO\_HIGH.

Rule #3. if  $F_{AR}$  is HIGH and  $F_{R01}$  is HIGH and  $F_{C3+}$  is not HIGH then Undersegmented is very LOW.

Rule #4. if  $F_{AR}$  is HIGH and  $F_{R01}$  is HIGH and  $F_{C3+}$  is HIGH then Undersegmented is LOW.

Rule #5. if  $F_{AR}$  is HIGH and  $F_{R01}$  is not HIGH then Undersegmented is MEDIUM.

Rule #6. if  $F_{AR}$  is LOW and  $F_{R01}$  is not HIGH then Undersegmented is HIGH.

Rule #7. if  $F_{AR}$  is LOW and  $F_{R01}$  is HIGH and  $F_{C3+}$  is not HIGH then Undersegmented is MEDIUM.

Rule #8. if  $F_{AR}$  is LOW and  $F_{R01}$  is HIGH and  $F_{C3+}$  is HIGH then Undersegmented is somewhat HIGH.

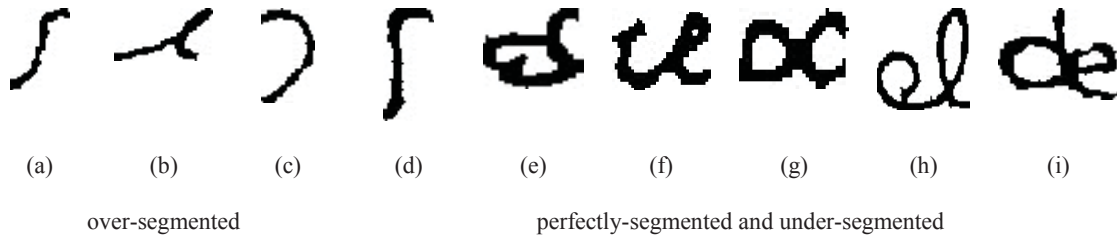
Rule #9. if  $F_{AR}$  is not LOW and  $F_{AR}$  is not HIGH and  $F_{R01}$  is HIGH then Undersegmented is LOW.

Rule #10. if  $F_{AR}$  is not LOW and  $F_{AR}$  is not HIGH and  $F_{R01}$  is not HIGH and (  $F_{R2+}$  is HIGH or  $F_{R3+}$  is HIGH ) then Undersegmented is HIGH.

Rule #11. if  $F_{AR}$  is not LOW and  $F_{AR}$  is not HIGH and  $F_{R01}$  is not HIGH and (  $F_{R2+}$  is not HIGH and  $F_{R3+}$  is not HIGH ) then Undersegmented is MEDIUM

### 5.3.2.3 Experimental Results

For the evaluation of our under-segmented detection approach we created a database of handwritten characters. Each image in the database is either part of a character (over-segmented), one character (perfectly-segmented), or more than one char (under-segmented). The corresponding label for each image is the integer that best describes the number of characters in the image. That is 0 for an over-segmented character, 1 for a perfectly-segmented character, and 2 or more for an under-segmented sequence of characters.



**Figure 5.9 Samples from database of handwritten characters for evaluation of under-segmented detection method.**

We automatically generated the images by applying the character segmentation algorithm to a randomly selected subset of words from the IAM database. Out of all segmented images, we chose 2000 over-segmented characters, 2000 perfectly-segmented characters and 12000 under-segmented sequences of characters. Next, we used our under-segmented detection method to automatically label the images, and finally we examined all the samples manually and corrected the labeling mistakes. Some samples of the images in this database are shown in Fig. 5.9.

The FIS-based classifier achieves a correct classification rate of %96 on this database. The correct classification rate for over-segmented and perfectly segmented samples is

96.6, and for under-segmented samples is 95.5. It should be noted that these rates are approximations to the real performance of the system which can be higher. In handwriting, we can find many shapes that can be considered as one or two characters (Fig. 5.9(f-h)). In such cases, it is better to segment (or over-segment) the shape, because we will have to resolve the ambiguity using recognition/context.



**Figure 5.10** Samples of under-segmented pairs of handwritten characters that are correctly classified by under-segmented detection method.

Fig. 5.10 shows samples of under-segmented pairs of handwritten characters that are correctly classified by fuzzy inference system. Fig. 5.11 shows some samples that are misclassified.



**Figure 5.11** Samples of handwritten characters that are misclassified by under-segmented detection method.

## Chapter 6

### *Cursive Character Recognition*

#### **6.1 Introduction**

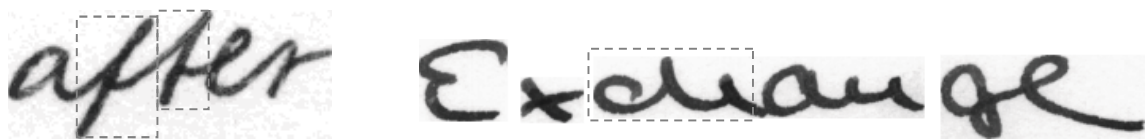
Recognition of printed characters using computers has been one of the first and most successful applications of pattern recognition. Optical Character Recognition (OCR) has been an active field of research for more than three decades. There are hundreds of hundreds of approaches proposed to recognition of machine-printed and handwritten characters for different scripts [CSSJ09]. For machine-printed Latin scripts, the problem can be considered as already solved at least when the level of noise is low [Fuj08]. On applications where clear imaging is available typical recognition rates for machine-printed characters exceed 99%. However, the difficulty is in dealing with handwritten characters (and also when the images are noisy). The difficulty of the recognition of handwritten characters lies in the fact that there can be as many handwriting styles as there are people. In fact, it is widely believed that each individual's handwriting is unique to themselves. In the discipline of forensic science, handwriting identification, which is the study of the identification or verification of the writer of a given handwritten document, is based on the principle that the handwritings of no two people are exactly alike. This means that the number of forms that a handwritten character can take is too many, making the recognition a difficult task even for humans.

'a' or 'c'?		'e' or 'c'?	
'a' or 'Q'?		'e' or 'R'?	
'A' or 'H'?		'g' or 'y'?	
'A' or 'R'?		'H' or 'M'?	
'a' or 'u'?		'H' or 'N'?	
'a' or 'w'?		'J' or 'I'?	
'b' or 's'?		'J' or 'N'?	
'b' or 'G'?		'J' or 's'?	
'b' or 'o'?		'k'/'K' or 'R'?	
'c' or 'L'?		'n'/'N' or 'm'/'M'?	
'c' or 'o'?		'n'/'N' or 'w'?	
'd' or 'J'?		'p'/'P' or 'f'?	
'd' or 'o'?		'r' or 'T'?	
'D' or 'O'?		'r' or 'v'/'V'?	
'D' or 'P'?		'r' or 'y'/'Y'?	

Figure 6.1 Fuzziness in handwriting. Examples of letters from NIST SD19 database which may be confused with each other.

Fig. 6.1 shows examples of pairs of letters from NIST SD 19 database [GBC+94] which may be confused with each other. According to our analysis on this database, there are 29 pairs of letters (lower case and upper case) which in some handwritten styles may be confused with each other. Fig. 6.1 shows samples of these confusing pairs of characters: <'a', 'c'>, <'a', 'Q'>, <'A', 'H'>, <'A', 'R'>, <'a', 'u'>, <'a', 'w'>, <'b', 's'>, <'b', 'G'>, <'b', 'o'>, <'c', 'L'>, <'c', 'o'>, <'d', 'J'>, <'d'/'D', 'o'/'O'>, <'D', 'P'>, <'e', 'c'>, <'e', 'R'>, <'g', 'y'>, <'H', 'M'>, <'H', 'N'>, <'J', 'I'>, <'J', 'N'>, <'J', 's'>, <'k'/'K', 'R'>, <'n'/'N', 'm'/'M'>, <'n'/'N', 'W'>, <'p'/'P', 'f' >, <'r', 'T'>, <'r', 'v'/'V'>, <'r', 'y'/'Y'>.

The samples shown in Fig. 6.1 are isolated letters in the sense that during the collection of the database participants were asked to write these letters separately within special forms. The problem is more challenging when the letters are written cursively. Fig. 6.2 shows samples of handwritten words from the IAM database [MB02] with letters that are difficult to recognize correctly in isolation.



These two letters are almost identical, whereas one is 'f' and the other is 't'.

The pair 'ch' can be recognized as 'di', 'ai', 'cu', 'ou', 'on', 'om' etc.

**Figure 6.2 Samples of handwritten words from the IAM database with letters that are difficult to recognize correctly in isolation.**

Despite the inherent challenges in handwritten characters, there has been considerable success in handwritten OCR systems. For isolated handwritten letters, the performance of



state-of-the-art techniques reported on standard databases of such as NIST SD 19 is around 95% to 96.82% [CSSJ09, MCS06]. For cursive handwritten letters, the state-of-the-art word recognition engines are reported to have a recognition rate of around 73.51% to 88.10% at character level [KBJO10, GLF+09]. Of course, a lower recognition rate at characters level does not necessarily correspond to a lower recognition rate at word level, partly because not all combinations of characters corresponds to lexicon entries. Despite the lower performance of recognition engines for cursive letters, the state-of-the-art handwritten word recognition approaches have achieved impressively high performances of around 72.11%-74.9% on very large lexicons (10,000 to 100,000 words) [GLF+09].

Artificial Neural Networks (ANNs) and Support Vector Machines (SVMs) have been the most successful classification paradigms for isolated character recognition. In benchmarking tests carried out on standard databases of handwritten letters and digits, the highest recognition rates are often achieved by variations of ANNs and SVMs, closely followed by Hidden Markov Models (HMMs) [CSSJ09, MCS06]. Which classifier is the best for a specific application depends on the nature of the data under consideration, and other factors including preprocessing of the data, representation of the data in the feature space, etc. New theoretical insights indicate that some of the best-established classification paradigms such as SVMs and Radial Basis Function (RBF) ANNs can be formulated in a way to have exactly the same or highly similar functionality [FKSO10]. Then, the difference will only be between the optimization (learning) algorithms used to find the parameters for a specific classifier. However, according to the No Free Lunch (NFL) theorem, there is no single learning algorithm that works best on all supervised learning problems [DHS00, HP02]. A recent research done a standard database of

handwritten digits [CMGS10] can be served as tangible evidence of the validity of the NFL theory in the realm of character recognition. It is shown that plain multi-layer perceptrons trained with the traditional back-propagation algorithm achieve one of the highest recognition rates ever reported for isolated handwritten digits<sup>1</sup>.

In the light of the NFL theorem, perhaps the most straightforward way of overcoming the inherent weaknesses of single classification approaches is to combine them. Ensemble methods in statistics and machine learning refer to the methods of combining several weak classifiers in an attempt to make a stronger classifier. It has been empirically shown that in many classification problems an ensemble of classifiers tend to yield better results than only one classifier [Rok10].

In this chapter we present our approach to cursive character recognition which is based on input perturbation and classifier combination. For each input pattern, firstly, we generate a few versions of it slightly different in shape, and then we recognize each one by an ensemble of neural networks. The idea behind the input perturbation is to make the classification more robust by submitting several slightly distorted versions of an input pattern along with the original pattern to a classifier and then combining the outputs. Ideally, from the classifier point of view, a slightly distorted version of a pattern is the same as the original pattern, and we should not gain any advantage by combining the classification results. However, in practice neither the process of feature extraction nor the classification is perfect. It has been shown that the perturbation method can improve the classification performance in challenging problems such as digit recognition [HB97]. In this chapter, we show that a considerably more challenging problem (56 vs. 10 classes) can benefit from the same method. To the best of our knowledge this is the first time that

---

<sup>1</sup> A strikingly low error rate of 0.35% on the MNIST database.

the perturbation method is applied to cursive character recognition and it is proved to be successful in increasing the classification performance. We perform the classification using a classifier combination method based on a modified Borda count. Furthermore, we devise a new weight update formula in order to counteract the tendency of the neural networks to over-fit the training data. Our experimental results show that the ensemble of neural networks trained with the new weight update formula in conjunction with the input perturbation improves the recognition rate for handwritten cursive characters.

## **6.2 Artificial Neural Network for Handwritten Character Recognition**

Our character recognition engine is based upon a feed forward neural network with an enhanced training method which dynamically penalizes the weights of the neurons in order to improve the generalization performance. In the following, firstly we will briefly describe the feature set that we extracted from characters and then the training mechanism.

### **6.2.1. Feature Extraction**

We extracted 363 features from each character image. These features include basic geometrical features, horizontal and vertical histogram features, Zernike moments [HK06], Fourier features, chain codes, and local gradient histogram features [RSP09a] extracted from different zones of the input image.

### 6.2.2. Training

The architecture of our neural network was a 3-layer feed-forward with 363 neurons in the input layer, 130 neurons in the first hidden layer, 50 neurons in the second hidden layer, and 26 output neurons. The activation function for each neuron was a sigmoid ranging from 0 to 1. We used the back-propagation learning algorithm with momentum and regularization which we implemented by a weight penalization scheme.

In back-propagation learning with momentum, the weight update formula for each weight is defined as below:

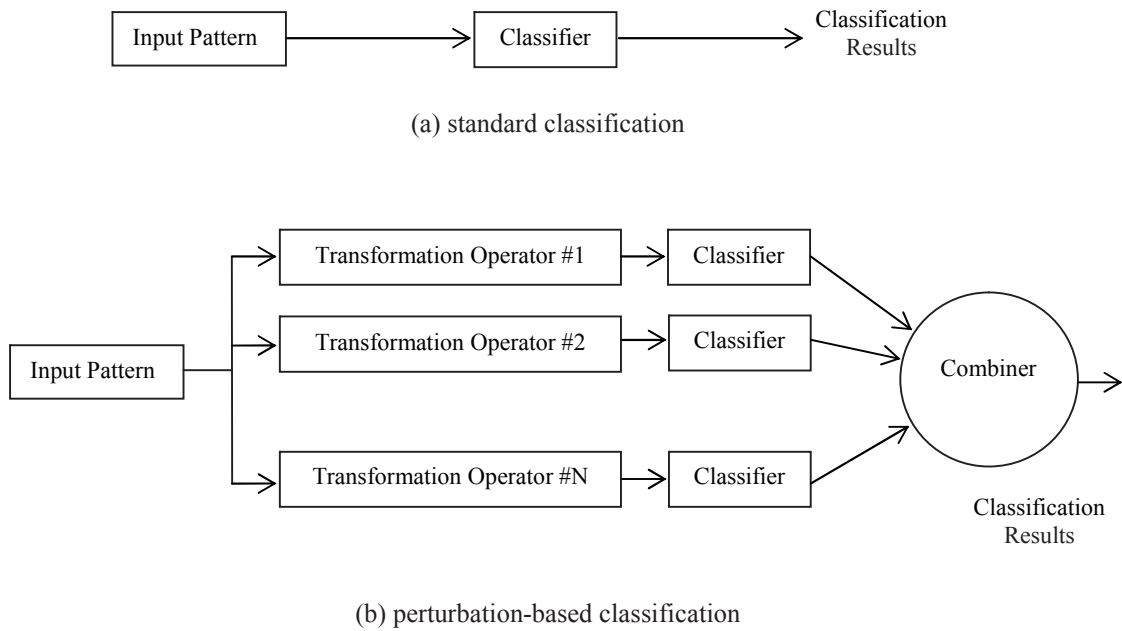
$$\Delta w_i(t) = -\rho \cdot \frac{\partial E}{\partial w_i(t)} + \alpha \Delta w_i(t-1) \quad (6.1)$$

Where  $E$  is the error, and  $\rho$  and  $\alpha$  are the learning rates.

It is well know that in order to guarantee good generalization ability, the number of degrees of freedom or the number of weights must be considerably smaller than the amount of information available for training. Regularization is common method for avoiding over-training or improving generalization ability. We implemented a regularization strategy by the so called weight decay scheme. We added a weight penalization term to the weight update rule (Equation 6.1) which led to the following weight-update rule:

$$\Delta w_i(t) = -\rho \cdot \frac{\partial E}{\partial w_i(t)} + \alpha \Delta w_i(t-1) - \lambda \frac{1}{1 + \frac{1}{N_j - 1} \sum_{i=0}^{N_j-1} |w_{ji}|} \quad (6.2)$$

Therefore, we penalized each weight by an amount which is related to the sum of weights of the connections which are going to the same neuron. Our experiments performed on an unseen data verify that this penalization scheme improves the recognition performance.



**Figure 6.3 Block diagram of perturbation-based classification versus standard classification.**

### 6.3 Perturbation Method for Character Recognition

Perturbation method is a way of boosting performance in classifiers [HB97, VB08]. Based on the assumption that an input pattern is distorted by a certain set of geometrical transformations, the perturbation method reduces the effect of distortion by classification of distorted versions of the input pattern. We chose a set of geometric transformations, such as rotation, slant, erosion, dilation, etc. Ideally, this set must contain all the possible transformations that may deform an input pattern. In order to classify an input pattern, we apply all the geometric transformations in the set to the pattern and then classify each distorted version separately, and finally combine the result of classifications. The

combination of classifier can be done by a fusion technique [BB08] such as majority voting.

A high-level block diagram of the perturbation method versus standard classification is given in Fig. 6.3. Note that in the basic perturbation method, the same classifier is used for all distorted (actually, anti-distorted) versions of the input pattern. However, in general we can use different classifiers in combination with different sets of transformation operators. The block diagram of the general perturbation-based classification is shown in Fig. 6.4.

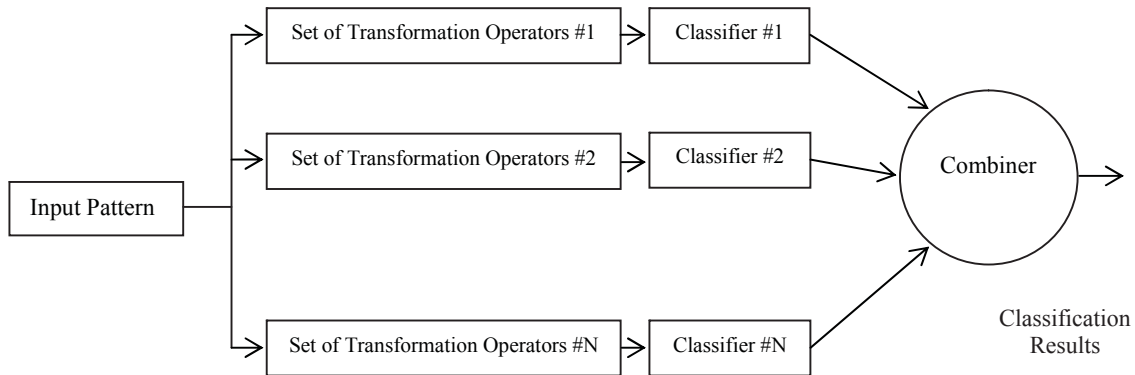
### **6.3.1 Transformation Operators**

In the current implementation, we have used eight transformation operators as listed below.

1. Identity transformation (returns the original pattern).
2. Rotation to the right by a random angle between  $1^\circ$  to  $3^\circ$ .
3. Rotation to the right by a random angle between  $4^\circ$  to  $6^\circ$ .
4. Rotation to the left by a random angle between  $1^\circ$  to  $3^\circ$ .
5. Rotation to the left by a random angle between  $4^\circ$  to  $6^\circ$ .
6. Stroke width normalization.
7. Horizontal dilation by a  $1 \times 3$  structuring element.
8. Vertical dilation by a  $3 \times 1$  structuring element.

