A HYBRID 2-D HMM AND MLP OCR SYSTEM FOR PROCESSING MULTI-FONT AND LOW-QUALITY ENGLISH DOCUMENTS

Nenghong Fu

A thesis
in
The Department
of
Computer Science

Presented in Partial Fulfillment of the Requirements
For the Degree of Master of Computer Science
Concordia University
Montréal, Québec, Canada

April 2004

Your file Votre référence
ISBN: 0-612-91032-6
Our file Notre référence
ISBN: 0-612-91032-6

Abstract

A Hybrid 2-D HMM and MLP OCR System for Processing Multi-Font and Low-quality English Documents

Nenghong Fu

Optical Character Recognition (OCR) has been researched for more than 50 years. To design a system to recognize clean printed documents with a higher recognition rate is not difficult today. However, it is still a challenge to develop an OCR system which can maintain a high recognition rate, regardless of the quality of the input documents and the fonts used.

This thesis presents a Hybrid 2-*Direction*(D) Hidden Markov Model (2-D HMM) and Multi-Layer Perceptron (MLP) OCR system for the recognition of Multi-font printed documents of varying qualities. It emphasizes on new methods proposed. First, a statistical analysis of the frequency of touching characters has been conducted, and some statistics of touching characters have been generated from real documents. Based on these statistical results which could be the first formal statistics on touching characters, a new classifier has been designed to recognize some frequent touching characters without segmentation. Second, a new hierarchical character classifier is presented to enhance character recognition accuracy. We group all characters into several categories according to character layout contextual information (Ascender, Descender and Center). Consequently we implement several independent classifiers to recognize the characters in each group.

In addition, a 2-D HMM is included in the hierarchical classifier to improve the character recognition rate, and an automatic builder of special touching character HMM is also described in this thesis. Finally, several techniques have been designed to improve the overall performance, e.g. hybrid 2-D HMM and MLP classifiers are used for character recognition, and some complementary sources of information are introduced in an integrated segmentation and recognition (ISR) module to recognize character strings, and an online lexicon is used for checking and correcting potential

classification errors. Based on the testing experiments, our proposed OCR system has achieved very promising performance compared with commercial OCR.

Acknowledgments

First of all, I would like to extend my sincere gratitude to my supervisor Dr. Ching Y. Suen for his guidance, encouragement, and help throughout my study at Concordia. The time and effort he dedicated to the project and my thesis are crucial and indispensable.

I am grateful to people whose studies have directly contributed to my thesis. Among them are Yun Li and Andrea Barretto de Souza whose preprocessing code libraries were used in my program, Nicola Nobile whose online lexicon function DLL was used in my program, Karim Abou-Moustafa who has provided me parts of his original 1-D HMM classes, and Dr. Qizhi Xu who has provided me her MLP classifier, orginal ISR and DP source codes and shared with me her knowledge and research experience in the field, Guiling Guo who is involved in building the database.

Special thanks should go to my colleagues Dr. Incheol Kim, Dr. Jiangxiong Dong who have given me their help at different times and shared my emotions.

I wish to thank all Cenparmians who helped me in one way or another since I joined CENPARMI: Beverly Abramovitz, Jun Zhou, Chunlei He, Ji Na Tan, Wumo Pan, Yan Zhang, Javad Sadri, etc.

Last but not least, I am indebted to my parents who brought me up with endless love and unbounded understanding. To my father-in-law and mother-in-law, they gave me very important help by taking care of my daughter for one year. To Ying Wei, my beloved wife, I am so grateful for her effort of always being there supporting me. To my daughter, Christina Xinyu Fu, I am so glad to have her during my study, and she has brought me a lot of happiness. Their care, understanding and encouragement enabled me to finish this thesis.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

During the past 50 years, substantial research efforts have been devoted to Optical Character Recognition (OCR) [1, 2]. The object of OCR is automatic reading of optically sensed document text materials (machine-printed or handwritten text) to translate human-readable characters into machine-readable codes (such as ASCII codes). OCR research using electronic and electro-mechanic methods dates back to the 1950's. This domain is one of the oldest research areas in the field of pattern recognition and it was initially directed toward machine-prints and was restricted to fixed-format applications. Today, commercial OCR packages are already available for clean machine-printed text documents with simple layouts, limited fonts and languages. Considerable work has also been done for multi-font and multi-languages (English, Chinese, Japanese, Korean, Arabic, Telugu, etc.) and some successful systems have also been developed to recognize handwritten texts. However, the analysis of documents with complex layouts, recognition of degraded machine-printed texts, and the recognition of unconstrained handwritten texts continue to require further research and improvements. This thesis aims at developing a hybrid OCR system for recognizing multi-font, degraded and machine-printed English texts. The methodologies presented in this thesis can also be applied to the OCR systems of wider scope, especially the proposed Integrated Segmentation Recognition module (ISR), 2-*Direction* Hidden Markov Model (2D-HMM) and automatic construction of touching character HMMs.

1.1 The Motivation

Written documents have been existing for a long time and have been developed as a way to extend human memory and to facilitate communication. Nowadays, as information technology develops rapidly, some arguments are arising on whether paper documents will be necessary [3]. However, Writing is one of man's most significant cultural achievements. If you look into the cultural supplements of our newspapers, you realize that they do not get tired of celebrating writing as a cultural monument of mankind. It is sure that paper is still and will continue to be one of the most commonly used media for storing and transporting people's ideas. Therefore, we can say that paper-based documents will continue to play a vital role in the real world because of their confidential, portable and convenient advantages.