Stroke width normalization is done by computing the skeleton of the pattern and the dilating it by a  $3 \times 3$  structuring element. Based on our experiments which will be

summarized at the end of this section, this set of transformation consistently results in a gain in recognition performance. However, it could be interesting to experiment with some other geometrical transforms such as shrink, perspective and slant.



**Figure 6.4 Block diagram of general perturbation-based classification.**

### 6.3.2 Combination of Classifiers

There are several different approaches to the combination of classifiers hypotheses [VGC01]. Borda count is one of the most popular methods of combining rankings, thanks to its simplicity and effectiveness. Several variants of Borda count have been proposed in the pattern recognition community [vES00]. In our perturbation-based recognition approach, we utilize the modified Borda count proposed in [VGC01]. In [VGC01] the authors showed the effectiveness of the modified Borda count in word recognition. However, here we apply the method at character level. Our experimental results show that the modified Borda count as a method of combining character classifiers, improves the overall recognition rate at word level. A summary of the modified Borda count is given below.

### 6.3.2.1 Modified Borda Count

The Borda count is a rank-based election method. In classifiers combination, we can consider each classifier as a voter, and each class as a candidate. Therefore, each classifier provides a ranking of classes, assuming that we use probabilistic or ranked classifiers. In the conventional Borda count, the winner is determined by giving each candidate a certain number of points corresponding to the position where it is ranked by each voter. Once all votes have been counted the candidate with the most points is the winner. The main advantage of conventional Borda count is that no voter can dominate the final decision. However, in classification problems the major disadvantage of the conventional Borda is that it ignores the confidence scores produced by different classifiers.

In order to overcome the disadvantage of the conventional method, the modified Borda adds three components to the conventional decision making process as follows:

- 1) The rank of a candidate is a percentage which is determined by the rank of the candidate among the top  $N$  candidates. Whereas in the conventional Borda, the rank of a candidate is the number of candidates that stand below it. The percentage-based rank in the modified Borda is calculated as follows:

$$\text{Rank}(C) = \begin{cases} 1 - \frac{(\text{position of } C \text{ in top } N \text{ candidates})}{N} & \text{if } C \text{ is among the top } N \text{ candidates} \\ 0 & \text{otherwise} \end{cases} \quad (6.3)$$



Where  $C$  denotes a candidate (i.e. class), and position of  $C$  is a zero-based index. Therefore, for  $N = 5$  for example, the rank of the first candidate is 1, the rank of the second candidate is  $1 - 1 / 5 = 0.8$ , and so on.

- 2) The percentage-based rank of a candidate is further adjusted by the confidence score that is assigned to the candidate by a voter (i.e. classifier). Let  $CS_C$  denote the confidence score assigned to the candidate  $C$ . Then, assuming that the confidence score has a value in the range  $[0, 1]$ , we simply adjust the percentage-based rank as follows:

$$\text{Rank}_{CA}(C) = \text{Rank}(C) \times CS_C. \quad (6.4)$$

- 3) The confidence-adjusted rank (i.e.  $\text{Rank}_{CS}$ ) that comes from each voter is further modified by a degree of credibility of the voter. The degree of credibility has a similar effect to the weight parameter in the weighted Borda count. In the simplest form, we can take the recognition rate of a classifier (computed on a validation set) as its degree of credibility. Then, we adjust the confidence-adjusted rank in order to obtain the total rank of a candidate as follows:

$$\text{Rank}_{Total}(C) = \text{Rank}_{CA}(C) \times D_{cr} \quad (6.5)$$

Where  $D_{cr}$  denotes the degree of credibility of the voter.

The result of the election is obtained by adding up the total ranks that each candidate receives from all the voters.

## 6.4 Experimental Results

We verified the effectiveness of our character recognition approach by performing experiments on a dataset of handwritten characters that is composed of isolated characters and cursive characters. We took the isolated characters from the standard NIST SD 19 database [GBC+94], and we generated the cursive characters by applying our character segmentation algorithm to the IAM database of handwritten words [MB02] and manually removing under-segmented and over-segmented characters.

We chose 930 samples for each class of character, and used 2/3 for training and the remaining 1/3 for testing. The division of the data into training and test parts was based on a random sampling. We carried out our experiments on 5 different random divisions of the data into training and test parts. The results reported in the following are the average of these 5 sets of experiments.

**Table 6.1 Correct classification rate for database of handwritten characters**

<b>Classification Rate</b>	Baseline Classification (1 neural network)	Input perturbation (8 transforms)	Classifier Combination (ensemble of 5 NNs)	Input perturbation + Classifier Combination
with weight decay	82.31	87.23	88.15	92.71
w/out weight decay	81.40	87.23	87.62	92.63

We obtained the highest classification rate when we used the input perturbation method (8 transforms) in conjunction with the classifier combination (5 neural networks). The classification rate in this case was around 92.7%, which was higher than when we only

used the perturbation or the classifier combination alone. Table 6.1 summarizes these results. As we can see in all cases, the classification rate that we obtain by either the perturbation method or the ensemble or both is considerably higher than the baseline classification (only one neural network without input perturbation). We can also see that the weight decay formula slightly improves the classification rate, especially when used in isolation.

## Chapter 7

### *Generalized Minimum Edit Distance for Handwritten Words*

#### **7.1 Introduction**

For the classification of document images based on arbitrary text queries, as we mentioned earlier, there are three general strategies. The first strategy is to transcribe the document image into text and then apply information retrieval techniques in the text domain. This approach is not efficient because the performance of the existing recognition techniques is not adequate for unconstrained handwritten documents [CBG09]. The second strategy is based on template matching methods by which we can compute the distance between images representing words. However, for this strategy to be applicable and effective, we need to have a set of handwritten images, at least 10 to 50 according to [vdZSH08], with different writing styles corresponding to each query word. Obviously, to collect a database of handwritten words for all possible query words is not feasible. The third strategy is based on analytical recognition methods that are the best suited for our application.

The main advantage of analytical recognition methods is their ability to recognize words based on character models, thereby obviating the need for having a database of handwritten words. In our application, we are interested to know how far a word image (i.e. an image representing a word) is from a text keyword. In other words, we wish to

find a distance function between word images and text keywords. For this purpose, we use a variation of the edit distance (a.k.a. the Levenshtein distance). The edit distance is a widely used measure of string similarity which was originally proposed for character strings with applications in spelling correction [Dam64]. However, since then many different variations of the basic edit distance have been proposed and applied to various problems including DNA analysis, fraud detection, pattern recognition etc. [DT10, OS06, Wei04, SM02, SKS96].

In the following, firstly we will briefly explain the classical edit distance where both sequences are character strings. Secondly, we will describe the extension of the edit distance for the case where one sequence is a character string and the other sequence is an image. Thirdly, we will show how to model the proposed edit distance by a Hidden Markov Model (HMM). Consequently, we will show that the costs for the edit operations can be learnt using the Expectation Maximization (EM) algorithm. Fourthly, we will present how to incorporate a priori knowledge into the edit distance using HMMs.

## 7.2 Classical Minimum Edit Distance

Let  $\Sigma$  be a finite alphabet and  $\Sigma^*$  be the set of all finite strings over  $\Sigma$ . Let  $x = x_1x_2\dots x_n$  and  $y = y_1y_2\dots y_m$  be two arbitrary strings of  $\Sigma^*$  of length  $n = |x|$  and  $m = |y|$  respectively. Let  $R^+$  be the set of nonnegative real numbers.

A string distance between  $x$  and  $y$  is characterized by a pair  $(\Sigma, cost_e)$  where  $cost_e : E \rightarrow R^+$  is the primitive cost function, and  $E = E_{substitute} \cup E_{insert} \cup E_{delete}$  is the alphabet of primitive edit operations.  $E_{substitute} = \Sigma \times \Sigma$  is the set of substitutions,  $E_{insert} = \{\epsilon\} \times \Sigma$  is the set of insertions, and  $E_{delete} = \Sigma \times \{\epsilon\}$  is the set of deletions. Each such pair  $(\Sigma, cost_e)$

induces a distance function  $d: \Sigma^* \times \Sigma^* \rightarrow R^+$  that maps a pair of strings to a nonnegative real value. The minimum edit distance  $d(x,y)$  between two strings  $x \in \Sigma^*$  and  $y \in \Sigma^*$  is defined by the following recursive equation:

$$d(x, y) = \min \begin{cases} \text{cost}_e(a, b) + d(x'x'', y'y'') & \text{where } x = x'ax'' \text{ and } y = y'by'' \\ \text{cost}_e(a, \varepsilon) + d(x'x'', y) & \text{where } x = x'ax'' \\ \text{cost}_e(\varepsilon, b) + d(x, y'y'') & \text{where } y = y'by'' \end{cases} \quad (7.1)$$

It should be noted that in the original version of the edit distance proposed by Levenshtein the cost of substitution, insertion and deletion is 1. However, as we see in Equation (7.1) these costs can be modeled by a function and they do not need to be the same. The calculation of Equation (7.1) can be done using dynamic programming in  $O(mn)$  time and space [WL75]. However, depending on the application, the distance can be calculated in a shorter time. For example, if we know that the distance between the two strings is small, then using lazy evaluation the equation can be calculated in  $O(m \cdot (1 + d))$  time, where  $d$  is the minimum edit distance [All92].

### 7.3 Generalized Minimum Edit Distance

In general, the alphabets that the two strings are defined on do not need to be the same. That is, we can define the minimum edit distance for two arbitrary strings  $x = x_1x_2\dots x_n$  and  $y = y_1y_2\dots y_m$  where  $x_i \in \Sigma$  for  $1 \leq i \leq n$  and  $y_j \in \Psi$  for  $1 \leq j \leq m$ .

In our application, we are interested in defining the distance between a sequence of characters  $x$  and a sequence of image regions  $y$  that is the output of that character segmentation algorithm. Therefore, an edit distance between the two sequences  $x$  and  $y$  is

characterized by a 4-tuple  $(\Sigma, \Psi, \mathcal{N}, cost_e)$  where  $\Sigma$  denotes the set of characters,  $\Psi$  denotes the set of image regions,  $\mathcal{N}$  is the neighborhood graph for the regions, and  $cost_e : E \rightarrow R^+$  is the primitive cost function which maps a primitive edit operation  $e \in E$  to a real value.

As discussed earlier, the character segmentation algorithm has to over-segment certain characters without using the context knowledge. In order to handle over-segmentation, we add a set of merging operations to the set of basic edit operations. As we know that the character segmentation algorithm may over-segment a character into up to three regions, we only need to define two merging operations, where one merges two neighboring regions, and the other one merges three neighboring regions together.

Therefore, we define the alphabet of primitive edit operations as follows:

$$E = E_{c|\varepsilon} \cup E_{\varepsilon|r} \cup E_{c|r} \cup E_{c|rr} \cup E_{c|rrr} \text{ where}$$

$$E_{c|\varepsilon} = \Sigma \times \{\varepsilon\} \text{ is the set of character insertions;}$$

$$E_{\varepsilon|r} = \{\varepsilon\} \times \Sigma \text{ is the set of region insertions;}$$

$$E_{c|r} = \Sigma \times \Psi \text{ is the set of substitutions of regions by characters;}$$

$$E_{c|rr} = \Sigma \times \Psi \text{ is the set of substitutions of 2-tuple of neighboring regions by characters;}$$

and

$$E_{c|rrr} = \Sigma \times \Psi \text{ is the set of substitutions of 3-tuple of neighboring regions by characters.}$$

Using these primitive edit operations, one can transform a sequence of characters to a sequence of image regions by either inserting a character, or inserting a region, or replacing a character by a region, or replacing a character by two neighboring regions, or replacing a character by three neighboring regions, and combinations of these operations.

The generalized minimum edit distance  $d(x,y)$  between a sequence of characters  $x \in \Sigma^*$  and a sequence of image regions  $y \in \Psi^*$  is defined by the following recursive equation:

$$d(x,y) = \min \begin{cases} \text{cost}_e(a, \varepsilon) + d(x'x'', y) & \text{where } x = x'ax'' \\ \text{cost}_e(\varepsilon, r) + d(x, y'y'') & \text{where } y = y'ry'' \\ \text{cost}_e(a, r) + d(x'x'', y'y'') & \text{where } x = x'ax'' \text{ and } y = y'ry'' \\ \text{cost}_e(a, rs) + d(x'x'', y'y'') & \text{where } x = x'ax'' \text{ and } y = y'rsy'' \\ \text{cost}_e(a, rst) + d(x'x'', y'y'') & \text{where } x = x'ax'' \text{ and } y = y'rsty'' \end{cases} \quad (7.2)$$

where characters are indicated by  $a$  and  $b$ , sequences of characters are indicated by  $x'$  and  $x''$ , image regions are indicated by  $r$ ,  $s$ , and  $t$ , and sequences of image regions are indicated by  $y'$  and  $y''$ .

### 7.3.1 Default Cost Functions

In the original version of the edit distance for character strings, the default cost function is 1, i.e. the cost of inserting a character, deleting a character or substituting a character by another character is 1. In the generalized edit distance which is defined between a sequence of characters and a sequence of image, we define the default cost functions in a similar way. We set the default cost of inserting a region equal to the default cost of inserting a character equal to 1. However, for the substitution operations, we obtain the cost by the ensemble of neural networks. Let's denote the ensemble of neural networks by  $\Omega$ . Assuming the we use probabilistic classifiers, the process of feature extraction, recognition and voting can be modeled by a function that maps a pair of region and character to a real number in the range  $[0, 1]$ , that is:  $\Omega: \Psi \times \Sigma \rightarrow [0, 1]$ , where  $\Psi$  is the set of image regions, and  $\Sigma$  is the set of characters.



Therefore, in order to determine the cost of substituting a region  $r \in \Psi$  by character  $c_i \in \Sigma$ , we recognize the region by the ensemble of neural networks and set the cost as follows:

$$\text{cost}_e(c_i, r) = 1 - \Omega(r, c_i) \quad (7.3)$$

Therefore, ideally when the region represents the character the cost is 0, and otherwise the cost is 1. The recursive definition of the generalized edit distance (Equation (7.2)) based on the default cost functions is rewritten as follows:

$$d(x, y) = \min \begin{cases} 1 + d(x'x'', y) & \text{where } x = x'ax'' \\ 1 + d(x, y'y'') & \text{where } y = y'ry'' \\ 1 - \Omega(r, a) + d(x'x'', y'y'') & \text{where } x = x'ax'' \text{ and } y = y'ry'' \\ 1 - \Omega(rs, a) + d(x'x'', y'y'') & \text{where } x = x'ax'' \text{ and } y = y'rsy'' \\ 1 - \Omega(rst, a) + d(x'x'', y'y'') & \text{where } x = x'ax'' \text{ and } y = y'rsty'' \end{cases} \quad (7.4)$$

## 7.4 Modeling Generalized Minimum Edit Distance Using HMMs

In this section we will show how to model the generalized edit distance that we introduced in the previous section by a HMM. The advantage of modeling the distance by using HMMs is twofold. First, we have a straightforward way to incorporate domain knowledge into the model. Second, we can learn the cost functions using training data.

In the following, we will briefly introduce the terminology of HMMs and their three fundamental problems, namely likelihood, decoding and learning. We will discuss the solution to each problem and how it is related to our application.

### 7.4.1 Hidden Markov Models

A HMM is a statistical tool to model a system that is assumed to be a Markov chain with unobserved (i.e. hidden) states. A Markov chain is a random process for which the Markov property<sup>2</sup> holds and the number of states that the process can be in is finite or countable. Therefore, a HMM can actually be considered as a nondeterministic Finite State Machine (FSM) where each state is associated with a random function. Within a discrete period of time  $t$ , the model is assumed to be in some state and generates an observation by a random function of the state. Based on the transition probability of the current state, the underlying Markov chain changes to another state at time  $t+1$ . The state sequence that the model passes through is unknown, only some probabilistic function of the state sequence that is the observations produced by the random function of each state can be seen. A HMM is characterized by the following elements:

$$N: \text{The number of states of the model} \quad (7.5)$$

$$S = \{s_1, s_2, \dots, s_N\}: \text{The set of states} \quad (7.6)$$

$$\Pi = \{ \pi_i = P(s_i \text{ at } t = 1) \}: \text{The initial state probabilities} \quad (7.7)$$

$$A = \{a_{ij} = P(s_j \text{ at } t+1 \mid s_i \text{ at } t)\}: \text{The state transition probabilities} \quad (7.8)$$

$$M: \text{The number of observation symbols} \quad (7.9)$$

---

<sup>2</sup> In simple terms, the Markov property states that the next state depends only on the current state and not on the past.

$$V = \{v_1, v_2, \dots, v_M\}: \text{The set of possible observation symbols} \quad (7.10)$$

$$B = \{b_i(v_k) = P(v_k \text{ at } t \mid s_i \text{ at } t)\}: \text{The symbol emission probabilities} \quad (7.11)$$

$$O_t \in V: \text{The observed symbol at time } t \quad (7.12)$$

$$T: \text{The length of observation sequence} \quad (7.13)$$

$$\lambda = (A, B, \Pi): \text{The compact notation to denote the HMM.} \quad (7.14)$$

With the following three constraints on initial probabilities, transition probabilities and observation probabilities:

$$\sum_{i=1}^N \pi_i = 1 \quad (7.15)$$

$$\sum_{j=1}^N a_{ij} = 1, \quad \forall i \quad (7.16)$$

$$\sum_{k=1}^M b_i(v_k) = 1, \quad \forall i \quad (7.17)$$

#### **7.4.1.1 Three Fundamental Problems for HMMs**

Most applications of HMMs need to solve the following problems:

*Problem 1: Likelihood* - Given a model  $\lambda = (A, B, \Pi)$ , how do we efficiently compute  $P(O | \lambda)$ , that is the probability of occurrence of the observation sequence  $O = O_1, O_2, \dots, O_T$ .

*Problem 2: Decoding* - Given the observation sequence  $O$  and a model  $\lambda$ , how do we choose a state sequence  $S = s_1, s_2, \dots, s_T$  so that  $P(O, S | \lambda)$ , the joint probability of the observation sequence  $O = O_1, O_2, \dots, O_T$  and the state sequence  $S = s_1, s_2, \dots, s_T$  given the model, is maximized. In other words, we want to find a state sequence that best explains the observation.

*Problem 3: Training* - Given the observation sequence  $O$ , how do we adjust the model parameters  $\lambda = (A, B, \Pi)$  so that  $P(O | \lambda)$  or  $P(O, S | \lambda)$  is maximized. In other words, we want to find a model that best explains the observed data.

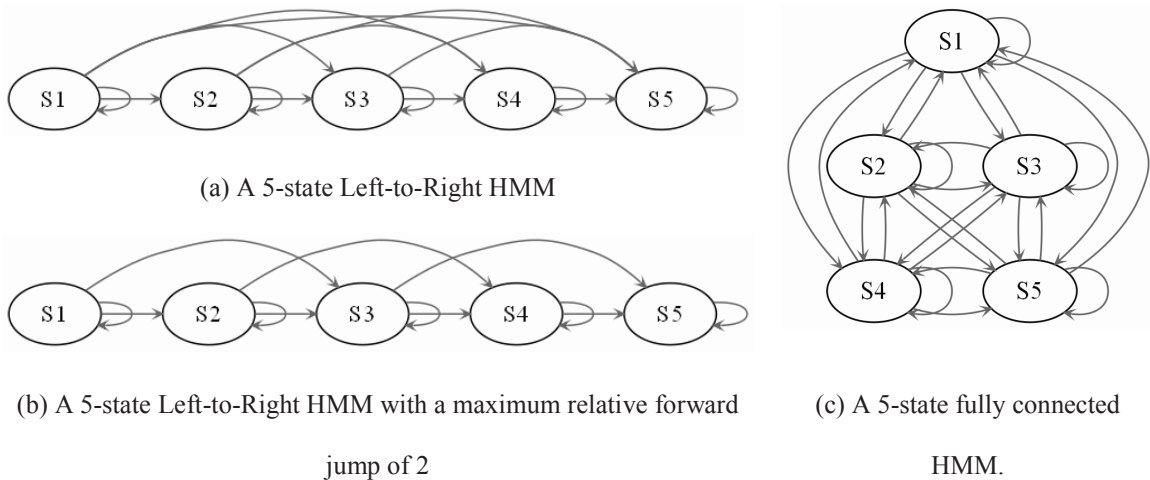
The solution to the likelihood problem is given by the so-called forward or the backward algorithm. The solution to the decoding problem is given by the Viterbi algorithm, and the solution to the learning problem is given by the segmental  $K$ -means or Baum-Welch algorithm [Rab89].