Is there a need for machine recognition of texts even paper documents will continue to be used? The answer is obvious. Although a document image can be displayed on screen (with clearly legible text) or printed, to the computer, the image is not readable and it can not be understood. OCR will make the text machine-readable and therefore should be considered for the following advantages:

- the text can be reused, edited and reformatted
- the text is available for full-text information retrieval (e.g., by internet search engines)
- the text can be coded in HTML or XHTML
- the text is available to adaptive equipment for the visually impaired
- the file size is of concern (in terms of storage or bandwidth to transmit)

As long as paper documents are used, there will be a need for the use of OCR. In addition, as the development of advanced video and image annotation and retrieval system and the application of E-learning, applying OCR technology is absolutely needed. In fact, many applications can be found in this area, such as financial document processing (cheques, credit card slips, etc.), postal address recognition for mail sorting, form (tax forms, census forms, etc.) processing, and establishment of digital libraries and other web-based OCR services, etc.

OCR addresses the above needs and has been investigated for many years [4, 5], particularly for machine-printed documents. Today, OCR software for the recognition of clean machine-printed texts has achieved higher performance. However, recognition rate for multi-font, multi-language and degraded machined printed texts is still low varying widely for multi-font, multi-language and low quality images [6]. Therefore, developing a multi-font OCR system with higher performance for the recognition of both clean and degraded machine-printed English documents is necessary and valuable. In addition, even Hidden Markov Model (HMM) and Multi Layer Percetron (MLP) neural networks have been widely used in OCR system, there is still a strong motivation to investigate new methods on the application of HMM and MLP. This is the focus of our research.

1.2 The Challenge

Developing a multi-font OCR system for the recognition of both clean and degraded machine-printed English documents is extremely challenging due to the difficulties of recognition of touching characters, confused characters and broken characters. Studies in [7, 8] showed that touching and broken characters seem to be the most important source of OCR problems.

1. Touching and Confused Characters
   When characters in a page are thickened by distortion, it tends to create many touching characters, especially for degraded documents. Another byproduct of characters with thick strokes is that the loops in letters 'a', 'e' and 'o' often get filled up completely or present only a minimal white portion in the center and produces some confused characters. Thus, the difficulty in recognizing the confused characters is increased. As shown in Figure 1, because of their thickness, touching characters and confused characters occur more often in degraded documents.

Figure 1: Touching and confused characters increase the recognition difficulty of OCR system

2. Broken characters occur very often in degraded documents. They are usually fragmented into small pieces, and character fragments could take any shape. This causes extreme difficulty for OCR system to recognize broken characters. As shown in Figure 2, fragments of broken characters ('3', '4', 'w', 's', 'A', 'v', etc) have made any OCR system difficult to recognize them correctly.

Figure 2: Broken characters increase the recognition difficulty of OCR system

The challenge also comes from the assessment of image quality. Basically, for a given clean printed document, most OCR systems can achieve a high performance. Even for some degraded documents, if the OCR systems have known the degraded information in advance, the OCR systems can also achieve acceptable performance by applying some image enhancement techniques corresponding to the degraded information (e.g., broken or touching characters).

Since it is difficult to correctly estimate the image quality and apply the corresponding restoration algorithm, it is hard to achieve a high performance for the recognition of degraded printed documents. Therefore, an image quality estimator would be essential to the operation of an OCR system. The quality information would be used to select an appropriate classifier or a set of weights for combining results obtained from a multiple classifier. In addition, meeting the demanding industrial requirements such as high processing speed, robustness and higher recognition rate, poses a big challenge in the development of OCR systems.

1.3 Thesis Organization

The first chapter introduces the motivation and challenge. A review of the state of the art for off-line printed recognition is given in Chapter 2. Chapter 3 will briefly describe an overview of the proposed OCR system and its components.

Chapter 4 will introduce the databases used in our OCR system, and also talk about the frequency statistics of touching characters.

Chapter 5 will mainly present feature extraction, vector quantization, and the most important part of our 2-*Direction* (D) Character Based HMM. In addition, an

automatic touching character HMM building method will be introduced. Moreover, Forward-Backward algorithm, *Baum-Welch* algorithm and *Viterbi* algorithm will be introduced briefly.

Chapter 6 will first give a brief view of MLP classifier and the combination with HMM classifier. Next, dynamic programming (DP) based touching character string recognition will be presented. Last, our hybrid 2-D HMM and MLP system, online lexicon will be described.

Chapter 7 will give more detailed experimental results for isolated character recognition, document recognition and comparison with commercial OCR systems.

Finally, Chapter 8 will list the major contributions of this thesis and the future research.

Chapter 2

State of the Art

2.1 Recognition Methods

Over the years, many systems have been developed for machine-printed document recognition. These systems use the methodologies of pattern recognition to assign an unknown sample to a predefined class. In the following discussion, general recognition techniques and their applications to machine-printed document recognition will be presented first.

2.1.1 General Recognition Techniques

Many techniques for machine-printed document recognition have been investigated based on three general approaches of pattern recognition: (i) template matching, (ii) statistical technique, and (iii) neural networks.

- Template Matching: Generally speaking, matching operations determine the degree of similarity between two vectors (groups of pixels, shapes, curvatures, etc.) in the feature space. Applications for machine-printed document recognition can be found in [9, 10, 11, 12, 13].

- Statistical Techniques: Statistic decision theory is concerned with statistical decision functions and a set of optimality criteria, which determine the probability of the observed pattern belonging to a certain class. Several popular recognition approaches belong to this domain:

1. The k-Nearest-Neighbor (k-NN) rule is a popular non-parametric recognition method, in which the a-*posteriori* probability is estimated from frequency of nearest neighbors of the unknown patterns. As a special case of k-NN rule, the nearest-neighbor rule classifies the unknown pattern to the class of the nearest sample. Higher performance has been achieved in [14, 15] by using this approach for machine-printed English document recognition.

2. Hidden Markov Model (HMM) is one of the most widely and successfully used techniques for machine-printed document recognition. It is defined as a stochastic process generated by two interrelated mechanisms: a Markov Chain having a finite number of states and a set of random functions, each of which is associated with a state [16, 17]. Basically, this approach builds one or more HMMs (character models) for each class of pattern and then builds word and sentence model by using each character model to estimate the maximum path probability for a given observed pattern. The main advantages of HMMs are: (i) the ability to model time-sequence data, (ii) segmentation-free recognition, and (iii) the ability to build any language model based on character models. Many applications of HMMs for machine-printed recognition have been reported in [18, 19, 20, 21, 22, 23, 24, 25].

3. Artificial Neural Networks (NN): An NN is defined as a computing structure consisting of a massively parallel interconnection of adaptive "neural" processors. NNs have been widely used in printed recognition problems and have achieved promising results in the past years [26, 27, 28]. One of the most popular networks used in this field is multi-layer perceptron networks (MLP).

   MLP was proposed by Rosenblatt [29] and elaborated by Minsky and Papert [30]. The perceptron, as the building block of MLP network, forms a weighted sum of n components of the input vector and adds a bias value. The results are then passed through a nonlinear activation function. MLP networks trained by back propagation are among the most popular and versatile forms of neural network classifiers.

2.1.2 Combinations

The combination of multiple classifiers has proved to be a powerful technique in many areas of pattern recognition research for the past decades. Some good recent surveys on this topic are [31, 32, 33, 34, 35]. Many combination techniques can be grouped and analyzed in different ways. In terms of implementation, combination methods can be classified into four architectures: (i) conditional, (ii) serial (hierarchical), (iii) parallel (multiple), and (iv) hybrid.

- Conditional Architecture: In this architecture, a primary classifier is first used. When it rejects a pattern, a secondary classifier, which can use more complex procedures, is adopted. This architecture is usually efficient in terms of computation.

- Serial Architecture: In this architecture, classifiers are applied in succession. The goal of each classifier is to reduce the number of classes such that the individual classifiers can be focused increasingly.

- Parallel Architecture: Using this architecture, all classifiers are first applied concurrently and independently. The decisions of the classifiers are then combined to produce a final decision according to the classification result of each classifier. There are three combination methods used in this architecture:

    1. Abstract level: the classifiers output only one label corresponding to the class to which the input pattern is thought to belong. The most straightforward way to combine such results is via a simple majority voting rule, whereby the result favoured by the majority of the classifiers is accepted. Another method of combining abstract classification results is the behaviour-knowledge space (BKS) method proposed by Huang and Suen [36]
    2. Rank order level: The classifier ranks all possible classes in order of likelihood.
    3. Measurement level: The classifier assigns a measurement value to each class reflecting the probability that the input pattern belongs to that class. some simple combination rules can be applied at this level, such as product rule, sum rule, min rule, median rule, max rule, and Bayesian rule [37, 35].

- Hybrid Architecture: This architecture is designed to combine the power of previous architectures.

More discussions about combination methods will be given in Section 6.2.2.

2.2 String Recognition

Though recognition of isolated characters or words has been studied extensively in the literature, research on string recognition is still limited. We summarize research on this topic and categorize it into two types: segmentation-based and segmentation-free methods:

1. Segmentation-Based method: This method tries to segment the string into its constituent words by applying various sentence-to-word segmentation followed by recognition of the segments. Each word segment is then segmented to a number of hypotheses which represent likely character segmentations of the word and a dynamic programming search technique is applied to find the best scores of hypotheses for each word segment or each string. The most difficult problem of this method is segmentation of touching characters. Much research has been done in segmentation of touching characters in the past decades [38, 39, 40, 41, 42, 43, 44, 45]. Applications based on this method can be found in [14, 15, 46, 26, 27] with better performance.

2. Segmentation-Free method: This method tries to recognize the entire string as a single unit and has been widely used in OCR systems for printed documents. The core of the recognition procedure is an HMM. It receives a sequence of observations generated from feature vectors and outputs a sequence of words, applying constraints from the language model. The feature vectors input to the HMM are usually extracted from a complete text line of image based on the sliding window technique. Typically, the HMM has a sentence model built from a word model based on a character model. The character model of HMM is built and trained on non-segmented data with ground truth. Thus no segmentation is required for the recognition of a text line and the segmentation of a text line into words is obtained automatically during the recognition stage based on *Viterbi* algorithm. Many applications in printed recognition based on this method have been developed [18, 19, 20, 21].