#### **7.4.1.2 Topologies of HMMs**

The structure of the state transition matrix  $A$  determines the topology of the HMM. Through the use of topologies we can incorporate domain knowledge in the HMM. In classification, the topology of the HMM is a determining factor in performance of the system [AMCS04]. One of the most widely used topologies in speech/text recognition is the so called Left-to-Right (LR) or Bakis model in which lower numbered states account for observations occurring prior to higher numbered states. The temporal order in LR-

HMMs is imposed by introducing structural zeros to the model in the form of the constraint  $\prod = \{1, 0, \dots, 0\}$  and  $a_{ij} = 0, i > j$  meaning that the model begins at the first (i.e. left most) state and at each time instant it can only proceed to the same or a higher numbered state. As a further constraint, in LR-HMM the number of forward jumps at each state is usually limited in order to restrict large state changes, i.e.  $a_{ij} = 0, j > i + \Delta$  for some fixed  $\Delta$ .

Figure 7.1 shows two LR-HMMs, one with limited maximum forward jumps and the other one without, versus a fully-connected HMM where each state in the model is reachable from any state within one transition. Fully-connected HMMs are also known as ergodic HMMs.



**Figure 7.1** Examples of HMMs with (a and b) and without (c) topological constraints.

LR topologies are the most straightforward models for 1D temporal signals such as speech. However, the image data is represented by a 2D matrix, where the temporal information is lost. The typical sliding window approach, where a narrow window is moved on the image from left to right (or vice versa), aims at recovering the temporal information from the 2D matrix representing the handwriting. Of course, when the

handwriting is written cursively with a considerable amount of slant and overlapping between neighboring characters, the sliding window approach cannot provide a good 1D representation for the underlying 2D signal. In order to obviate this problem, multi-stream HMMs [KPBH10], 2D-HMMs and their variations [KA94, LNG00, CK04] have been proposed.

2D HMMs are natural extensions of traditional HMMs for 2D signals. However, it can be shown that when a 2D-HMM is modeled by an equivalent 1D-HMM, the number of states is exponential [MMM00], which means that the order of the decoding and learning algorithms is not polynomial anymore, but exponential. In order to reduce the complexity of 2D-HMMs, some authors have proposed topologies that are not fully-connected but rather composed of loosely-coupled super-states<sup>3</sup>. Each super-state is usually a LR-HMM, and the complete model is formed by linking these super-states. These models are called Pseudo 2D-HMMs (P2D-HMMs) [KA94, CK04]. Given that the number of connections between the inner states of a super-state and the inner states of another super-state is zero or few, the order of the states required for the P2D-HMM is polynomial. For modeling images, a typical approach based on P2D-HMMs is to model each row of the image by one super-state, which is based on the unrealistic assumption that the states sequence in each row is independent of the states sequences of its neighboring rows. The reduction in the complexity of P2D-HMMs is obtained at the cost of over-simplifying the model which is sometimes based on unrealistic assumptions.

The HMM model that we will propose in the next section is a Generalized HMM (GHMM) with an ergodic topology. The main property of the GHMM that we will utilize

---

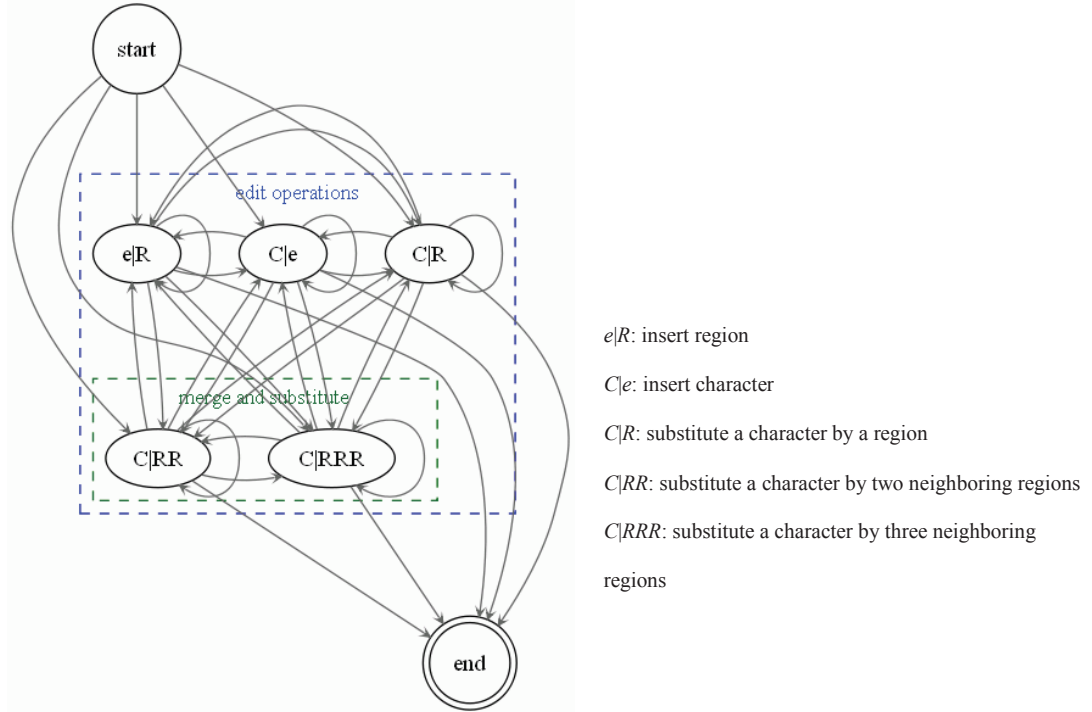
<sup>3</sup> A system is said to be a loosely coupled when each of its components has little or no knowledge of the definitions of other separate components.

is the relaxation of the additivity constraint of probability measures [MG00]. The advantage of our Generalized Ergodic HMM (GEHMM) over P2D-HMMs is to provide an exact model for the temporal information present in the handwriting with a feasible number of states.

#### 7.4.2 Modeling Generalized Minimum Edit Distance Using GEHMMs

The direct extension of minimum edit distance or Dynamic Time Warping (DTW) methods to images when applied at pixel level is not efficient. This is due to the fact that the underlying Markov models are 2D-HMMs or P2D-HMMs which are either non-practical or over-simplified in general.

In our proposed approach, the observation sequence is not image pixels, but rather image regions that correspond to characters or sub-characters. Thus, we can build a model whose states logically correspond to the edit operations (insertion, substitution and merging). Consequently, the number of states will be constant and small. Fig. 7.2 shows the HMM corresponding to the generalized minimum edit distance defined by Equation (7.2), where the five edit operations are models by five states:  $S_{edit} = \{s_1, s_2, s_3, s_4, s_5\}$  with  $s_1 := e|R$ ,  $s_2 := C|e$ ,  $s_3 := C|R$ ,  $s_4 := C|RR$  and  $s_5 := C|RRR$ .



**Figure 7.2** GEHMM corresponding to the generalized minimum edit distance defined by Equation (7.2)

The output alphabet in this model is the set of image regions that is  $O = R = \{f_i(x, y) \mid f_i: \text{characteristic function corresponding to region } r_i\}$ . We have assumed that the input image is binary; therefore we can represent a region by a characteristic function. Formally, for a binary image with  $M$  rows and  $N$  columns, an arbitrary image region  $r_i$  is denoted by the characteristic function  $f_i: X \times Y \rightarrow \{0, 1\}$  where  $x \in X, y \in Y, X := \{0, 1, \dots, M-1\}$  and  $Y := \{0, 1, \dots, N-1\}$ , and  $f_i(x, y)$  is 1 if  $(x, y) \in r_i$ , and 0 otherwise.

#### 7.4.2.1 Initial and Transition Probabilities

In the beginning, we can use any edit operation equally likely, thus the initial probabilities are the same:  $\pi_i = 1/5, 1 \leq i \leq 5$ .



Then, we can use any edit operation equally likely, thus the transition probabilities in each and every state are the same:  $P(s_i | s_j) = 1/6$ ,  $1 \leq i, j \leq 5$ . Note that from each edit state we can go to the final state (“end”), therefore we have  $5 + 1 = 6$  in the denominator.

#### 7.4.2.2 Observation Probabilities

In the substitution state ( $s_3 := C|R$ ), the probability of observing a region  $r_i$  is the maximum probability that a character can describe  $r_i$ , which is determined by the ensemble of neural networks:

$$P(\text{observing } r_i \text{ as a single character} | s = s_3) = \max_c \Omega(r_i, c), c \in C \quad (7.18)$$

Obviously, we are not only interested in computing the distance but also in recognizing the image, thus we keep the character that best describes the region as well (i.e.  $\arg \max_c \Omega(r_i, c)$ ).

The probability of observing 2-tuples of neighboring regions in  $s_4 := C|RR$ , and the probability of observing 3-tuples of neighboring regions in  $s_5 := C|RRR$ , is defined in a similar way. However, we also have to take the neighborhood relations between regions into account. In state  $s_4$ , the probability of observing region  $r_i$  and region  $r_j$  as a single character is defined as follows:

$$P(\text{observing } r_i \cup r_j \text{ as a single character} | s = s_4) = \max \{P(r_i \text{ and } r_j \text{ being neighbor and mergeable}) \cdot \Omega(r_i \cup r_j, c)\}, c \in C \quad (7.19)$$

We calculate the probability of two regions being neighbor and mergeable as follows:

$$\begin{aligned}
 P( r_i \text{ and } r_j \text{ being neighbor and mergeable } ) = \\
 P( r_i \text{ and } r_j \text{ being mergeable } | r_i \text{ and } r_j \text{ being neighbor } ) \cdot \\
 P( r_i \text{ and } r_j \text{ being neighbor } ).
 \end{aligned} \tag{7.20}$$

The probability of two regions being neighbors is defined by the neighborhood graph  $\mathcal{N}$ :

$$P( r_i \text{ and } r_j \text{ being neighbor } ) = N(i, j) \tag{7.21}$$

Where  $N$  is the weighted adjacency matrix corresponding to  $\mathcal{N}$ . Note that, in general, the neighborhood graph is a weighted graph with weights between 0 and 1. This allows for any two regions to be considered neighbors with a degree of truth between 0 and 1, rather than being either neighbor or not neighbor.

Assuming that the ensemble of classifiers is able to reject an input pattern that does not belong to any classes, we can merge any two regions given that they are neighbors. That is:

$$P( r_i \text{ and } r_j \text{ being mergeable } | r_i \text{ and } r_j \text{ being neighbor } ) = 1 \tag{7.22}$$

Therefore, we rewrite Equation (7.19) as follows:

$$P( \text{observing } r_i \cup r_j \text{ as a single character } | s = s_4 ) =$$

$$\max \{ N(i, j) \cdot \Omega(r_i \cup r_j, c) \}, c \in C \quad (7.23)$$

In state  $s_5$ , the probability of observing regions  $r_i$ ,  $r_j$  and  $r_k$  as a single character, assuming that any three regions are mergeable given that they are neighbors, is similarly calculated as follows:

$$\begin{aligned} & P(\text{observing } r_i \cup r_j \cup r_k \text{ as a single character} \mid s = s_5) = \\ & \max \{ P(r_i \text{ and } r_j \text{ and } r_k \text{ being neighbor and mergeable}) \cdot \Omega(r_i \cup r_j \cup r_k, c) \} = \\ & \max \{ P(r_i \text{ and } r_j \text{ and } r_k \text{ being neighbor}) \cdot \Omega(r_i \cup r_j \cup r_k, c) \}, c \in C \end{aligned} \quad (7.24)$$

We define the probability of three regions being neighbors in terms of the probability of two regions being neighbors as follows:

$$\begin{aligned} & P(r_i \text{ and } r_j \text{ and } r_k \text{ being neighbors}) = \\ & \max \{ \\ & \quad P(r_i \text{ and } r_j \text{ being neighbors}) \cdot P(r_i \text{ and } r_k \text{ being neighbors}), \\ & \quad P(r_i \text{ and } r_j \text{ being neighbors}) \cdot P(r_j \text{ and } r_k \text{ being neighbors}), \\ & \quad P(r_i \text{ and } r_k \text{ being neighbors}) \cdot P(r_j \text{ and } r_k \text{ being neighbors}) \}. \end{aligned} \quad (7.25)$$

It is straightforward to extend Equation (7.24) to the case of more than three regions if necessary. In general, the probability of  $n$  regions being neighbors, given that the probability of any two pairs of regions being neighbors is known, is a Minimum

Spanning Tree (MST) problem that can be solved by a number of classical algorithms including Kruskal's [KvT05].

The region insertion state ( $s_1 := e|R$ ) is to model regions that do not correspond to any characters. These are extra regions that correspond to background noise, misspellings or parts of characters from upper or lower text lines. By default, we assume that a region is equally likely to be extra or not, that is we set the probability of observing any region in  $s_1$  to 0.5.

Similarly, the character insertion state ( $s_2 := C|e$ ) is to model characters that do not correspond to any regions. This region allows for a handwritten word with some missing characters to be matched with a lexicon entry. A study of common misspellings shows that a double strike is the most likely cause of a missing character; that is where people forget to add the second character of a double character. As the likelihood of a double character occurring in a word is low, we can conclude that the likelihood of a character being absent in a word is much lower than the likelihood of it being present. By default, we set the probability of observing the empty region (denoted by the symbol  $e$ ) in  $s_2$  to 0.1. The probability of observing any non-empty region  $r \in R - \{e\}$  in  $s_2$  is 0. For the purpose of decoding that we will explain in the next section, we keep the inserted character in this state. In this basic model, we assume that all characters are equally likely to be inserted. However, later on we will show that how these likelihoods can be learnt from training data, so for example the insertion of character 'l' is more likely than 'z'.

Note that the definitions of observation probabilities as above requires the model to be a GHMM [MG00] because the observation probabilities in each state do not sum to 1.

### ***7.4.2.3 Decoding: Recognition of Handwritten Words Using the GEHMM Model***

Having defined the initial, transition and observation probabilities, we can use the model to recognize a handwritten word that is represented by a sequence of regions. The transcription of the handwritten word is simply obtained by decoding; i.e. finding the sequence of states that best describes the observation sequence. As mentioned in the previous section, every state corresponds to a character, except for the insert region state ( $s_1 := e|R$ ). We can assume that  $s_1$  corresponds to the empty character. Thus, the transcription of the handwritten word is obtained by concatenating the characters that correspond to the most likely state sequence.

### **7.4.3 Incorporating A Priori Knowledge to GEHMMs for Handwritten Word Recognition**

The GEHMM model that we introduced in the previous section is a versatile tool for the recognition of handwritten words. However, the basic 5-state model of Fig. 7.2 does not have any knowledge about the lexicon. In this section, we will show that through the use of more states, we can incorporate into the model a priori knowledge about the lexicon, spelling errors and noise.

The number of states that we need to represent the a priori knowledge is proportional to the size of the alphabet. The first version that we propose has 159 states, and the more elaborate version has 315 states. Therefore, compared to the basic model, the number of states is considerably higher, however still constant and manageable.

### 7.4.3.1 Adding Knowledge about the Lexicon

Character  $n$ -gram models provide the most straightforward way to incorporate knowledge about the lexicon into a Markov model. A character  $n$ -gram is a subsequence of  $n$  characters from a given sequence of characters. A character  $n$ -gram model is a probabilistic model for predicting the next character in such a sequence. In general,  $n$ -gram models can be used for any sequences from a finite alphabet.  $N$ -gram models have been widely used in statistical natural language processing, compression, speech and handwriting recognition [MS99].

The most widely used  $n$ -gram models are based on the shortest  $n$ -grams ( $n = 1, 2$  and  $3$ ) that are referred to as unigrams ( $n = 1$ ), bigrams ( $n = 2$ ) and trigrams ( $n = 3$ ). In our application, we only use unigram and bigram models. In the following, we will describe how to obtain character unigram and bigram models based on a lexicon.

Formally, the task of predicting the next character can be stated as estimating the probability function  $P$ :

$$P(c_n | c_1, c_2, \dots, c_{n-1}) = \frac{P(c_1, c_2, \dots, c_{n-1}, c_n)}{P(c_1, c_2, \dots, c_{n-1})}, c_i \in C \quad (7.26)$$

In other words, we wish to use the history of the previous items (i.e. characters) to predict the next item.

Let  $Count(c_1 c_2 \dots c_{n-1})$  be the frequency of the sequence  $c_1 c_2 \dots c_{n-1}$ , and  $Count(c_1 c_2 \dots c_n)$  be the frequency of the sequence  $c_1 c_2 \dots c_n$  in the training data (i.e. lexicon). Now, the Maximum Likelihood Estimate (MLE) estimate for the probability of a certain  $n$ -gram  $c_1 c_2 \dots c_n$  is defined as follows:

$$P(c_1, c_2, \dots, c_n) = \frac{\text{Count}(c_1 c_2 \dots c_n)}{N}, c_i \in C \quad (7.27)$$

Where  $N$  is total number of all  $n$ -grams appearing in the training data. The MLE estimate for the conditional probability function  $P$  is defined as follows:

$$P_{\text{MLE}}(c_n | c_1, c_2, \dots, c_{n-1}) = \frac{\text{Count}(c_1 c_2 \dots c_{n-1} c_n)}{\text{Count}(c_1 c_2 \dots c_{n-1})}, c_i \in C \quad (7.28)$$

In particular, using the MLE estimates the character unigram model is defined as follows:

$$P_{\text{MLE}}(c_i) = \frac{\text{number of words in the lexicon starting with character } c_i}{\text{total number of words in the lexicon}}, c_i \in C \quad (7.29)$$

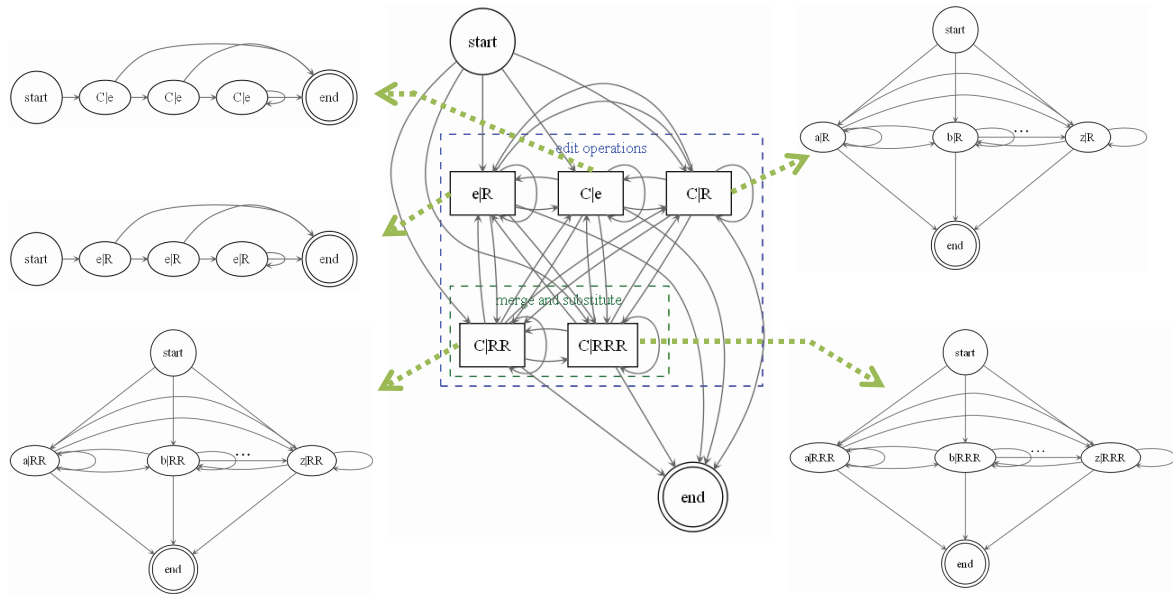
And the character bigram model is defined as follows:

$$P_{\text{MLE}}(c_i | c_j) = \frac{\text{number of occurrences of } c_j c_i \text{ in all words in the lexicon}}{\text{total number of occurrences of } c_j \text{ in all words in the lexicons}}, c_i, c_j \in C \quad (7.30)$$

The unigram model specifies the initial probabilities and the bigram model specifies the transition probabilities in the GEHMM model.

The character unigram model estimates the probability of observing a certain character in the beginning of a word, which is the initial probability of going to a state that represent the character in the GEHMM model. The character bigram model estimates the probability of observing a certain character given that the previous character is known,

which is the transition probability of going from the state that represents the previous character to the state that represents the desired character.

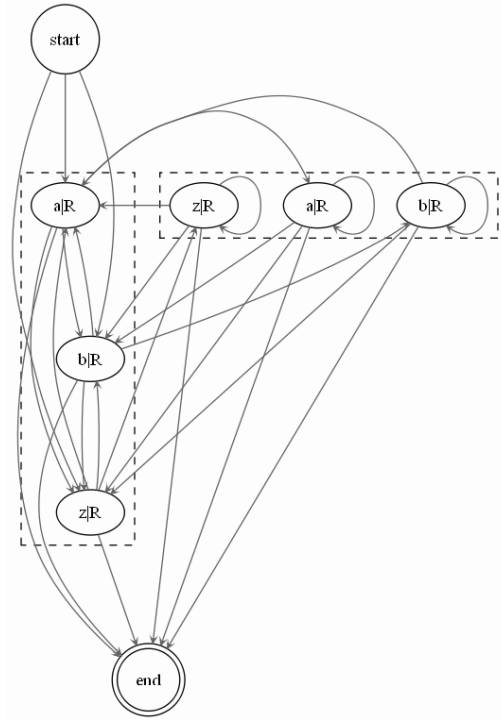


**Figure 7.3 159-state enhanced GEHMM model for word recognition.**

According to the above discussion, in order to include the unigram and bigram models into the GEHMM model, we have to have a separate state for any character. Therefore, each character state ( $s_1$ ,  $s_2$ ,  $s_3$  and  $s_4$ ) in the 5-state model of Figure 7.2 has to be decomposed to 26 states. We also decompose the region insertion state ( $s_1$ ) into a few states which allows for the model to impose a constraint on the number of regions that can be inserted consecutively. We assume that in the process of matching a word with a sequence of image regions, the insertion of 3 regions in a row and the insertion of more than 3 regions in a row are equally unlikely events. Therefore, we decompose  $s_1$  into 3 states. We can impose the same constraint on the character insertion state ( $s_2$ ). Therefore,



the character insertion state ( $s_2$ ) in the 5-state model has to be decomposed into  $3 \times 26 = 78$  states. Fig. 7.3 shows the whole model that is composed of  $6 \times 26 + 3 = 159$  states.



**Figure 7.4** Decomposition of the character substitution state based on the character trigram model.

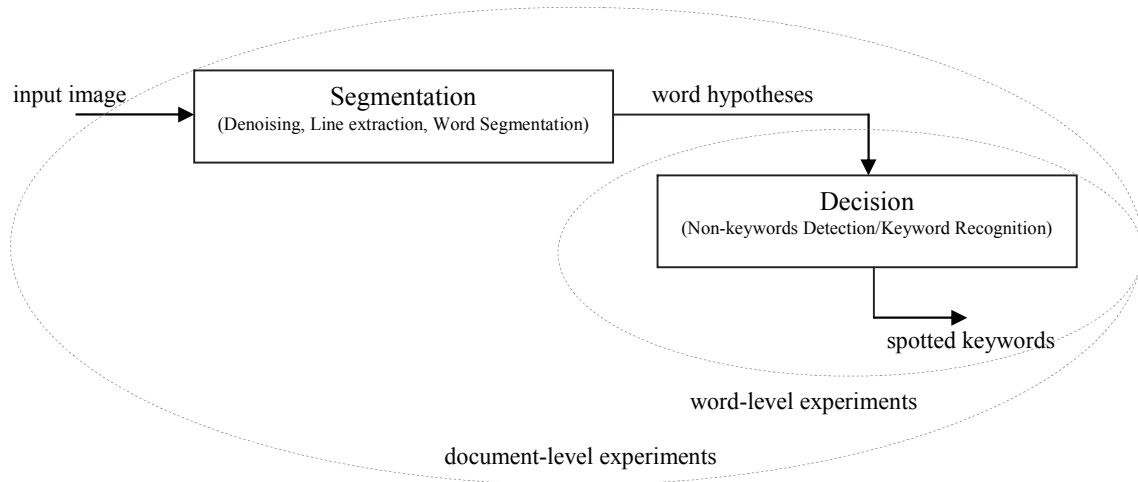
If we wish to incorporate more knowledge about the lexicon into the model, we can use character trigram models at the cost of more states. In order to represent the trigram model, each character state ( $s_1$ ,  $s_2$ ,  $s_3$  and  $s_4$ ) in the 5-state model of Figure 7.2 has to be decomposed to  $2 \times 26 = 52$  states because we need to be able to show a history of size 2 (i.e. all possible pairs of characters). The decomposition of the character substitution state based on the character trigram model is shown in Fig. 7.4. Therefore, the GEHMM model based on the character trigram model will have  $6 \times 2 \times 26 + 3 = 315$  states.

# Chapter 8

## *Experimental Results, Future Work and Conclusion*

### 8.1 Outline of Keyword Spotting System and Design of Experiments

In this chapter, we present an experimental analysis of the proposed keyword spotting system over a collection of real-world documents. The keyword spotting system is composed of two major parts: segmentation and decision. In the former, we generate the word hypotheses. In the latter, we decide whether a generated word hypothesis is a keyword or not (Fig. 8.1).



**Figure 8.1** Two major modules of keyword spotting system and levels of experiments.

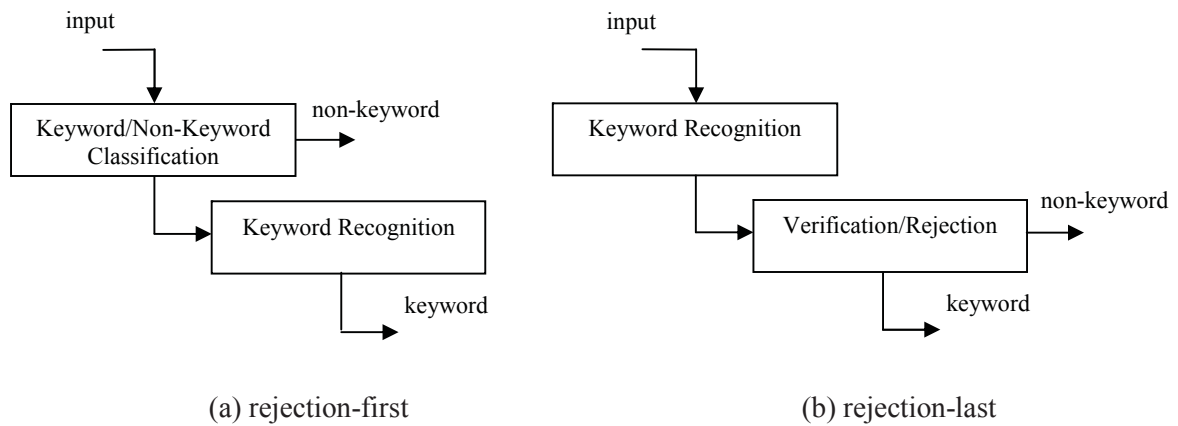
In order to gain a better insight into the performance of the keyword spotting system, we conduct two types of experiments: word-level and document-level. In the word-level experiments, we assume that we have already generated the word hypotheses out of

which we wish to separate keyword from non-keyword. In the document-level experiments, we evaluate the performance of the segmentation and decision steps combined.

### **8.1.1 Separation of Keywords from Non-keywords**

It has been shown through numerous studies that the recognition performance of word recognition engines is inversely proportional to the size of the lexicon [KSS05, Fuj08, GLF+09]. State-of-the-art handwriting recognition algorithms achieve very high performances on small size lexica (several tens of words). However, they achieve poorly on large size lexica (tens of thousands of words). In a word spotting application, the lexicon that is used in the documents may be large or even unlimited, because there are always identifiers (personal/city names), typos, etc. that we may not have seen before. However, the lexicon of keywords that we are interested in processing is limited and small (between one to a few tens of words). Therefore, the recognition problem in the word spotting context can be greatly simplified given that we can separate non-keywords from keywords with an acceptable accuracy.

In other words, in the word spotting context, we can think of the word recognition problem as a two-level classification. In one level, we decide whether the input image is a keyword or not, which is a binary classification task. In the other level, we classify the input image to one of the keyword classes given that we know it is a keyword. Therefore, recognition-based approaches to keyword spotting can be divided into two major categories depending on whether we perform keyword/non-keyword classification before or after keyword recognition as shown in Fig. 8.2.



**Figure 8.2 Two major approaches to recognition-based keyword spotting.**

The basic idea behind rejection-first approaches is to firstly detect and remove word candidates that are unlikely to be keywords, and secondly, recognize the surviving candidates using the keyword recognition algorithm. The keyword/non-keyword classification is based on some global features such as word profiles. In some recent word spotting systems [RSP09b], such simple features as word lengths have been used. It must be noted that the use of coarse global features only allow for the pruning of non-keywords that are significantly different in shape than keywords. In such cases, the keyword recognition algorithm must have the capability of handling Out-Of-Vocabulary (OOV) inputs, or otherwise, those non-keywords that pass the first step would be inevitably classified as keywords (i.e. false positives).

In rejection-last approaches, the rejection of non-keywords is postponed to the last step of recognition. The input image which could be either a keyword or a non-keyword is sent to the keyword classifier. If the keyword classifier has a separate class for OOV inputs, the non-keyword input is (hopefully) assigned to the OOV class. Otherwise, the non-

keyword input is assigned to a keyword class. The basic idea behind rejection-last approaches is that a non-keyword that is classified as a keyword normally gets a lower confidence score by the recognition engine, compared to the average confidence score of valid samples of that keyword class.

The problem of keyword/non-keyword separation has traditionally been approached by simplistic methods. Most word spotting systems encompass some pruning techniques based on global features and/or some rejection schemes based on confidence scores thresholding [RSP09b, FFMB12]. We are not aware of any systematic study on the extent to which standard machine learning approaches can be useful in the separation of a set of keywords from a set of non-keywords that belong to an open lexicon where we may not have seen all lexicon entries in advance. We hope that the research results that we present in this chapter would shed some light on this topic.

The four major approaches to keyword/non-keyword separation that we study are as follows:

1) Binary Classification based on Global Features:

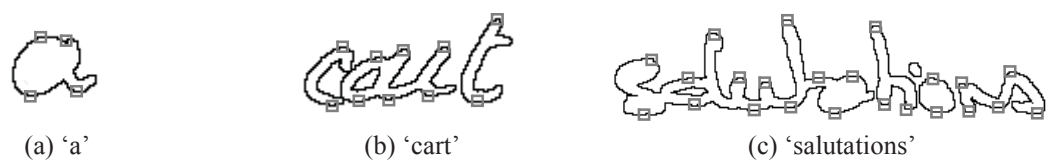
In this approach, we represent the input image by a set of global features and then classify the image as keyword or non-keyword using a standard binary classification algorithm. The results of previous research works show that the most successful feature extraction methods for holistic handwriting recognition are based on gradient features and Gabor filters [LNSF04, Liu07, WDL05]. Gradient-based features have also been successfully utilized in word spotting systems [RP08]. Therefore, we experimented with both gradient-based features and Gabor-based features. As for the classification

algorithm, we experimented with the Support Vector Machine (SVM) method with linear, polynomial and Radial Basis Function (RBF) kernels.

## 2) Binary Classification based on Local Features:

In this approach, we represent the input image by a set of local features that are dependent on the geometrical or temporal aspect of the image. As such, the number of features may be different from one image to another. Consequently, classification algorithms based on local features require distance functions that are defined on sequences (i.e. variable-length vectors).

An example of a local feature extraction method for cursive handwriting is given in [xDKSP05], where a word image is represented by the set of maxima points on the upper external contour and the set of minima points on the lower external contour of the image (Fig. 8.3). Therefore, the number of features is proportional to the length of the image. Normally, the longer the word, the more features we need to describe it.



**Figure 8.3** Examples of local minima/maxima contour points of handwritten words.

The most widely-used ways of comparing sequences of different lengths are based on generative models such as Hidden Markov Model (HMM), or *optimal alignment* methods such as Dynamic Time Warping (DTW) or the Generalized Minimum Edit (GME) distance that we introduced in the previous chapter. In our first set of experiments, we use

the local minima/maxima features [xDKSP05] along with the DTW distance. We experimented with both the classical unconstrained version of the DTW and a modified version that adds a locality constraint [SC78].

Although local/temporal features provide a sensible way of representing signals, classification based on sequences requires more elaborate techniques than classification based on fixed-length vectors. The most straightforward classification algorithm based on sequence distance functions is the K-Nearest Neighbor (K-NN) classifier that we will use for the purpose of these experiments. We must note that the K-NN algorithm is a type of lazy learning which generally requires the computation of the distance from a new sample to all training samples at the run time, and thus it is not efficient in terms of memory and speed.

Later, we will discuss how to replace the K-NN classifier with more efficient classification methods using Kernel Methods (KMs). KMs provide an elegant approach to the processing of arbitrary data structures. Using suitable kernels, classical classification methods, such as perceptrons, that were originally designed for the processing of fixed-length vectors can be readily applied to other common types of data such as sequences, trees and etc.

### 3) Generalized Minimum Edit Distance Thresholding:

In this approach, we model the keywords individually (one model per keyword). The keyword model is based on the Generalized Minimum Edit Distance (GMED) that we presented in the previous chapter. Then, we learn two distance distributions for each keyword using a validation set: one distance distribution for all samples of the keyword

and one distance distribution for all samples of all other words, where all distances are computed using the corresponding keyword model. Then, for a new image, we compute the distance against each keyword model and then classify the image as keyword or non-keyword based on whether the distance is closer to the keyword distribution or non-keyword distribution.

Since in this approach we model the keywords separately, it might happen that two (or more) different keyword models classify the same input image as keyword. Therefore, we must have an arbitration strategy to settle the conflict in such cases. There are two main strategies: exactly-one arbitration and at-least-one arbitration. In the exactly-one arbitration, we accept the input image as keyword if it is accepted by exactly one keyword model. In the at-least-one arbitration, we accept the input image as keyword if it is accepted by at least one keyword model, assigning it to the keyword class with the minimum distance. Generally speaking, using the exactly-one arbitration strategy, the word spotting system would achieve a lower false positive rate at the cost of a higher false negative rate; on the other hand, using the at-least-one arbitration strategy the system would achieve a lower false negative rate at the cost of a higher false positive rate.

#### 4) Normalized Probability Thresholding based on Universal Background Models:

This approach is an extension of distance/likelihood thresholding approaches. The idea is to learn one Universal Background Model (UBM) for all words along with individual models for keywords, and then use the universal model to normalize the raw score (distance or likelihood) that is obtained from specific keyword models.



The idea of universal or cohort models for score normalization has long been studied in the speech recognition community [RDL+92, RQD00]. However, it was not until recently that the UBM technique was introduced to the realm of keyword spotting [RSP09b].

In [RSP09b], the authors proposed Gaussian Mixture Models (GMMs) for the modeling of the universal lexicon. In this model, each keyword is modeled separately using a left-to-right HMM. At the classification time, the score obtained from a keyword HMM is divided by the universal score that is computed by the universal GMM; if the resultant normalized score exceeds a threshold, the image is accepted as keyword. Again, since keywords are modeled separately, we need an arbitration strategy as we discussed before. We will experiment with both *exactly-one* and *at-least-one* strategies.

Finally, it must be noted that the universal lexicon does not need to be modeled by GMMs. The reason that the authors in [RSP09b] use GMMs is their low memory and computational requirements compared to HMMs; remembering that a GMM is a special case of a HMM with only one state. In terms of classification performance, the experimentations carried out in [RSP09b] show that GMMs and HMMs result in more or less the same performance for the modeling of the universal lexicon. Our experiments also verify these results, meaning that GMMs provide an efficient approach to the modeling of large amount of universal data.

## 8.2 Training Data

The ultimate objective of our research is the spotting of user-specified keywords in real-world handwritten documents. Consequently, as mentioned earlier, we cannot rely on a

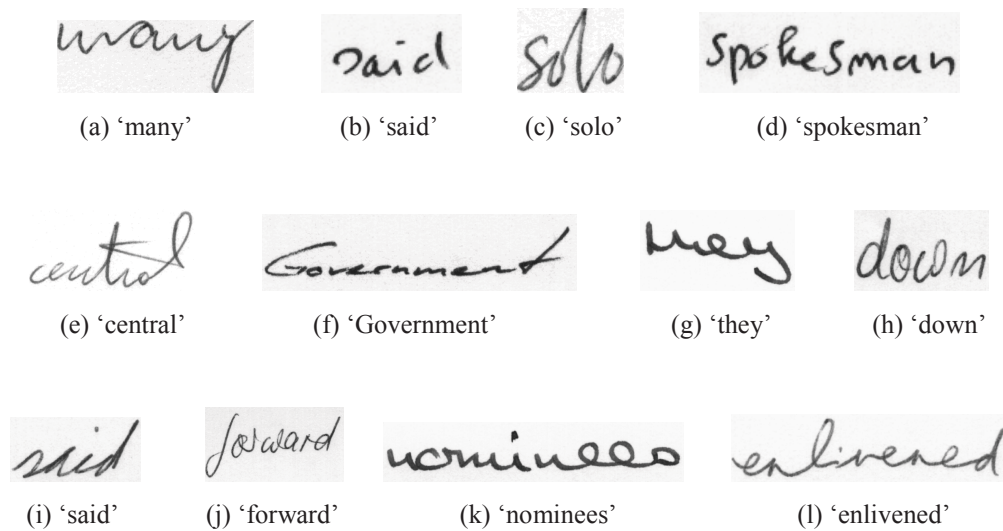
database of training images for all possible keywords. Therefore, we devised the GME distance which is an analytical approach to recognition that requires trained models of handwritten character. For the training of character models and adjusting the cost parameters of the GME distance we used the NIST SD 19 [GBC+94] and the IAM databases [MB02] (more details in Chapter 6 and 7). Therefore, the GME distance provides a similarity measure between an arbitrary text keyword and a word image without the need for further training on a specific lexicon/dataset.