2.3 Survey of published OCR systems for machine-printed document recognition

Zhi-Dan Feng and Qiang Huo developed a confidence guided progressive search and fast match technique for clean printed Chinese/English OCR system [15]. This system is a segmentation-based OCR system. The character recognizer of this system is a multiple prototype based Nearest-Neighbor classifier using Euclidean distance and the character verification models are also k-Nearest-Neighbor classifier. For each input character line image, it first tries to construct the segmentation graph for the whole line. Then the recognition engine dynamically searches the best path from left to right based on confidence measurements. Each confidence value is taken from the combination of recognition result and verification result using predefined rules. If the confidence value of each hypothesized image is not acceptable, then an over-segmentation step is used and the same procedure is applied again to find the best path. In order to speedup the searching process, a fast match tree technique is employed to speedup the matching process. Finally, the recognition result is refined through a post-processing module. Experimental result of this system shows that a 99.11% recognition rate is achieved for a total of 1862 mixed Chinese/English text lines . In another paper [14], they proposed a method using MCE-based character-pair modelling and negative training techniques to improve their OCR performance further with a recognition speed of 134-204 per second.

Issam Bazzi et al designed an Omnifont Open-Vocabulary OCR system using HMM for clean printed English and Arabic [21, 19]. This system focuses on three main points. First, it is intended to be language-independent. Second, the training and recognition are performed using an existing continuous speech recognition system (BYBLOS), with no modification. Third, no presegmentation is performed - neither at the character nor at the word levels. Thus it is a segmentation-free approach. This system consists of 5 steps: (i)preprocessing of the scanned-text training data coupled with ground truth, (ii) feature extraction based on sliding window technique, (iii) building character model of HMM from the feature vectors and corresponding ground truth, (iv) training using forward-backward algorithm, and (v) recognition using different knowledge sources estimated in the training (character model, lexicon, and grammar) to find the best character sequence. Experimental results of this system

show that 2.1% and 0.8% character error rate (CER) are achieved without and with 30k-lexicon, respectively. In addition, They also applied this system for degraded documents by using degraded training data (fax data), and 0.6% CER is achieved on clean data and 2.7% on fax data [20].

Marcel Brun et al proposes a multi-resolution classification tree in OCR system in [10, 47]. Two important ideas, the morphological operators and classification trees, are used for character recognition. The recognition process is viewed as a multi-step procedure, that is, objects are classified into subclasses and then each of the subclasses is further classified into more subclasses, and this process is repeated for each subclass until a full classification is performed. The designs of both the morphological operators and the classification trees are based on training from sample pairs of observed-ideal images. In order to apply the classification tree for both training and recognition, images need to be adequately prepared. The characters need to be segmented and labelled in the natural order of occurrence in the image. Experimental results show that 98.43% of characters on clean English documents are correctly recognized and 88.58% on noised documents.

Chapter 3

Proposed Character Recognition System

Considering the challenge of recognition of multi-font degraded machine printed documents, we have proposed a hybrid 2-D HMM and MLP OCR system for recognition of English documents printed in multi-font and varying qualities.

3.1 System Overview

The overall architecture of our proposed system for recognition of machine-printed English documents of multi-font and varying qualities is shown in Figure 3.

This system includes three basic modules corresponding to pre-processing, recognition, and post-processing. More descriptions are given below.

3.2 Pre-processing

Preprocessing is very important to any recognition system, and it also plays a significant role in our system. The input of the preprocessing module is a binarized character line image. In this module, detection of connected components, skew detection and correction, and detection of reference lines [48, 49] are first performed. For noise filtering, a new algorithm related to automatic filter selection using image quality assessment has been proposed and very promising results have been produced on degraded documents [48]. After filtering, word detection and pre-segmentation