**Figure 8.4** Samples of synthesized images for two words 'adhesion' and 'resiliation'.

However, for the training of the keyword/non-keyword classification schemes that we discussed in the previous section, we do require word-level training data. Even if we use an analytical recognition algorithm based on sub-character/character models along with a threshold-based approach to the rejection of non-keywords (approach 3 and 4 above), we still need a validation dataset of keyword samples and non-keyword samples in order to tune the rejection/acceptance thresholds for each keyword. In order to address this problem, we created a word image synthesizer that generates image samples for an arbitrary word based on handwritten fonts. For this purpose, we collected 184

handwritten fonts for the web. Fig. 8.4 shows samples of synthesized images for two words ‘adhesion’ (membership) and ‘resiliation’ (cancellation) that are among keywords that we want to spot in our collection of French documents.



**Figure 8.5** Samples of handwritten words from the IAM database.



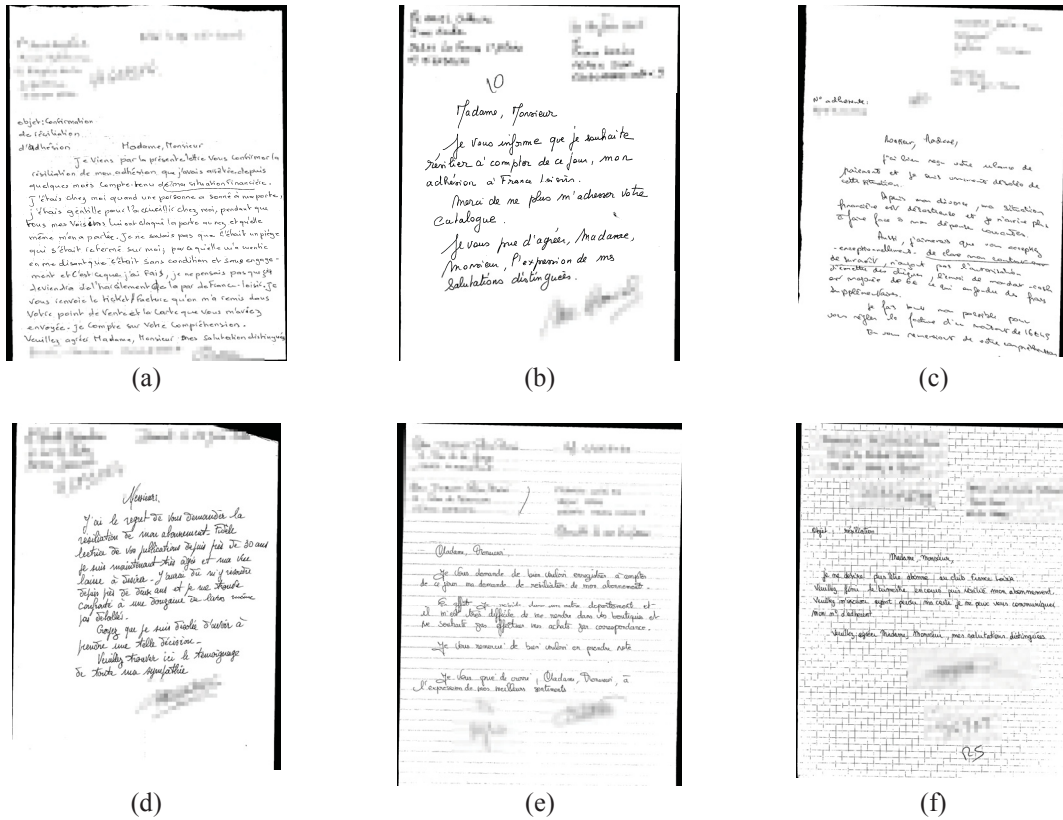


Figure 8.7 Samples of handwritten mails from our French documents database.

For the document-level experiments, however, as can be seen in Fig. 8.6, the simple controlled layout of the IAM documents may not reflect all the challenges that we may encounter in the processing of real-world handwritten documents, such as non-uniform skew and touching lines that we discussed in Chapter 1. Therefore, aside from the IAM database, we carried out our document-level experiments on a proprietary database of real-world French documents<sup>4</sup>. All samples in this database are real-world handwritten mails that are submitted to the customer-support department of a company by its wide-range of clients from France and French-speaking Africa. The task of the word spotting system is to find certain keywords, and then direct the customer request to the responsible

<sup>4</sup> This database belongs to IMDS Software, the key industrial partner and sponsor of this research project. For more information, please visit: [www.imds-world.com](http://www.imds-world.com).

department accordingly. Examples of keywords of interest in this application are: “resiliation” (cancellation), “adhesion” (membership), “contrat” (contract), “chomage” (unemployment), “santé” (health) and etc. Fig. 8.7 show samples of handwritten mails from our French documents database, where the personal information are pixelated to protect the customer identity.

## **8.4 Experiments**

In the following, we will present our experimental results on the English and French test databases described above. We start with the word-level experiments and then move on to the document-level experiments. Along the way, we compare and contrast our results with state-of-the-art word spotting systems for both the English [FFMB12] and French [RSPSL10] languages.

### **8.4.1 Word-Level Experiments**

We divide our word-level experimental results into two categories: word recognition and word spotting. In the former, our goal is to analyze the performance of our proposed approach along with some other popular methods for the recognition of small size lexica; that make up the keyword sets. In the latter, the focus of our experiments is the separation and recognition of a small set of known keyword classes from a large or open set of non-keywords.

### 8.4.1.1. Word Recognition Experiments

In order to evaluate the performance of the GME distance approach to the recognition of handwritten words, we carried out several sets of experiments on the IAM database. In order to keep the evaluation process as unbiased as possible, in the following experiments, we did not use the part of the IAM database that was already used for the generation of the cursive characters database and the training of the underlying cursive character models as described in Chapter 6.

**Table 8.1 Average recognition rate of the GME approach over several test subsets of IAM.**

Model	Cost Functions	Test Lexicon Size			
		10	20	50	100
5-state	Default	91.0	83.6	80.4	74.3
	Trained	92.1	84.7	81.3	75.7
	Adapted	92.4	85.1	81.7	76.2
159-state	Default	93.0	86.8	82.1	77.0
	Trained	93.4	87.2	82.7	77.6
	Adapted	94.1	87.7	82.8	77.8
315-state	Default	93.0	87.5	82.6	77.4
	Trained	93.5	88.1	83.1	78.0
	Adapted	94.3	88.7	83.5	78.5

The GME approach to word recognition can be used with or without the training of the associated cost functions as we discussed in Chapter 7. In fact, using the default cost functions we can accomplish word-level recognition without the need for word-level training data. However, given that word-level training data are available, we can optimize the cost functions for a specific distribution of data using the hidden Markov model framework. The model can be trained using a wide range of handwriting samples from different writers or it can be further adapted for a specific person by using their

handwriting samples. In the following, the former model will be referred to as “trained”, and the latter model will be referred to as “trained/adapted”, or simple “adapted”.

Moreover, the GME model may or may not use the lexicon knowledge. The three variations that we discussed in the previous chapter are the basic 5-state model (without knowledge of the lexicon), the 159-state model (with unigram and bigram knowledge of the lexicon) and the 315-state model (with unigram, bigram and trigram knowledge of the lexicon).

Table 8.1 shows the average recognition rate of the GME approach over several test subsets of the IAM database with different lexicon sizes. Again, for the trained and adapted models, the training and test subsets are completely disjoint.

The reason we limited the lexicon size to 100 words is that in a typical word spotting application, the lexicon of interest contains only a few tens of keyword. In the French keyword spotting application that we mentioned in the previous section, the keyword lexicon contains 48 keywords. The number of effective classes is even smaller; because among these keywords, we have conjugations and plural forms; for example “contrat” (contract) and “contrats” (contracts), or “résilier” (cancel) and “résiliée” (cancelled) which essentially indicate the same class/action. This means that if the recognition algorithm mistakes “contrat” for “contrats” or vice versa, it is not counted as an error from the word spotting view point, because they belong to the same *keyword family*. However, in the word recognition experiments that we summarize in this section, we simply treat all words as separate classes; so for example, in the test sets of the IAM database, we have “American” and “Americans” as two different classes.



The recognition results summarized in Table 8.1 indicate that adding the trigram knowledge to the model slightly improves the recognition results over the unigram/bigram model, which is in turn better than the basic model without any knowledge about the lexicon. The performance difference between the trigram-based models and the unigram/bigram-based models is 0.5% on average. However, the performance difference between the unigram/bigram-based models and the basic models without the lexicon knowledge is around 2.0% on average. This means that the major recognition improvement is obtained by virtue of the unigram/bigram knowledge. In summary, the most elaborate model (with adaptation given that a writer’s handwriting is available) improves the recognition rate over the model basic model (without training/adaptation) from 3.1% to 5.1%. It is interesting to note that the GME approach can achieve an acceptable word recognition performance without any training at the word-level.

**Table 8.2 Average recognition rate of the GME approach over several test subsets of IAM without perturbation-based character recognition.**

Model	Cost Functions	Test Lexicon Size			
		10	20	50	100
5-state	Default	90.1	81.1	76.3	69.1
	Trained	90.9	82.0	77.2	70.2
	Adapted	91.5	82.3	77.5	70.8
159-state	Default	92.1	84.1	77.9	71.5
	Trained	92.4	84.6	78.6	72.1
	Adapted	93.0	85.2	78.7	72.3
315-state	Default	92.3	85.1	78.5	72.1
	Trained	92.6	85.7	79.3	72.9
	Adapted	92.2	86.2	79.6	73.4

One advantage of the GME model is that it can be combined with any character recognition algorithm. The results that we summarized in Table 8.1 were obtained using the perturbation-based cursive character recognition algorithm that we described in Chapter 6. In order to show the effectiveness of the perturbation-based approach in the context of word recognition, we repeated the same experiments as in Table 8.1 but without character perturbation. The results are summarized in Table 8.2. As can be seen, the perturbation-based character recognition improves the word recognition performance in all cases, by 1% (for the smallest lexicon) to 5% (for the largest lexicon). The average recognition improvement due to the perturbation is 3.3%.

**Table 8.3 Average recognition rate of implemented handwritten word recognition algorithms several test subsets of IAM.**

Model		Training	Test Lexicon Size			
			10	20	50	100
LR-D-HMM		Trained	83.1	78.2	70.5	63.3
		Adapted	83.9	79.1	71.2	64.1
LR-C-HMM		Trained	84.5	82.0	71.8	65.5
		Adapted	85.3	82.7	72.2	66.1
MS-HMM		Trained	86.7	83.2	78.9	75.7
		Adapted	87.4	84.1	80.6	76.9
P2D-HMM		Trained	87.2	84.0	80.4	77.1
		Adapted	88.1	84.9	82.0	78.0
GME (EHMM + NN ensemble)	5-state	Default	91.0	83.6	80.4	74.3
		Trained	92.1	84.7	81.3	75.7
		Adapted	92.4	85.1	81.7	76.2
	159-state	Default	93.0	86.8	82.1	77.0
		Trained	93.4	87.2	82.7	77.6
		Adapted	94.1	87.7	82.8	77.8
	315-state	Default	93.0	87.5	82.6	77.4
		Trained	93.5	88.1	83.1	78.0
		Adapted	94.3	88.7	83.5	78.5

In order to put our results into perspective, we experimented with several popular HMM-based approaches to word recognition. A direct comparison of the results we presented in Table 8.1 and published works is not quite meaningful; because first of all, not all public databases define a standard training and test sets (the IAM database is an example); second, different word recognition algorithms use different pre-processing/post-processing techniques that may considerably affect the performance, even with the same training and testing sets of the same database; and third, the evaluation criteria may not be the same. Therefore, in order to see how the proposed 2D GME approach compares with traditional HMM-based approaches, we carried out our experiments on the same training and test sets based on our implementation of these algorithms with the same pre-processing steps for all algorithms.

The four common pre-processing steps include: 1) image height normalization; 2) stroke-width normalization based on skeletonization [ZS84] followed by dilation; 3 and 4) skew correction and slant correction based on horizontal and vertical projection profiles. Currently, we do not apply any post-processing techniques such as lexicon reduction or verification based on language models. The handwriting recognition algorithms that we experimented with include: 1) Left-to-Right Discrete HMM (LR-D-HMM) [CLK95]; 2) Left-to-Right Continuous HMM (LR-C-HMM) [RSP09b]; 3) Multi-Stream HMM (MS-HMM) [KPBH10]; and Pseudo 2D-HMM (P2D-HMM) [KA94]. The experimental results are shown in Table 8.3., where we copied the GME results from Table 8.1 for the sake of comparison. As can be seen, the GME approach based on a 159-state or 315-state EHMM and ensemble of neural networks significantly outperforms the other HMM-based approaches on small lexica. However, as the size of the lexicon grows, the GME

approach, which is based on an exact 2D model, and the P2D-HMM show almost the same performance. The interesting observation is that all the three HMM models that use some kind of 2D information, namely MS, P2D, and GME, outperform LR HMMs in all cases, particularly when the lexicon grows.

Although, the focus of this research is not to recognize large size lexica, it is worth mentioning that our manual inspection of misrecognized words shows that the major source of the error, is due to the difficulties in the recognition of curve characters. As we mentioned in Chapter 6, the amount of variations in cursive forms of characters is so high that in order for the classifier to resolve the ambiguity, it will have assign an unknown input shape to all possible character classes. The right sequence of characters is then chosen by using a larger context; that is, in the simplest form, a lexicon of words which acts as a constraint over the sequence of character hypotheses. However, as the lexicon size increases, the chance of a sequence of shapes being matched against more than one valid sequence of characters (i.e. lexicon entry) also increases, which in general, results in lower top-1 recognition rates for words.

It should be emphasized that the word recognition results that we presented above are based on an *exact (all or nothing)* evaluation criterion; meaning that the output of the word recognition algorithm is considered correct if and only if the top-1 word recognition hypothesis and the ground truth transcription of the input word image match character by character for all character positions. In [GLF+09] the authors have presented a word recognition system based on a new recurrent neural network architecture that achieves a reported word recognition rate of 73.3% to 74.9% over very large lexica of words (~5000

to 20000). However, these results are based on an *approximate* evaluation criterion which the authors refer to as word accuracy:

$$\text{word accuracy} = 100 \times \left(1 - \frac{\text{insertions} + \text{substitutions} + \text{deletions}}{\text{total length of test set transcriptions}}\right) \quad (8.1)$$

Therefore, for example if the word “Americans” is recognized as “American”, based on the approximate criterion that is used in [GLF+09], the calculated performance is  $100 \times (1 - 1/9) \approx 88.9\%$ . However, based on the exact criterion that we use, the calculated performance is 0% in this example.

#### **8.4.1.2. Word Spotting Experiments**

In the word spotting experiments we analyze a more general problem than in the previous section. We are not only interested in the recognition of keywords, but also in the detection (separation) of non-keywords from keywords. In section 8.1.1, we examined several major approaches to keyword/non-keyword detection. In the following we analyze the performance of these methods based on two sets of experiments. In the first set, we assume the global lexicon is “closed”, that is all non-keyword classes are known. In the second set, we drop the closed-lexicon assumption; that is we assume that there is always a chance of seeing a non-keyword that is outside the global lexicon that we know. This case, which we will refer to as “open” lexicon experiments, allows us to gain a better understanding of how word spotting algorithms would perform in real-world situations where there are always special names and typos that we must not spot as keywords.

It should be mentioned that the evaluation criteria in word spotting experiments are different from word recognition experiments. In a word spotting application, that is an example of an Information Retrieval (IR) process on scanned documents, we are looking for a “truth” that is the keywords of interests in the input documents; and we ideally want the algorithm to return “the truth, only the truth, and nothing but the truth”. In IR applications, this correctness measure is stated in terms of True Positive (TP), True Negative (TN), False Positive (FP) and False Negative (FN) rate. The definition of each of these terms in our word spotting application is as follows:

TP rate: the percentage of keywords that are correctly spotted by the algorithm.

TN rate: the percentage of non-keywords that are correctly not spotted by the algorithm.

FP rate: the percentage of non-keywords that are incorrectly spotted by the algorithm.

FN rate: the percentage of keywords that are incorrectly not spotted by the algorithm.

The ideal performance is equivalent of a TP rate of 100%, a TN rate of 100%, a FP rate of 0% and a FN rate of 0%. TP rate and FN rate sum to 1, therefore we only need one of these quantities to obtain the other. Similarly, for TN rate and FP rate. Thus, we need a pair of quantities to state the performance of a word spotting algorithm. One quantity (either TP or FN) states how well the algorithm performs in the retrieval of relevant information (i.e. keywords), and the other quantity (either TN or FP) states how well the

algorithm performs in the filtering of irrelevant information (non-keywords). Sometimes, it is preferred to state the performance in terms of a *precision-recall* pair. Precision states the probability that a (randomly selected) spotted word is relevant. Recall states the probability that a (randomly selected) keyword is spotted. In terms of TP, TN, FP and FN:

$$\text{precision} = \frac{\text{TP}}{\text{TP} + \text{FP}} \quad (8.2)$$

$$\text{recall} = \frac{\text{TP}}{\text{TP} + \text{FN}} \quad (8.3)$$

The ideal performance is equivalent of a precision rate of 100% and a recall rate of 100%. In order to see how far the performance of a word spotting algorithm is from the ideal performance, and thus be able to compare different algorithms, it is often easier to state the performance by a single quantity, rather than a pair of quantities. Numerous measures in the IR literature have been proposed for this purpose. Our experimental results are based on the so-called F-measure (a.k.a. F-score) that is one of the most popular measures for expressing the precision-recall pair as a single number. The basic form of the F-measure, denoted by  $F_1$  or simply  $F$ , is defined as the harmonic mean of precision and recall:

$$F = 2 \cdot \frac{\text{precision} \cdot \text{recall}}{\text{precision} + \text{recall}} \quad (8.4)$$

This basic form of F-measure treats precision and recall as equally important. However, in some applications, it may be preferable to attach more importance to precision than

recall or vice versa. Therefore, the general form of the F-measure, denoted by  $F_\beta$ , is defined as follows:

$$F_\beta = (1 + \beta^2) \cdot \frac{\text{precision} \cdot \text{recall}}{\beta^2 \cdot \text{precision} + \text{recall}} \quad (8.5)$$

The value of  $\beta$  determines whether we want to give more importance to precision or recall. With  $\beta > 1$ , we give  $\beta$  times as much importance to recall as precision, and with  $\beta < 1$ , we give  $1 / \beta$  times as much importance to precision as recall. With  $\beta = 1$ , we arrive at the basic or the balanced form of the F-measure.

In real world IR applications, normally as the precision rate increases the recall rate decreases and vice versa. In the extreme case, we can achieve a perfect recall by spotting all words (keywords and non-keywords), and on the other hand, we can achieve a faultless precision by spotting no words. Therefore, it is important to make a sensible compromise between precision and recall based on the application requirements.

In our case, the client prefers more accurate processing of a smaller amount of documents over a less accurate processing of a larger amount of documents. In terms of precision and recall, better precision is more important than better recall. For this reason, aside from the balanced F-measure, we report our results in terms of  $F_{0.5}$  measure as well ( $\beta = 0.5$  resulting in an evaluation criterion with precision being 2 times more important than recall).