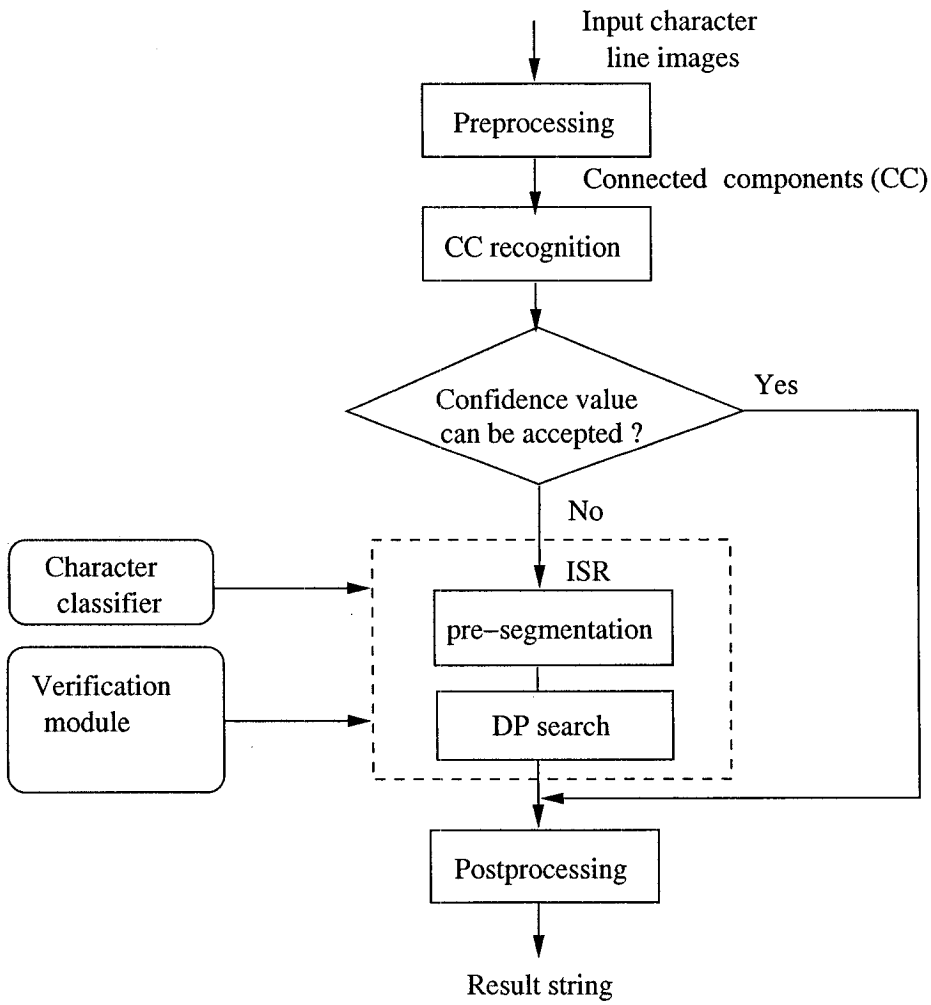


Figure 3: Diagram of System Overview

step are performed.

The pre-segmentation module can segment italic or slanted connected components by using slant projection, contour analysis and finding shortest paths [40, 50]. Figures 4, 5 and 6 list some examples related to pre-processing.

3.3 Classification

Classification is the kernel part of our system. It includes feature extraction, a hybrid 2-D HMM and MLP classifier and a verification module. There are two sets of features extracted from each image. One set of features is pixel density feature and contour distance feature used in the HMM classifier. The other is mesh and gradient features

Figure 4: Samples before and after pre-processing(filtering)

applied in the MLP classifier.

The key part of the classification module is a hybrid 2-D HMM and MLP classifier which is different from other OCR systems described in [14, 19, 21, 20, 15, 46]. We also combine character based HMMs with MLP classifier. However, we see HMM as a recognizer, and combine it with MLP classifier in each recognition stage based on sum rule of confidence value. More details will be presented in Chapters 4 and 5.

The inputs to the classification module are connected components (CC). A simple CC recognition module is first activated to recognize the CCs. This module basically consists of an isolated character classifier and a Touching character classifier.

The isolated character classifier is a hierarchical character classifier which combines 2-D HMM and MLP classifier based on character layout contextual information (ascender, descender and center). Unlike [51] and other papers in which the character

Figure 5: Image with reference lines detected

Figure 6: Sample of pre-segmentation

layout information was used in different stages for different purposes, e.g. merging broken characters and verifying character recognition results, our hierarchical character classifier first classifies the input pattern into one of the subgroups of classes and then identifies its class in the subgroup. Based on the baseline (reference line) information obtained in the preprocessing stage, character contextual classes (ascender, descender and center) can be detected according to their locations with respect to the baselines, and four classifiers are implemented for different groups of characters.

The recognition result from the isolated character classifier will be accepted or rejected according to the corresponding confidence value. The rejected CC will be sent to the touching character classifier and similarly the result will be rejected or accepted. The rejection conditions of both the isolated classifier and the touching classifier have been set very strictly. If a CC cannot be recognized by the CC recognition module with a high confidence, the Integrated Segmentation and Recognition (ISR) module which includes pre-segmentation and dynamic programming (DP) search parts will be called to recognize it. In the meantime, a verification module is used to verify each character candidate.

On the other hand, in order to take advantage of character based HMMs, we also use a new automatic building method developed in our system to build special touching character HMMs (STCHMM) to recognize some special touching characters (e.g. 'fi', 'fl', 'ft', 'tl', etc.). More discussions about the character classifier and the ISR module will be given in Chapters 5 and 6.

3.4 Post-processing

The outputs from the recognition stage, including character symbols and the corresponding confidence values for a character line image, are sent to the post-processing module. The kernel part of the post-processing module is an online lexicon which includes several functions and can be called by both the classification module and the post-processing module. A set of rules is being developed to utilize lexicon knowledge and n-gram statistics, etc. to detect and correct potential character recognition errors. Common errors produced by our document recognition system are also collected and used in developing the rules.

Chapter 4

Database

Three types of database are used in our OCR system. The first one is document database used for text line testing and validation. The second one is isolated character database used for training and testing of our character based HMM and MLP classifier. The last one is touching character database used for training and testing of our touching character classifier.

4.1 Document Database

The document databases used in our system include CENPARMI document database (CENPARMI-DOCDB) and Fujitsu document database (FUJITSU-DOCDB). In addition, another special Bigram database (CENPARMI-SPDB) has been used to test the performance of our classifier.

CENPARMI-DOCDB consists of 250 relatively clean documents binarized at 300 dpi collected from different sources with noise removed by commercial software. All documents are then automatically segmented into 14097 text line images. About 25 fonts and 10 types of sizes are found in this database.