**Table 8.4 Average performance of binary classification approaches to keyword/non-keyword separation over closed lexica for different sizes of keywords and non-keywords.**

Classification Approach		Keywords/Non-keywords Sets															
		1 keyword vs. 50 non-keywords				5 keywords vs. 500 non-keywords				20 keyword vs. 1000 non-keywords				50 keyword vs. 5000 non-keywords			
		P	R	F	F <sub>0.5</sub>	P	R	F	F <sub>0.5</sub>	P	R	F	F <sub>0.5</sub>	P	R	F	F <sub>0.5</sub>
Global Binary SVM+Gradient		90.1	91.8	90.9	90.4	83.3	80.9	82.1	82.8	59.1	63.1	61.0	59.9	52.7	57.8	55.1	53.6
Global Binary SVM+Gabor		92.5	90.7	91.6	92.1	85.1	81.2	83.1	84.3	60.7	61.0	60.9	60.8	52.7	59.1	55.7	53.9
Local Binary KNN+LMM		98.2	97.4	97.8	98.0	97.5	94.3	95.9	96.8	95.2	94.6	94.9	95.1	93.6	90.1	91.8	92.9
GMED Thresholding	OR	99.1	99.5	99.3	99.2	97.1	93.5	95.2	96.4	88.2	82.7	85.4	87.0	85.0	80.9	82.9	84.1
	XOR	99.8	99.1	99.4	99.7	98.0	93.4	95.6	97.0	89.9	82.5	86.0	88.3	86.3	80.1	83.1	85.0
UBM Thresholding	OR	98.8	99.5	99.1	98.9	97.5	94.0	95.7	96.7	91.3	88.0	89.6	90.6	90.2	84.7	87.4	89.0
	XOR	99.5	99.2	99.4	99.4	97.9	93.6	95.7	97.0	92.1	87.2	89.6	91.1	91.3	83.5	87.2	89.6

#### 8.4.1.2.1 Closed-Lexicon Word Spotting Experiments

We carried out our closed-lexicon experiments on several randomly selected subsets of the IAM database with different sizes for the keywords and non-keywords sets. The results are summarized in Table 8.4 for four experimental cases ranging from small-size keywords set/small-size non-keywords set to modest-size keywords set/large-size non-keywords set. In each case, we repeated the experiments for 5 randomly selected sets of keywords and non-keywords; thus every number reported in Table 8.4 is the average value of 5 independent experiments. For the thresholding-based approaches (GMED and

UBM), the arbitration strategy is denoted by OR  $\equiv$  at-least-one, and XOR  $\equiv$  exactly-one rules as described in Section 8.1.1.

As can be seen, the poorest performances are associated with the global binary approaches where we model all keywords by one model and all non-keywords by another model; while the highest performances are associated with model-based approaches where we model each keyword separately. The conjecture that local approaches, in this particular binary classification problem, work better than global approaches is supported by the fact that the highest performance is achieved by the KNN approach. By keeping all training samples of all keywords and non-keywords, the KNN approach provides the most elaborate and accurate keyword/non-keyword separation model; however, needless to mention, the KNN classifier entails high memory and computation requirements that limit its practicality. According to Table 8.4, the second best performance is associated with the UBM thresholding approach, which is consistently better than the GME thresholding, particularly for larger sets of keywords and non-keywords. The higher performance of the UBM approach is attributed to the so-called filler or negative model for non-keywords, which is missing in the simple GME thresholding approach where we model the keywords only.

In summary, if we have  $n$  training samples belonging to a family of  $N$  keywords, and  $m$  training samples belonging to a family of  $M$  non-keywords, the KNN approach, which results in the best performance, is composed of  $n + m$  models (i.e. exemplars). The UBM thresholding approach, which has the second best performance, is composed of  $N + 1$  models ( $N$  keyword models + 1 non-keyword model). The GME thresholding approach, which has the third best performance, is composed of  $N$  models. And the two global

binary approaches, which lead to the lowest performances, are composed of 1 model each.

**Table 8.5. Average performance of binary classification approaches to keyword/non-keyword separation over open lexica for different sizes of keywords and non-keywords.**

Classification Approach		Keywords/Non-keywords Sets															
		Train															
		1 keyword vs. 50 non-keywords				5 keywords vs. 500 non-keywords				20 keyword vs. 1000 non-keywords				50 keyword vs. 5000 non-keywords			
		Test															
		1 keyword vs. 50 non-keywords + 1000 OOV non-keywords				5 keywords vs. 500 non-keywords + 2000 OOV non-keywords				20 keyword vs. 1000 non-keywords + 5000 OOV non-keywords				50 keyword vs. 5000 non-keywords + 5000 OOV non-keywords			
		P	R	F	F <sub>0.5</sub>	P	R	F	F <sub>0.5</sub>	P	R	F	F <sub>0.5</sub>	P	R	F	F <sub>0.5</sub>
Global Binary	SVM+Gradient	78.1	83.4	80.7	79.1	79.6	80.4	80.0	79.8	54.7	63.0	58.6	56.2	49.9	57.0	53.2	51.2
Global Binary	SVM+Gabor	79.4	80.2	79.8	79.5	80.0	80.8	80.4	80.2	56.2	59.9	58.0	56.9	50.4	58.7	54.2	51.9
Local Binary	KNN+LMM	90.9	96.5	93.6	92.0	93.0	94.1	93.5	93.2	93.4	94.1	93.7	93.5	91.2	89.8	90.5	90.9
GMED Thresholding	OR	92.1	95.1	93.6	92.7	93.0	95.1	94.0	93.4	85.3	80.6	82.9	84.3	82.2	80.6	81.4	81.9
	XOR	94.5	91.7	93.1	93.9	94.2	92.6	93.4	93.9	86.8	80.0	83.3	85.3	83.9	79.8	81.8	83.0
UBM Thresholding	OR	92.0	95.5	93.7	92.7	94.2	95.7	94.9	94.5	90.9	86.9	88.9	90.1	89.7	82.6	86.0	88.2
	XOR	94.0	93.6	93.8	93.9	94.5	95.4	94.9	94.7	91.0	85.4	88.1	89.8	90.1	81.9	85.8	88.3

#### **8.4.1.2.2 Open-Lexicon Word Spotting Experiments**

The results of the open-lexicon experiments are summarized in Table 8.5. All settings are the same as the closed-lexicon experiments, except for the test sets. In each experiment, the lexicon of the test set is a super set of the lexicon of the corresponding training set (for non-keywords). Therefore, we can gain an understanding of the ability of each classifier in the rejection of unseen irrelevant information (i.e. OOV non-keywords).

In terms of the classification performance, we observe the same trend in Table 8.5 as in the closed-lexicon experiments, with the KNN approach again resulting in the best performance for larger lexica.

However, as is expected, the performance measures in open-lexicon experiments are slightly lower than the corresponding closed-lexicon experiments. The reason is obviously OOV non-keywords that have similar features to the already seen keyword instances. In a real-world word spotting application, once these OOV non-keywords are detected, we can add them to the non-keyword lexicon and re-train (refine) the keyword/non-keyword classifier consequently. The automatic generation of image samples that we discussed in Section 8.2, serves this purpose as it obviates the need for the manual gathering of real handwritten samples for new words.

#### **8.4.2 Document-Level Experiments**

As we mentioned before, the purpose of document-level experiments is to give us an idea of the average performance of the word spotting system at the document level. In more tangible terms, we are interested to estimate the chance of reliably hitting a given keyword in a given document under real-world circumstances.

Based on our modular approach to word spotting, we can obtain a rough estimate of the document-level performance by multiplying the performances of the two major modules of the system. These two major modules are segmentation and decision, where the decision module itself is composed of two sub-modules: keyword/non-keyword classification and keyword recognition. Roughly speaking, we can achieve a performance of ~90% in each of these three independent units; therefore in theory, the overall performance is expected to be around  $90\%^3 \approx 72\%$ .

However, in practice, firstly, there are many other factors that may affect the performance, and secondly, as we mentioned before, most often there is a compromise that has to be made between the quality (precision) and the quantity (recall) of automatic processing. Therefore, it is more insightful to express the average performance in terms of quality and quantity rather than a single number.

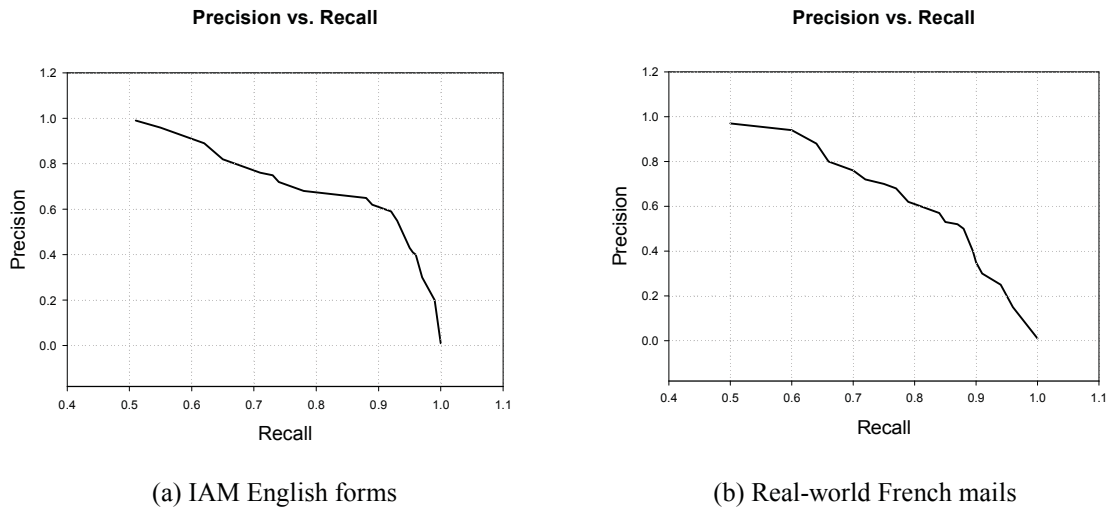
**Table 8.6 List of keywords for French keyword spotting experiments.**

Adhère	Résilier	Clore	chômage
adhérer	résiliee	Cesser	Financier
adhésion	résiliation	stoppée	financiers
adhérent	Annulé	stopper	financière
adhérents	annuler	Santé	financières
adhérente	annulations	Radié	décès
Abonné	Arrêt	radiée	décédé
abonnement	arrêté	radier	décéder
abonnements	arrêter	Rayer	décédée
Résilie	arrêtée	Romper	contrat

We already saw samples of handwritten documents from the IAM English database and our proprietary French database in Fig. 8.6 and 8.7. In order to assess the overall performance of the word spotting algorithm, we choose 100 documents from each database. For the IAM database, we randomly chose 5 subsets of the lexicon as the keyword sets, and therefore we carried out 5 independent experiments to assess the average word spotting performance. For the French database, the set of keywords are given in Table 8.6. These are mostly “action” words, as we described earlier, that the customer service of the company wishes to spot in the clients’ mails. It should be mentioned that it often happens that certain keywords appear more than once in a document; for example if a keyword appears in the title of the document, it will appear in the body as well. Typically, it is unnecessary to spot all instances of a keyword in a document. In other words, if the algorithm hits only one instance of a keyword within a document, then the missing instances of the keyword should not count as false negatives. However, in the following results, we treat different instances of a keyword within a document as independent entities. This will lead to a less biased estimate of the word spotting performance, and consequently a more meaningful comparison with other methods.

The average precision-recall curves for the IAM database of English forms and our collection of real-world French mails are shown in Fig. 8.8. The maximum F-measure on the IAM database is 74.7%, while the maximum F-measure on the French mails is 73.9%. As we mentioned before, the main difference between the IAM database of forms and the French collection of mails is the controlled vs. uncontrolled writing environment that

mainly affects the document layout. Therefore, these results imply that the average performance is only slightly compromised ( $\sim 1\%$ ) by the uncontrolled writing environment of the real-world handwritten mails.



**Figure 8.8 Average keyword spotting performance in terms of precision-recall curves.**

Our results, even without any word/document-level training, are comparable with two state-of-the-art word spotting systems for English [FFMB12] and French [RSPSL10] documents. Both systems use a portion of the groundtruth database for the training/adaptation of the underlying recognition models. In [FFMB12], the authors have also utilized a bigram language model, which equips the keyword spotting system with more context knowledge. Again we must emphasize that a direct comparison of published results is not quite meaningful. But in order to get a general idea of how these algorithms perform, we briefly mention their main results. In [RSPSL10], the authors report an average false rejection rate of 40% and an average a false acceptance rate of 0.26%. This means that the algorithm retrieves 60% of the keywords correctly, while it

returns 26 out of every 10,000 non-keywords by mistake. In some of our experiments using the KNN+LMM approach, we could achieve a false positive rate of as low as 5 out of 10,000 non-keywords at almost the same true positive rate (~58%). In [FFMB12], the authors report average precision rates of 41% to 94% on different databases under a variety of settings for writer adaptation and language models. The average precision rate of our proposed algorithm is 59.6% on the IAM database, and 56.7% on the French mails collection, which can be considered as a competent performance in view of the fact that we used no parts of the documents collection for the adjustment or adaptation of the word spotting algorithm.

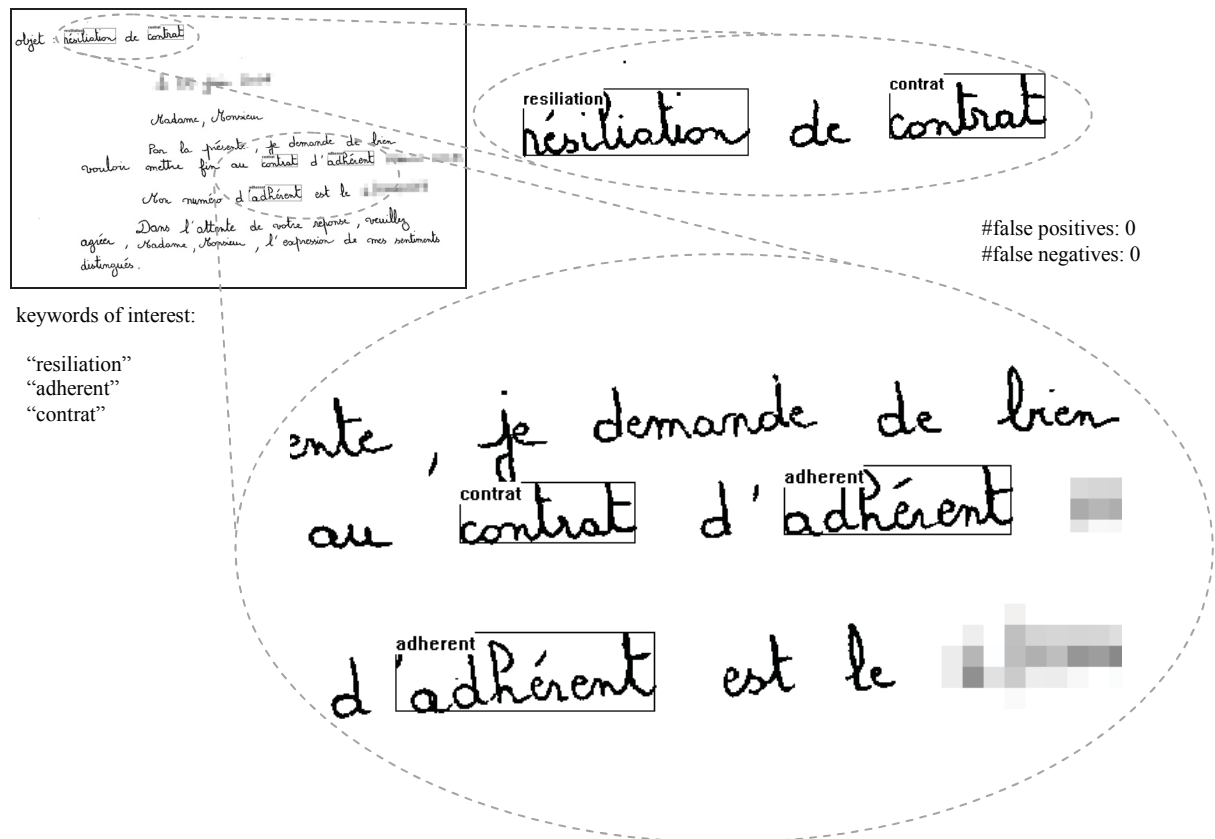


Figure 8.9 Sample output of proposed word spotting system for a handwritten document from the database of incoming mails.



Our proposed keyword spotting algorithm is integrated into AD'DOC IIM, The Automated Document Capture Solution of IMDS Software<sup>5</sup>. The average processing time for a typical handwritten mail (Fig. 8.7), including all the steps from noise removal to keyword recognition, is around 2-3 seconds on a Pentium 4 2.4 GHz PC, depending on the size of the document (i.e. the number of words). It must be mentioned that, although all the algorithms are implemented in C++, the major objective of the first phase of the development was obviously the correctness of the algorithms rather speed optimization. The current implementation is single-threaded, and obviously does not use the full capacity of now prevalent multi-core processors. Our modular approach easily supports parallel processing; as the text lines are independent, we can carry out the two major operations (word segmentation and keyword detection/recognition) within each line of text concurrently. We expect the processing time would be improved by a factor of 4 to 8 times (~0.25 to 0.75 sec per document) after the code is optimized for speed.

## **8.5 Future Work**

The research on keyword spotting constitutes a broad category of disciplines from image processing, computational geometry, pattern recognition and machine learning to statistic decision theory, information retrieval and language modeling. Roughly speaking, the state-of-the-art performance of general keyword spotting systems is around 50-60%, which means that the problem of automatic keyword spotting has still a long way to go before reaching maturity.

---

<sup>5</sup> For more information about AD'DOC IIM please visit [www.imds-world.com/en/software.html](http://www.imds-world.com/en/software.html)

Based on the current state of our work, we propose the following areas as the future directions for the keyword spotting research:

- Investigation of geometrical perturbation models for handwritten fonts.
- Development of adaptation techniques for generative models of handwriting.
- Synthesis of training data for arbitrary words based on generative models of handwriting.
- Investigation of online learning methods in the context of binary keyword/non-keyword classification for arbitrary keywords.
- Analysis of one-class learning methods (i.e. one-class SVM) for the separation of limited sets of keywords from unlimited sets of non-keywords.
- Study and development of robust features for cursive handwritten characters.
- Development of dual learning techniques for addressing the inherent problem of fuzziness in handwritten character shapes.
- Combination of local and global techniques for the improvement of the recognition of handwritten words.
- Investigation of template-based and segmentation-free keyword spotting methods for complex layout and unconstrained documents.

## **8.6 Conclusion**

Keyword spotting is the core problem in search, classification and retrieval of document images. We presented a top-down approach to the spotting of arbitrary keywords in handwritten document images. The main goal of our approach was the development of a

methodology for the automatic processing of real-world documents with a reliable performance without the need for the manual gathering of real handwritten samples for new words or the manual adjustment of the underlying algorithms for new datasets. To this end, we studied the challenges that we encounter in the processing of real-world documents and we proposed efficient algorithms to address the three major problems encompassing keyword spotting, namely, denoising, line/word segmentation and keyword detection/recognition.

The main contribution of our work was the development of a generalized minimum edit distance for handwritten words. We showed this distance is equivalent to an Ergodic Hidden Markov Model (EHMM), therefore we were able to use the standard Expectation Maximization (EM)-based optimization algorithms for the adjustment of the associated cost functions of the proposed distance. The main advantage of our approach was to provide an exact model for the temporal information present in the handwriting with a feasible number of states (less than a few hundred). To the best of our knowledge, this is the first work to present an exact 2D model for handwritten words while satisfying practical constraints. Other contributions of this research were the development of eight algorithms as follows:

- 1) Removal of page margins based on corner detection in projection profiles. For further information, please see [HBS09].
- 2) Removal of noise patterns in handwritten images using expectation maximization and fuzzy inference systems, which is the extension of the noise removal method that we discussed in Chapter 3. For further information, please see [HBS12].

- 3) Extraction of text lines and words based on Fast Fourier-based Steerable (FFS) filtering.
- 4) Development of a statistical hypothesis testing method based on Markov chains and HMMs for word segmentation algorithms. For further information, please see [HSB+12].
- 5) Segmentation of characters based on skeletal graphs.
- 6) Detection of under-segmented characters using fuzzy inference systems.
- 7) Merging of broken characters based on graph partitioning. For further information, please see [HBS11].
- 8) Recognition of handwritten cursive characters based on input perturbation and classification combination.

We carried out extensive experiments on a benchmark database of handwritten English documents and a real-world collection of handwritten French documents. The results indicate that even without any word/document-level training, our proposed approach provides a competent performance which is comparable with two state-of-the-art word spotting systems for English and French documents.

## List of Publications

Below is a list of papers published or under consideration based on my thesis:

Mehdi Haji, Tien D. Bui and Ching Y. Suen, “Removal of Noise Patterns in Handwritten Document Images Using Expectation Maximization and Fuzzy Inference Systems,” *Pattern Recognition*, 2012. (under second review)

Mehdi Haji , Kalyan A. Sahoo, Tien D. Bui, Ching Y. Suen and Dominique Ponson, “Statistical Hypothesis Testing for Handwritten Word Segmentation Algorithms,” *2012 International Conference on Frontiers in Handwriting Recognition (ICFHR-2012)*. (submitted)

Mehdi Haji, T. D. Bui and C. Y. Suen, “Automatic extraction of numeric strings in unconstrained handwritten document images,” *Document Recognition and Retrieval XVIII, Proceedings of SPIE, Volume 7874, Jan. 2011*.

Mehdi Haji , Tien D. Bui, C. Y. Suen, “Simultaneous Document Margin Removal and Skew Correction Based on Corner Detection in Projection Profiles,” *Image Analysis and Processing , Lecture Notes in Computer Science, Volume 5716/2009, 2009*.

## Appendix

### *A1. Rule Base for Detection of Dots and Small Noises*

Rule #1. if Normalized Height is SMALL\_COMPARED\_TO\_NASW and Normalized Width is SMALL\_COMPARED\_TO\_NASW and Normalized YCOG is BOTTOM then Dot is very LOW and Small Noise is very HIGH.

Rule #2. if Normalized Height is SMALL\_COMPARED\_TO\_NASW and Normalized Width is SMALL\_COMPARED\_TO\_NASW and Normalized YCOG is TOP then Dot is MEDIUM and Small Noise is HIGH.

Rule #3. if Normalized Height is SMALL\_COMPARED\_TO\_NASW and Normalized Width is EQUAL\_TO\_NASW and Normalized YCOG is BOTTOM and Aspect Ratio is AROUND\_1 then Dot is LOW and Small Noise is MEDIUM.

Rule #4. if Normalized Height is SMALL\_COMPARED\_TO\_NASW and Normalized Width is EQUAL\_TO\_NASW and Normalized YCOG is BOTTOM and Aspect Ratio is not AROUND\_1 then Dot is LOW and Small Noise is LOW.

Rule #5. if Normalized Height is SMALL\_COMPARED\_TO\_NASW and Normalized Width is EQUAL\_TO\_NASW and Normalized YCOG is TOP and Aspect Ratio is AROUND\_1 then Dot is HIGH and Small Noise is MEDIUM.

Rule #6. if Normalized Height is SMALL\_COMPARED\_TO\_NASW and Normalized Width is EQUAL\_TO\_NASW and Normalized YCOG is TOP and Aspect Ratio is not AROUND\_1 then Dot is MEDIUM and Small Noise is MEDIUM.

Rule #7. if Normalized Height is SMALL\_COMPARED\_TO\_NASW and Normalized Width is LARGE\_COMPARED\_TO\_NASW and Normalized YCOG is BOTTOM then Dot is very LOW and Small Noise is LOW.

Rule #8. if Normalized Height is SMALL\_COMPARED\_TO\_NASW and Normalized Width is LARGE\_COMPARED\_TO\_NASW and Normalized YCOG is TOP then Dot is LOW and Small Noise is LOW.

Rule #9. if Normalized Height is EQUAL\_TO\_NASW and Normalized Width is SMALL\_COMPARED\_TO\_NASW and Normalized YCOG is BOTTOM and Aspect Ratio is AROUND\_1 then Dot is LOW and Small Noise is HIGH.

Rule #10. if Normalized Height is EQUAL\_TO\_NASW and Normalized Width is SMALL\_COMPARED\_TO\_NASW and Normalized YCOG is BOTTOM and Aspect Ratio is not AROUND\_1 then Dot is VERY LOW and Small Noise is MEDIUM.

Rule #11. if Normalized Height is EQUAL\_TO\_NASW and Normalized Width is SMALL\_COMPARED\_TO\_NASW and Normalized YCOG is TOP then Dot is HIGH and Small Noise is MEDIUM.

Rule #12. if Normalized Height is EQUAL\_TO\_NASW and Normalized Width is EQUAL\_TO\_NASW and Normalized YCOG is BOTTOM and Aspect Ratio is AROUND\_1 then Dot is LOW and Small Noise is MEDIUM.

Rule #13. if Normalized Height is EQUAL\_TO\_NASW and Normalized Width is EQUAL\_TO\_NASW and Normalized YCOG is BOTTOM and Aspect Ratio is not AROUND\_1 then Dot is very LOW and Small Noise is LOW

Rule #14. if Normalized Height is EQUAL\_TO\_NASW and Normalized Width is EQUAL\_TO\_NASW and Normalized YCOG is TOP and Aspect Ratio is AROUND\_1 then Dot is very HIGH and Small Noise is very LOW.

Rule #15. if Normalized Height is EQUAL\_TO\_NASW and Normalized Width is EQUAL\_TO\_NASW and Normalized YCOG is TOP and Aspect Ratio is not AROUND\_1 then Dot is somewhat HIGH and Small Noise is very LOW.

Rule #16. if Normalized Height is EQUAL\_TO\_NASW and Normalized Width is LARGE\_COMPARED\_TO\_NASW and Normalized YCOG is BOTTOM and Aspect Ratio is AROUND\_1 then Dot is LOW and Small Noise is LOW.

Rule #17. if Normalized Height is EQUAL\_TO\_NASW and Normalized Width is LARGE\_COMPARED\_TO\_NASW and Normalized YCOG is BOTTOM and Aspect Ratio is not AROUND\_1 then Dot is very LOW and Small Noise is very LOW.

Rule #18. if Normalized Height is EQUAL\_TO\_NASW and Normalized Width is LARGE\_COMPARED\_TO\_NASW and Normalized YCOG is TOP then Dot is MEDIUM and Small Noise is very LOW.

Rule #19. if Normalized Height is LARGE\_COMPARED\_TO\_NASW and Normalized Width is SMALL\_COMPARED\_TO\_NASW and Normalized YCOG is BOTTOM then Dot is very LOW and Small Noise is very LOW.

Rule #20. if Normalized Height is LARGE\_COMPARED\_TO\_NASW and Normalized Width is SMALL\_COMPARED\_TO\_NASW and Normalized YCOG is TOP then Dot is LOW and Small Noise is very LOW.



Rule #21. if Normalized Height is LARGE\_COMPARED\_TO\_NASW and Normalized Width is EQUAL\_TO\_NASW and Normalized YCOG is BOTTOM then Dot is very LOW and Small Noise is very LOW.

Rule #22. if Normalized Height is LARGE\_COMPARED\_TO\_NASW and Normalized Width is EQUAL\_TO\_NASW and Normalized YCOG is TOP and Aspect Ratio is AROUND\_1 then Dot is HIGH and Small Noise is very LOW.

Rule #23. if Normalized Height is LARGE\_COMPARED\_TO\_NASW and Normalized Width is EQUAL\_TO\_NASW and Normalized YCOG is TOP and Aspect Ratio is not AROUND\_1 then Dot is LOW and Small Noise is very LOW

Rule #24. if Normalized Height is LARGE\_COMPARED\_TO\_NASW and Normalized Width is LARGE\_COMPARED\_TO\_NASW then Dot is very LOW and Small Noise is very LOW.

## ***A2. Rule Base for Detection of Dashes***

Rule #1. if Normalized Height is LOW and Normalized Width is MEDIUM and Denseness is LOW and Eccentricity is not HIGH then Dash is LOW.

Rule #2. if Normalized Height is LOW and Normalized Width is MEDIUM and Denseness is LOW and Eccentricity is HIGH and Orientation is not HORIZONTAL then Dash is MEDIUM.

Rule #3. if Normalized Height is LOW and Normalized Width is MEDIUM and Denseness is LOW and Eccentricity is HIGH and Orientation is HORIZONTAL then Dash is somewhat HIGH.

Rule #4. if Normalized Height is LOW and Normalized Width is MEDIUM and Denseness is not LOW and Eccentricity is not HIGH then Dash is MEDIUM.

Rule #5. if Normalized Height is LOW and Normalized Width is MEDIUM and Denseness is not LOW and Eccentricity is HIGH and Orientation is not HORIZONTAL then Dash is MEDIUM.

Rule #6. if Normalized Height is LOW and Normalized Width is MEDIUM and Denseness is not LOW and Eccentricity is HIGH and Orientation is HORIZONTAL then Dash is HIGH.

Rule #7. if Normalized Height is LOW and Normalized Width is not MEDIUM and Denseness is LOW and Eccentricity is not HIGH then Dash is very LOW.

Rule #8. if Normalized Height is LOW and Normalized Width is not MEDIUM and Denseness is LOW and Eccentricity is HIGH and Orientation is not HORIZONTAL then Dash is very LOW.

Rule #9. if Normalized Height is LOW and Normalized Width is not MEDIUM and Denseness is LOW and Eccentricity is HIGH and Orientation is HORIZONTAL then Dash is MEDIUM.

Rule #10. if Normalized Height is LOW and Normalized Width is not MEDIUM and Denseness is not LOW and Eccentricity is not HIGH then Dash is very LOW.

Rule #11. if Normalized Height is LOW and Normalized Width is not MEDIUM and Denseness is not LOW and Eccentricity is HIGH and Orientation is not HORIZONTAL then Dash is LOW.

Rule #12. if Normalized Height is LOW and Normalized Width is not MEDIUM and Denseness is not LOW and Eccentricity is HIGH and Orientation is HORIZONTAL then Dash is MEDIUM.

Rule #13. if Normalized Height is MEDIUM and Normalized Width is MEDIUM and Denseness is LOW and Eccentricity is not HIGH then Dash is very LOW.

Rule #14. if Normalized Height is MEDIUM and Normalized Width is MEDIUM and Denseness is LOW and Eccentricity is HIGH and Orientation is not HORIZONTAL then Dash is LOW.

Rule #15. if Normalized Height is MEDIUM and Normalized Width is MEDIUM and Denseness is LOW and Eccentricity is HIGH and Orientation is HORIZONTAL then Dash is MEDIUM.

Rule #16. if Normalized Height is MEDIUM and Normalized Width is MEDIUM and Denseness is not LOW and Eccentricity is not HIGH then Dash is LOW.

Rule #17. if Normalized Height is MEDIUM and Normalized Width is MEDIUM and Denseness is not LOW and Eccentricity is HIGH and Orientation is not HORIZONTAL then Dash is LOW.

Rule #18. if Normalized Height is MEDIUM and Normalized Width is MEDIUM and Denseness is not LOW and Eccentricity is HIGH and Orientation is HORIZONTAL then Dash is MEDIUM.

Rule #19. if Normalized Height is MEDIUM and Normalized Width is not MEDIUM and Denseness is LOW and Eccentricity is not HIGH then Dash is very LOW.

Rule #20. if Normalized Height is MEDIUM and Normalized Width is not MEDIUM and Denseness is LOW and Eccentricity is HIGH and Orientation is not HORIZONTAL then Dash is very LOW.

Rule #21. if Normalized Height is MEDIUM and Normalized Width is not MEDIUM and Denseness is LOW and Eccentricity is HIGH and Orientation is HORIZONTAL then Dash is LOW.

Rule #22. if Normalized Height is MEDIUM and Normalized Width is not MEDIUM and Denseness is not LOW and Eccentricity is not HIGH then Dash is very LOW.

Rule #23. if Normalized Height is MEDIUM and Normalized Width is not MEDIUM and Denseness is not LOW and Eccentricity is HIGH and Orientation is not HORIZONTAL then Dash is very LOW.

Rule #24. if Normalized Height is MEDIUM and Normalized Width is not MEDIUM and Denseness is not LOW and Eccentricity is HIGH and Orientation is HORIZONTAL then Dash is LOW.

Rule #24. if Normalized Height is HIGH and Normalized Width is MEDIUM and Denseness is not LOW and Eccentricity is HIGH and Orientation is HORIZONTAL then Dash is LOW.

Rule #25. if Normalized Height is HIGH and not ( Normalized Width is MEDIUM and Denseness is not LOW and Eccentricity is HIGH and Orientation is HORIZONTAL ) then Dash is very LOW.

## References

- [All92] L. Allison. Lazy dynamic-programming can be eager. *Inf. Process. Lett.*, 43:207–212, September 1992.
- [AMCS04] K. T. Abou-Moustafa, M. Cheriet, and C. Y. Suen. On the structure of hidden markov models. *Pattern Recognition Letters*, 25(8):923 – 931, 2004.
- [ASW03] Jan-Mark Geusebroek Arnold, Arnold W. M. Smeulders, and Joost Van De Weijer. Fast anisotropic gauss filtering. *IEEE Transactions on Image Processing*, 12:2003, 2003.
- [BB08] Roman Bertolami and Horst Bunke. Hidden markov model-based ensemble methods for offline handwritten text line recognition. *Pattern Recognition*, 41(11):3452 – 3460, 2008.
- [Bra99] Ronald N. Bracewell. *The Fourier Transform & Its Applications*. McGraw-Hill Science/Engineering/Math; 3 edition, 1999.
- [Bre65] J. E. Bresenham. Algorithm for computer control of a digital plotter. *IBM System Journal*, 4(1):25–30, 1965.
- [CB05] Dongwook Cho and Tien D. Bui. Multivariate statistical modeling for image denoising using wavelet transforms. *Signal Processing: Image Communication*, 20(1):77 – 89, 2005.
- [CBG09] Huaigu Cao, Anurag Bhardwaj, and Venu Govindaraju. A probabilistic method for keyword retrieval in handwritten document images. *Pattern Recognition*, 42(12):3374–3382, 2009.

- [CK04] Beom-Joon Cho and Jin H. Kim. Print keyword spotting with dynamically synthesized pseudo 2d hmms. *Pattern Recognition Letters*, 25(9):999 – 1011, 2004.
- [CLK95] Wongyu Cho, Seong-Whan Lee, and Jin H. Kim. Modeling and recognition of cursive words with hidden markov models. *Pattern Recognition*, 28(12):1941 – 1953, 1995.
- [CLLT02] L. Cinque, S. Levaldi, L. Lombardi, and S. Tanimoto. Segmentation of page images having artifacts of photocopying and scanning. *Pattern Recognition*, 35(5):1167 – 1177, 2002.
- [CMGS10] Dan C. Ciresan, Ueli Meier, Luca M. Gambardella, and Juergen Schmidhuber. Deep big simple neural nets excel on handwritten digit recognition. March 2010.
- [CSSJ09] Paulo R. Cavalin, Robert Sabourin, Ching Y. Suen, and Alceu S. Britto Jr. Evaluation of incremental learning algorithms for hmm in the recognition of alphanumeric characters. *Pattern Recognition*, 42(12):3241 – 3253, 2009. New Frontiers in Handwriting Recognition.
- [Dam64] Fred J. Damerau. A technique for computer detection and correction of spelling errors. *Commun. ACM*, 7:171–176, March 1964.
- [DHS00] Richard O. Duda, Peter E. Hart, and David G. Stork. *Pattern Classification (2nd Edition)*. Wiley-Interscience, 2000.
- [DPB09] Xiaojun Du, Wumo Pan, and Tien D. Bui. Text line segmentation in handwritten documents using mumford-shah model. *Pattern Recogn.*, 42(12):3136–3145, 2009.

- [DT10] Mohammad Reza Daliri and Vincent Torre. Shape recognition based on kernel-edit distance. *Computer Vision and Image Understanding*, 114(10):1097 – 1103, 2010.
- [EDC97] Kamran Etemad, David Doermann, and Rama Chellappa. Multiscale segmentation of unstructured document pages using soft decision integration. *IEEE Trans. Pattern Anal. Mach. Intell.*, 19:92–96, January 1997.
- [EYSSG99] A. El-Yacoubi, R. Sabourin, C. Y. Suen, and M. Gilloux. An hmm-based approach for off-line unconstrained handwritten word modeling and recognition. *IEEE Trans. Pattern Anal. Mach. Intell.*, 21(8):752–760, 1999.
- [FA91] William T. Freeman and Edward H. Adelson. The design and use of steerable filters. *IEEE Trans. Pattern Anal. Mach. Intell.*, 13:891–906, September 1991.
- [FFB10] Volkmar Frinken, Andreas Fischer, and Horst Bunke. A novel word spotting algorithm using bidirectional long short-term memory neural networks. In Friedhelm Schwenker and Neamat El Gayar, editors, *Artificial Neural Networks in Pattern Recognition*, volume 5998 of *Lecture Notes in Computer Science*, pages 185–196. Springer Berlin / Heidelberg, 2010. 10.1007/978-3-642-12159-3\_17.
- [FFMB12] Volkmar Frinken, Andreas Fischer, R. Manmatha, and Horst Bunke. A novel word spotting method based on recurrent neural networks. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 34(2):211–224, feb. 2012.
- [FKSO10] Dominik Fisch, Bernhard Kühbeck, Bernhard Sick, and Seppo J. Ovaska. So near and yet so far: New insight into properties of some well-known classifier paradigms. *Information Sciences*, 180(18):3381 – 3401, 2010.



- [FO08] Leandro A. F. Fernandes and Manuel M. Oliveira. Real-time line detection through an improved hough transform voting scheme. *Pattern Recogn.*, 41:299–314, January 2008.
- [Fuj08] Hiromichi Fujisawa. Forty years of research in character and document recognition-an industrial perspective. *Pattern Recogn.*, 41(8):2435–2446, 2008.
- [FWL02] Kuo-Chin Fan, Yuan-Kai Wang, and Tsann-Ran Lay. Marginal noise removal of document images. *Pattern Recognition*, 35(11):2593 – 2611, 2002.
- [GBC<sup>+</sup>94] Michael D. Garris, James L. Blue, Gerald T. Candela, Gerald T. C, Darrin L. Dimmick, Jon Geist, Patrick J. Grother, Stanley A. Janet, and Charles L. Wilson. Nist form-based handprint recognition system. Technical report, Technical Report NISTIR 5469 and CD-ROM, National Institute of Standards and Technology, 1994.
- [GCB09] Venu Govindaraju, Huaigu Cao, and Anurag Bhardwaj. Handwritten document retrieval strategies. In *AND '09: Proceedings of The Third Workshop on Analytics for Noisy Unstructured Text Data*, pages 3–7, New York, NY, USA, 2009. ACM.
- [GLF<sup>+</sup>09] A. Graves, M. Liwicki, S. Fernandez, R. Bertolami, H. Bunke, and J. Schmidhuber. A novel connectionist system for unconstrained handwriting recognition. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 31(5):855 –868, may. 2009.
- [GSL09] Basilis Gatos, Nikolaos Stamatopoulos, and Georgios Louloudis. Icdar 2009 handwriting segmentation contest. In *Proceedings of the 2009 10th International*

*Conference on Document Analysis and Recognition, ICDAR '09*, pages 1393–1397, Washington, DC, USA, 2009. IEEE Computer Society.