CENPARMI-SPDB consists of 936 binarized clean text line images also scanned at good resolution. All text lines are composed of 24336 alphabet bigram characters ('aa', 'ab', ..., 'ZZ') (72072 characters) with 2 fonts, 2 sizes and 10 spacings(-2, -1, 0, auto, +1, +2, +3, +4, +5, +6) between two characters of each bigram character. Totally 34.4% touching characters are found in this database. Figure 7 shows some examples.

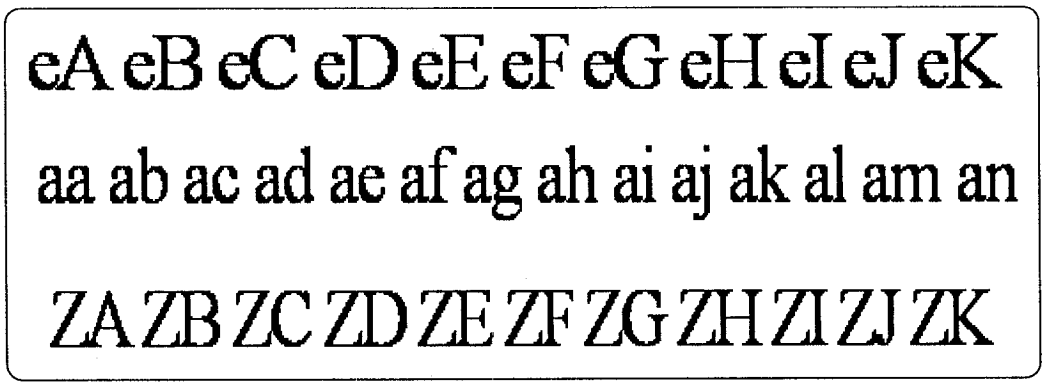


Figure 7: Samples of bigram characters in CENPARMI-SPDB database

FUJISTU-DOCDB includes 704 degraded documents provided by Fujitsu Laboratories Ltd. This database has three distinctive characteristics. First, each document contains only one printed line in English. There are no graphics, tables or drawings in any document. Second, a large variety of fonts, sizes and styles including some nonsense words, proper names and web URLs can be found among the documents. The third characteristic, and also the most important for our system, is that almost all images suffer from at least one of the following types of degradation:

- Broken characters
- Touching characters
- Salt-and-pepper noise

The distribution of type of degradation was analyzed manually and shown in Table 1. Figure 8 lists some sample images from both CENPARMI-DOCDB and FUJITSU-DOCDB.

4.2 Isolated Character Database

In order to recognize different fonts and styles of documents, we have built a multi-font artificial isolated database (CENPARMI-ISODB1) for the training and testing of our character based HMM and MLP classifiers. The CENPARMI-ISODB1 was generated from manual typing by typewriter or computer and scanning at 300 dpi smoothed by

Table 1: Distribution of type of degradation in FUJITSU-DOCDB

Type of Degradation	Distribution of Images	
	Number of Images	Percentage
Only Broken Characters	369	50.14%
Only Touching Characters	45	6.11%
Only Salt-and-Pepper Noise	17	2.31%
Broken,Touching characters with Salt-and-Pepper noise	220	34.14%
No Degradations	53	7.30%
Total	704	100%

commercial software. It consisted of 92 classes, 26 lower letters, 26 capital letters, 10 numerals, 30 punctuations, 20 fonts, with different sizes and 4 styles(normal, bold, italic and bold italic) for a total of 37940 images.

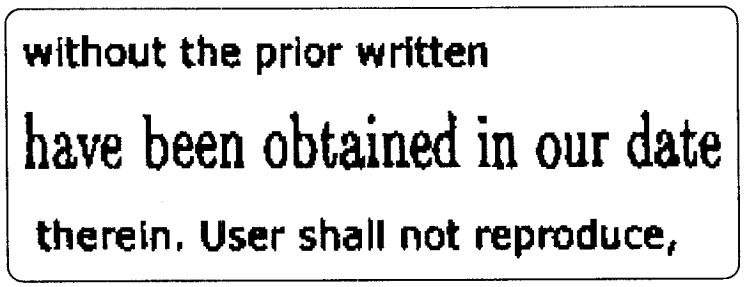
The training set currently used in our system consists of only 75 classes(62 alphabets and 13 common punctuations (! $ % & ( ) / >< ? @ [ ]) for a total of 16666 images from the original database. For some small punctuations(, ; : . = ' " etc.), we have used structural information, baseline information, size and position to recognize them. The testing set consisted of 11659 images.

In addition, we have also generated another isolated database(CENPARMI-ISODB2) extracted directly from real noised documents. Since for some characters we cannot find enough training samples from real documents, we have created some isolated character samples artificially and added some noise to them to simulate degraded data. It brought us 9239 training and 3559 testing samples but excluding punctuations. Therefore, the total training set for isolated character recognition contains 25905 samples (9239+16666) and the testing set consists of 15218 samples (11659+3559).

4.3 Touching Character Database

4.3.1 Statistics on touching characters

Segmenting touching characters (TC) is a challenging task of any OCR system using segmentation based approach. In order to improve the performance of recognizing touching characters, we have generated some statistics on touching characters [52]. About 264000 touching character components were extracted semi-automatically from



Sample images in CENPARMI–DOCDB database

Sample images in FUJITSU–DOCDB database

Figure 8: Sample images in CENPARMI-DOCDB and FUJITSU-DOCDB database

150 documents in CENPARMI-DOCDB. These touching character components were manually screened, and statistical results have been generated and summarized in Tables 2, 3, and 4.

Table 2: Distribution (%) of type of touching characters in CENPARMI-DOCDB (2 TC, 3 TC, ..., present 2, 3, 4, ..., characters touching together

percentage of TC	2 TC	3 TC	4 TC	>=5 TC
7.37	80.03	14.65	3.68	1.64

As far as we know that statistics on touching characters in printed documents have not been published before, and we believe that this statistical information is very useful in recognition and post-processing stages of document processing. We have built a new segmentation-free recognition module (named as Touching Character classifier) to recognize 57 most frequent touching characters (touching characters in

Table 3: Touching frequency (top 50 in descending order)

Order	TC	Freq.	Order	TC	Freq.	Order	TC	Freq.	Order	TC	Freq.
1	an	7.69	14	re	1.23	27	ty	0.79	40	gn	0.52
2	th	6.31	15	am	1.22	28	rs	0.77	41	ts	0.46
3	ar	3.52	16	th	1.22	29	co	0.77	42	oc	0.46
4	al	3.17	17	ri	0.92	30	rv	0.67	43	ai	0.46
5	as	1.81	18	po	0.92	31	ry	0.65	44	hi	0.45
6	be	1.76	19	te	0.92	32	tw	0.63	45	ee	0.45
7	rm	1.56	20	od	0.91	33	ro	0.61	46	wi	0.45
8	ed	1.51	21	ti	0.87	34	ff	0.61	47	tri	0.43
9	in	1.47	22	rn	0.86	35	art	0.61	48	to	0.43
10	pe	1.46	23	ta	0.86	36	ca	0.59	49	ak	0.39
11	fi	1.42	24	ru	0.85	37	ra	0.58	50	all	0.37
12	ec	1.37	25	ce	0.85	38	di	0.57			
13	es	1.36	26	se	0.83	39	ct	0.56			

Table 3 and "ur", "gy", "ul", "tru", "are", "the", and "ari"). The training set will be presented in the next section.

4.3.2 Most Frequent Touching Character database

The most frequent touching character database (CENPARMI-TPDB) is a by-product of our research of frequency statistics of touching characters mentioned above. In our system, we chose only 5444 images from 57 most frequent touching characters based on our statistic results to build 57 most frequent touching character database. The training set for the 57 most frequent touching characters contains 70 samples of each touching character for a total of 3936 samples from 4890 images. The testing set takes the rest of samples with average 15 samples of each touching character for a total of 954 samples.

Table 4: Contribution of touching characters in different positions

Character	1 Pos	2 Pos	3 Pos	Character	1 Pos	2 Pos	3 Pos
a	**23.25**	5.28	7.92	A	0.39	0.11	0.00
b	2.63	0.12	0.20	B	0.09	0.05	0.02
c	**4.28**	3.29	5.45	C	0.22	0.06	0.02
d	1.72	3.12	4.05	D	0.21	0.04	0.00
e	**7.32**	**11.49**	**12.72**	E	0.19	0.03	0.04
f	3.45	1.34	1.11	F	0.13	0.02	0.02
g	2.84	0.52	1.50	G	0.02	0.01	0.02
h	1.42	**9.09**	0.71	H	0.03	0.01	0.00
i	2.87	**8.30**	**14.07**	I	0.22	0.07	0.06
j	0.02	0.00	0.00	J	0.00	0.00	0.00
k	0.55	0.67	1.11	K	0.30	0.01	0.00
l	0.53	4.98	**7.96**	L	0.16	0.03	0.00
m	1.26	4.33	2.61	M	0.10	0.26	0.20
n	2.18	**13.33**	6.86	N	0.08	0.11	0.12
o	2.43	4.63	3.28	O	0.17	0.09	0.02
p	3.30	0.94	1.05	P	0.23	0.04	0.22
q	0.04	0.28	0.34	Q	0.02	0.00	0.00
r	**12.14**	**10.09**	5.87	R	0.18	0.09	0.02
s	2.43	6.82	**8.20**	S	0.16	0.04	0.06
t	**15.60**	2.85	**8.06**	T	1.46	0.07	0.06
u	2.46	2.21	2.55	U	0.04	0.03	0.02
v	0.48	0.83	0.32	V	0.02	0.01	0.02
w	1.02	0.87	1.11	W	0.10	0.01	0.00
x	0.46	0.31	0.28	X	0.00	0.02	0.00
y	0.45	2.30	1.66	Y	0.02	0.10	0.00
z	0.32	0.16	0.06	Z	0.02	0.01	0.00

Chapter 5

2-*Direction* character based HMM (2-D HMM)

Both discrete and continuous Hidden Markov Models (HMM) have been successfully applied to character recognition since its original application in speech recognition. Our 2-D HMM is based on discrete HMM (DHMM) because of its low computation cost [16]. For the discrete HMM, two essential issues related to its performance are feature extraction and vector quantization which will be presented in the next two sections. Other points related to our 2-D HMM will also be discussed in next sections with more details.