[GW07] Rafael C. Gonzalez and Richard E. Woods. *Digital Image Processing (3rd Edition)*. Prentice Hall, 2007.

[HAI07] T. Hamamura, T. Akagi, and B. Irie. An analytic word recognition algorithm using a posteriori probability. *Document Analysis and Recognition, International Conference on*, 2:669–673, 2007.

[HB97] Thien M. Ha and Horst Bunke. Off-line, handwritten numeral recognition by perturbation method. *IEEE Trans. Pattern Anal. Mach. Intell.*, 19(5):535–539, 1997.

[HBS09] M. Mehdi Haji, Tien D. Bui, and Ching Y. Suen. Simultaneous document margin removal and skew correction based on corner detection in projection profiles. In *ICIAP '09: Proceedings of the 15th International Conference on Image Analysis and Processing*, pages 1025–1034, Berlin, Heidelberg, 2009. Springer-Verlag.

[HBS11] Mehdi Haji, T. D. Bui and C. Y. Suen, “Automatic extraction of numeric strings in unconstrained handwritten document images,” *Document Recognition and Retrieval XVIII*, Proceedings of SPIE, Volume 7874, Jan. 2011.

[HBS12] Mehdi Haji, Tien D. Bui and Ching Y. Suen, “Removal of Noise Patterns in Handwritten Document Images Using Expectation Maximization and Fuzzy Inference Systems,” *Pattern Recognition*, 2012. (under second review)

[HK06] Sun-Kyoo Hwang and Whoi-Yul Kim. A novel approach to the fast computation of zernike moments. *Pattern Recogn.*, 39(11):2065–2076, 2006.

- [HP02] Y.C. Ho and D.L. Pepyne. Simple explanation of the no-free-lunch theorem and its implications. *Journal of Optimization Theory and Applications*, 115:549–570, 2002. 10.1023/A:1021251113462.
- [HSB+12] Mehdi Haji , Kalyan A. Sahoo, Tien D. Bui, Ching Y. Suen and Dominique Ponson, “Statistical Hypothesis Testing for Handwritten Word Segmentation Algorithms,” *2012 International Conference on Frontiers in Handwriting Recognition (ICFHR-2012)*. (submitted)
- [JS97] J.-S. R. Jang and C.-T. Sun. *Neuro-Fuzzy and Soft Computing: A Computational Approach to Learning and Machine Intelligence*. Prentice Hall, 1997.
- [JVVYV98] Lucas J Van Vliet, Ian T. Young, and Piet W. Verbeek. Recursive gaussian derivative filters. In *Proceedings of the 14th International Conference on Pattern Recognition-Volume 1 - Volume 1*, ICPR '98, pages 509–514, Washington, DC, USA, 1998. IEEE Computer Society.
- [KA94] Shyh-Shiaw Kuo and O.E. Agazzi. Keyword spotting in poorly printed documents using pseudo 2-d hidden markov models. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 16(8):842–848, August 1994.
- [KAA<sup>+</sup>00] A. Kolcz, J. Alspector, M. Augusteijn, R. Carlson, and G. Viorel Popescu. A line-oriented approach to word spotting in handwritten documents. *Pattern Analysis & Applications*, 3:153–168, 2000. 10.1007/s100440070020.
- [Kav10] Ergina Kavallieratou. Text line detection and segmentation: uneven skew angles and hill-and-dale writing. In *Proceedings of the 2010 ACM Symposium on Applied Computing*, SAC '10, pages 59–60, New York, NY, USA, 2010. ACM.

- [KB06] D.J. Kennard and W.A. Barrett. Separating lines of text in free-form handwritten historical documents. In *Document Image Analysis for Libraries, 2006. DIAL '06. Second International Conference on*, pages 12 pp. –23, april 2006.
- [KBJO10] Alessandro L. Koerich, Alceu de S. Britto Jr., and Luiz Eduardo S. de Oliveira. Verification of unconstrained handwritten words at character level. In *Frontiers in Handwriting Recognition (ICFHR), 2010 International Conference on*, pages 39 –44, 2010.
- [KG97] Gyeonghwan Kim and Venu Govindaraju. A lexicon driven approach to handwritten word recognition for real-time applications. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19:366–379, 1997.
- [KPBH10] Yousri Kessentini, Thierry Paquet, and AbdelMajid Ben Hamadou. Off-line handwritten word recognition using multi-stream hidden markov models. *Pattern Recogn. Lett.*, 31(1):60–70, 2010.
- [KSS05] Alessandro L. Koerich, Robert Sabourin, and Ching Y. Suen. Recognition and verification of unconstrained handwritten words. *IEEE Trans. Pattern Anal. Mach. Intell.*, 27(10):1509–1522, 2005.
- [KvT05] Jon Kleinberg and Éva Tardos. *Algorithm Design*. Addison Wesley, 2005.
- [LGH07] G. Louloudis, B. Gatos, and C. Halatsis. Text line detection in unconstrained handwritten documents using a block-based hough transform approach. In *Proceedings of the Ninth International Conference on Document Analysis and Recognition - Volume 02, ICDAR '07*, pages 599–603, Washington, DC, USA, 2007. IEEE Computer Society.

- [LGP09] G. Louloudis, B. Gatos, I. Pratikakis, and C. Halatsis. Text line and word segmentation of handwritten documents. *Pattern Recognition*, 42(12):3169 – 3183, 2009. New Frontiers in Handwriting Recognition.
- [Liu07] Cheng-Lin Liu. Normalization-cooperated gradient feature extraction for handwritten character recognition. *IEEE Trans. Pattern Anal. Mach. Intell.*, 29(8):1465–1469, 2007.
- [LLE07] Yann Leydier, Frank Lebourgeois, and Hubert Emptoz. Text search for medieval manuscript images. *Pattern Recognition*, 40(12):3552 – 3567, 2007.
- [LNG00] Jia Li, A. Najmi, and R.M. Gray. Image classification by a two-dimensional hidden markov model. *Signal Processing, IEEE Transactions on*, 48(2):517–533, February 2000.
- [LNSF04] Cheng-Lin Liu, Kazuki Nakashima, Hiroshi Sako, and Hiromichi Fujisawa. Handwritten digit recognition: investigation of normalization and feature extraction techniques. *Pattern Recognition*, 37(2):265–279, February 2004.
- [LRM04] V. Lavrenko, T.M. Rath, and R. Manmatha. Holistic word recognition for handwritten historical documents. pages 278 – 287, 2004.
- [LSF94] L. Likforman-Sulem and C. Faure. *Advances in Handwriting and Drawing : a multidisciplinary approach*, chapter Extracting text lines in handwritten documents by perceptual grouping, pages 21–38. Europia, 1994.
- [LSHF95] L. Likforman-Sulem, A. Hanimyan, and C. Faure. A hough based algorithm for extracting text lines in handwritten documents. In *Proceedings of the Third International Conference on Document Analysis and Recognition (Volume 2) - Volume 2*, ICDAR '95, pages 774–, Washington, DC, USA, 1995. IEEE Computer Society.

- [LSZT07] Laurence Likforman-Sulem, Abderrazak Zahour, and Bruno Taconet. Text line segmentation of historical documents: a survey. *Int. J. Doc. Anal. Recognit.*, 9:123–138, April 2007.
- [LTW96] D. X. Le, G. R. Thoma, and H. Wechsler. Automated borders detection and adaptive segmentation for binary document images. In *Proceedings of the International Conference on Pattern Recognition (ICPR '96) Volume III-Volume 7276 - Volume 7276*, ICPR '96, pages 737–, Washington, DC, USA, 1996. IEEE Computer Society.
- [LZD<sup>+</sup>08a] Yi Li, Yefeng Zheng, D. Doermann, S. Jaeger, and Yi Li. Script-independent text line segmentation in freestyle handwritten documents. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 30(8):1313–1329, aug. 2008.
- [LZD<sup>+</sup>08b] Yi Li, Yefeng Zheng, David Doermann, Stefan Jaeger, and Yi Li. Script-independent text line segmentation in freestyle handwritten documents. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 30:1313–1329, 2008.
- [MB02] U.-V. Marti and H. Bunke. The iam-database: an english sentence database for offline handwriting recognition. *International Journal on Document Analysis and Recognition*, 5(1):39–46, 2002.
- [MCS06] Jonathan Milgram, Mohamed Cheriet, and Robert Sabourin. One against one or one against all: Which one is better for handwriting recognition with svms? In Guy Lorette, editor, *Tenth International Workshop on Frontiers in Handwriting Recognition*, La Baule (France), 10 2006. Universite de Rennes, Suvisoft.
- [MDES09] Walid Magdy, Kareem Darwish, and Motaz El-Saban. Efficient language-independent retrieval of printed documents without ocr. In Jussi Karlgren, Jorma Tarhio,

and Heikki Hyyrö, editors, *String Processing and Information Retrieval*, volume 5721 of *Lecture Notes in Computer Science*, pages 334–343. Springer Berlin / Heidelberg, 2009. 10.1007/978-3-642-03784-9\_33.

[MG00] M.A. Mohamed and P. Gader. Generalized hidden markov models. i. theoretical frameworks. *Fuzzy Systems, IEEE Transactions on*, 8(1):67–81, February 2000.

[MGB09] Robert Milewski, Venu Govindaraju, and Anurag Bhardwaj. Automatic recognition of handwritten medical forms for search engines. *International Journal on Document Analysis and Recognition*, 11:203–218, 2009. 10.1007/s10032-008-0077-1.

[MMM00] B. Merialdo, S. Marchand-Maillet, and B. Huet. Approximate viterbi decoding for 2d-hidden markov models. In *Acoustics, Speech, and Signal Processing, 2000. ICASSP '00. Proceedings. 2000 IEEE International Conference on*, volume 6, pages 2147–2150 vol.4, 2000.

[MR05] R. Manmatha and Jamie L. Rothfeder. A scale space approach for automatically segmenting words from historical handwritten documents. *IEEE Trans. Pattern Anal. Mach. Intell.*, 27(8):1212–1225, 2005.

[MS99] Christopher D. Manning and Hinrich Schütze. *Foundations of Statistical Natural Language Processing*. MIT Press, 1999.

[Neg04] Michael Negnevitsky. *Artificial Intelligence: A Guide to Intelligent Systems (2nd Edition)*. Addison Wesley, 2004.

[NSK07] B. Gatos N. Stamatopoulos and A. Kesidis. Automatic borders detection of camera document images. In *2nd International Workshop on Camera-Based Document Analysis and Recognition (CBDAR'07)*, Curitiba, Brazil, 2007.

- [NSV92] George Nagy, Sharad Seth, and Mahesh Viswanathan. A prototype document image analysis system for technical journals. *Computer*, 25:10–22, July 1992.
- [OC02] Frank Hoffmann Luis Magdalena Oscar Cordon, Francisco Herrera. *Genetic Fuzzy Systems: Evolutionary Tuning and Learning of Fuzzy Knowledge Bases (Advances in Fuzzy Systems - Applications & Theory)*. World Scientific Publishing Company, 2002.
- [O’G93] L. O’Gorman. The document spectrum for page layout analysis. *IEEE Trans. Pattern Anal. Mach. Intell.*, 15:1162–1173, November 1993.
- [OS06] Jose Oncina and Marc Sebban. Learning stochastic edit distance: Application in handwritten character recognition. *Pattern Recognition*, 39(9):1575 – 1587, 2006.
- [PD03] U. Pal and Sagarika Datta. Segmentation of bangla unconstrained handwritten text. In *Proceedings of the Seventh International Conference on Document Analysis and Recognition - Volume 2, ICDAR ’03*, pages 1128–, Washington, DC, USA, 2003. IEEE Computer Society.
- [PK04] W. Peerawit and A. Kawtrakul. Marginal noise removal from document images using edge density. In *4th Information and Computer Engineering Postgraduate Workshop, Phuket, Thailand*, 2004.
- [PTVF07] William H. Press, Saul A. Teukolsky, William T. Vetterling, and Brian P. Flannery. *Numerical Recipes 3rd Edition: The Art of Scientific Computing*. Cambridge University Press, 3 edition, August 2007.



- [Rab89] L.R. Rabiner. A tutorial on hidden markov models and selected applications in speech recognition. *Proceedings of the IEEE*, 77(2):257–286, February 1989.
- [RDL<sup>+</sup>92] Aaron E. Rosenberg, Joel DeLong, Chin-Hui Lee, Biing-Hwang Juang, and Frank K. Soong. The use of cohort normalized scores for speaker verification. In *Proceedings of the International Conference on Spoken Language Processing (ICSLP'92)*, pages 599–602, 1992.
- [RMKI09] Amjad Rehman, Dzul kifli Mohamad, Fajri Kurniawan, and Mohammad Ilay. Performance analysis of segmentation approach for cursive handwriting on benchmark database. *Computer Systems and Applications, ACS/IEEE International Conference on*, 0:265–270, 2009.
- [Rok10] Lior Rokach. Ensemble-based classifiers. *Artif. Intell. Rev.*, 33:1–39, February 2010.
- [RP08] Jose A. Rodriguez and Florent Perronnin. Local gradient histogram features for word spotting in unconstrained handwritten documents. In *Proceedings of the 1st International Conference on Handwriting Recognition (ICFHR'08)*, August 2008.
- [RQD00] Douglas A. Reynolds, Thomas F. Quatieri, and Robert B. Dunn. Speaker verification using adapted gaussian mixture models. *Digital Signal Processing*, 10(1-3):19–41, 2000.
- [RSP09a] J.A. Rodriguez-Serrano and F. Perronnin. Handwritten word image retrieval with synthesized typed queries. pages 351–355, jul. 2009.

- [RSP09b] Jose A. Rodriguez-Serrano and Florent Perronnin. Handwritten word-spotting using hidden markov models and universal vocabularies. *Pattern Recognition*, 42(9):2106–2116, 2009.
- [RSPSL10] Jose A. Rodriguez-Serrano, Florent Perronnin, Gemma Sanchez, and Josep Lladós. Unsupervised writer adaptation of whole-word hmms with application to word-spotting. *Pattern Recognition Letters*, 31(8):742–749, 2010.
- [SC78] H. Sakoe and S. Chiba. Dynamic programming algorithm optimization for spoken word recognition. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 26:43–49, 1978.
- [SKS96] Giovanni Seni, V. Kripásundar, and Rohini K. Srihari. Generalizing edit distance to incorporate domain information: Handwritten text recognition as a case study. *Pattern Recognition*, 29(3):405 – 414, 1996.
- [SLYT07] Te-Hsiu Sun, Chih-Chung Lo, Po-Shen Yu, and Fang-Chih Tien. Boundary-based corner detection using k-cosine. pages 1106 –1111, oct. 2007.
- [SM02] Klaus U. Schulz and Stoyan Mihov. Fast string correction with levenshtein automata. *International Journal on Document Analysis and Recognition*, 5:67–85, 2002. 10.1007/s10032-002-0082-8.
- [SPJ97] Anikó Simon, Jean-Christophe Pret, and A. Peter Johnson. A fast algorithm for bottom-up document layout analysis. *IEEE Trans. Pattern Anal. Mach. Intell.*, 19:273–277, March 1997.
- [SSG05] Zhixin Shi, Srirangaraj Setlur, and Venu Govindaraju. Text extraction from gray scale historical document images using adaptive local connectivity map. In

*ICDAR '05: Proceedings of the Eighth International Conference on Document Analysis and Recognition*, pages 794–798, Washington, DC, USA, 2005. IEEE Computer Society.

[SSG09] Zhixin Shi, Srirangaraj Setlur, and Venu Govindaraju. A steerable directional local profile technique for extraction of handwritten arabic text lines. In *ICDAR '09: Proceedings of the 2009 10th International Conference on Document Analysis and Recognition*, pages 176–180, Washington, DC, USA, 2009. IEEE Computer Society.

[SvBKB07] Faisal Shafait, Joost van Beusekom, Daniel Keysers, and Thomas Breuel. Page frame detection for marginal noise removal from scanned documents. In Bjarne Ersbøll and Kim Pedersen, editors, *Image Analysis*, volume 4522 of *Lecture Notes in Computer Science*, pages 651–660. Springer Berlin / Heidelberg, 2007. 10.1007/978-3-540-73040-8\_66.

[SvBKB08] Faisal Shafait, Joost van Beusekom, Daniel Keysers, and Thomas Breuel. Document cleanup using page frame detection. *International Journal on Document Analysis and Recognition*, 11:81–96, 2008. 10.1007/s10032-008-0071-7.

[SWB<sup>+</sup>07] Thomas Serre, Lior Wolf, Stanley Bileschi, Maximilian Riesenhuber, and Tomaso Poggio. Robust object recognition with cortex-like mechanisms. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 29:411–426, 2007.

[VB08] Tamás Varga and Horst Bunke. Perturbation models for generating synthetic training data in handwriting recognition. In Simone Marinai and Hiromichi Fujisawa, editors, *Machine Learning in Document Analysis and Recognition*, volume 90 of *Studies in Computational Intelligence*, pages 333–360. Springer Berlin / Heidelberg, 2008. 10.1007/978-3-540-76280-5\_13.

- [vdZSH08] T. van der Zant, L. Schomaker, and K. Haak. Handwritten-word spotting using biologically inspired features. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 30(11):1945–1957, nov. 2008.
- [vES00] Merijn van Erp and Lambert Schomaker. Variants of the borda count method for combining ranked classifier hypotheses. In *Seventh International Workshop on Frontiers in Handwriting Recognition*, pages 443–452, 2000.
- [VGC01] B. Verma, P. Gader, and W. Chen. Fusion of multiple handwritten word recognition techniques. *Pattern Recognition Letters*, 22(9):991–998, 2001.
- [WDL05] Xuewen Wang, Xiaoqing Ding, and Changsong Liu. Gabor filters-based feature extraction for character recognition. *Pattern Recognition*, 38(3):369–379, 2005.
- [Wei04] Jie Wei. Markov edit distance. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 26(3):311–321, 2004.
- [WL75] Robert A. Wagner and Roy Lowrance. An extension of the string-to-string correction problem. *J. ACM*, 22:177–183, April 1975.
- [xDKSP05] Jian xiong Dong, Adam Krzyzak, Ching Y. Suen, and Dominique Ponson. Low-level cursive word representation based on geometric decomposition. In *International Conference on Machine Learning and Data Mining (MLDM'05)*, pages 590–599, 2005.
- [ZEP10] Konstantinos Zagoris, Kavallieratou Ergina, and Nikos Papamarkos. A document image retrieval system. *Engineering Applications of Artificial Intelligence*, 23(6):872–879, 2010.
- [ZJ10] Xin Zhang and Xili Jing. Image denoising in contourlet domain based on a normal inverse gaussian prior. *Digit. Signal Process.*, 20(5):1439–1446, 2010.

- [ZS84] T. Y. Zhang and C. Y. Suen. A fast parallel algorithm for thinning digital patterns. *Commun. ACM*, 27:236–239, March 1984.
- [ZT01] Zheng Zhang and Chew Lim Tan. Recovery of distorted document images from bound volumes. *Document Analysis and Recognition, International Conference on*, 0:0429, 2001.
- [ZTMR01] A. Zahour, B. Taconet, P. Mercy, and S. Ramdane. Arabic hand-written text-line extraction. In *Document Analysis and Recognition, 2001. Proceedings. Sixth International Conference on*, pages 281 –285, 2001.