5.1 Feature Extraction

Feature extraction plays an important role in the pattern recognition domain. Selection of a good feature extraction method is an important step in achieving good performance of OCR system. Given the large number of feature extraction methods reported in the literature, a newcomer to the field is faced with a question: which feature extraction method is the best for a given application. An experimental evaluation method is the best way to select the best feature for a specific application. However, implementing all the feature extraction methods is an enormous task. In addition, the performance also depends on the type of classifier used. Different feature types may need different types of classifiers. Also, the classification results reported in the literature are not comparable because they are based on different data sets.

A feature extraction method that proves to be successful in one application domain may turn out not to be very useful in another domain.

In practice, the requirements of a good feature extraction method make selection of the best method for a given application a challenging task. Therefore, we have to find the suitable features for our 2-D HMM classifiers based on experimental evaluation. Several features have been tested and finally two complementary features have been selected in our 2-D HMM classifier for isolated character recognition based on the experimental results as follows:

1. Pixel density feature

   Given binarized character image, the following operations are done to extract the pixel density feature:

   - Normalization of character image: In order to keep the same dimensionality of each feature vector requested by our 2-D HMM classifier and make the feature invariant to variable size and shape distortion of characters, the image is normalized to have a height of 40 pixels and the width is changing accordingly to keep the aspect ratio. The normalization method used here is a very simple linear normalization method which is described as:

     $$m = x.\frac{M}{X} \; and \; n = y.\frac{N}{Y}$$

     x,y: Pixel coordinates of original image

     m,n: Pixel coordinates of normalized binary image mapping on the pixel (x, y)

     X,Y: Width and Height of original image

     M,N: Width and Height of normalized image ($N = 40, M = N.\frac{X}{Y}$)

   - A sliding window (height=40,width=3 and overlap=2) is used moving from left to right. From each sliding window (vertical strip) a feature vector v_h is extracted using the following criteria:

     $$\begin{aligned} r_i &= \frac{\#\ of\ Black\ Pixels}{Sliding\ Window\ Width} \quad 1 \leq i \leq 40 \\ v_h &= (r_1, r_2, ..., r_i) \end{aligned}$$

     so that the values of r_i are always between 0 and 1. The dimensionality of v_h is 40. In order to build our 2-D HMM, another pixel density

feature v_v is also extracted from a sliding window that moves from top to bottom. Figure 9 shows the feature vectors extracted from the sliding window in both horizontal and vertical directions for pixel density feature. This image has resulted in 35 horizontal and 41 vertical sliding windows

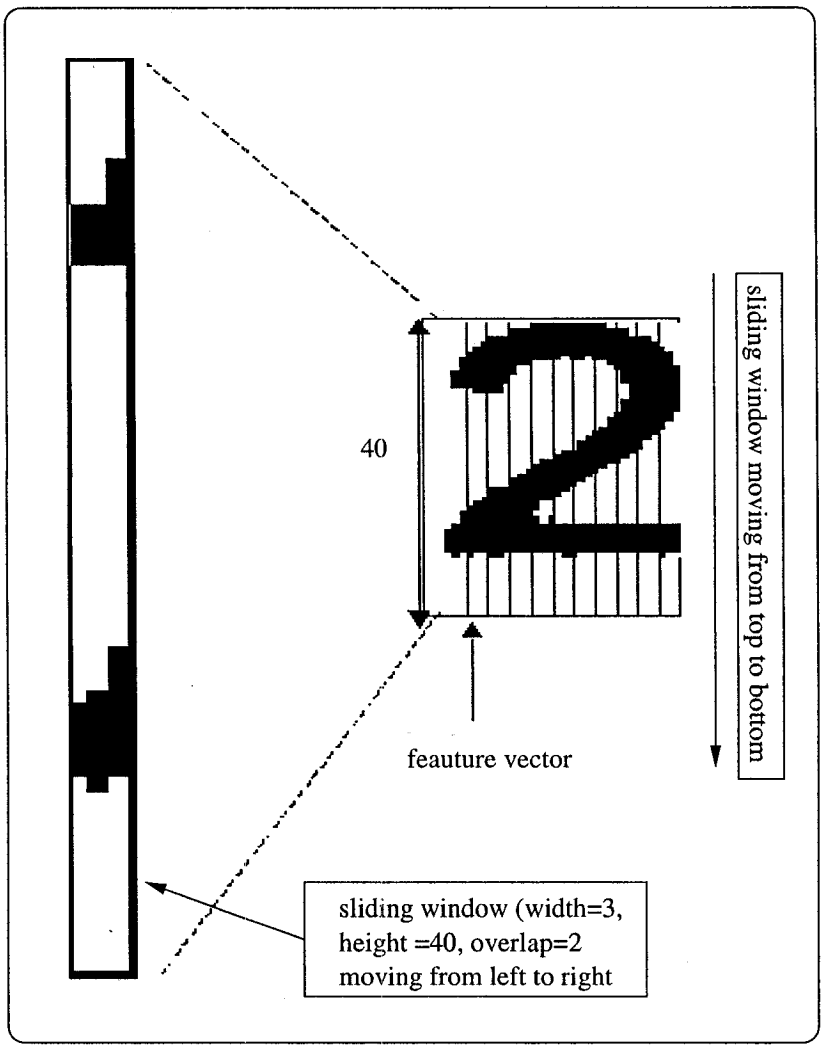


Figure 9: Pixel density feature extraction using a sliding window in 2 Directions

and therefore produced 35 horizontal and 41 vertical feature vectors with a dimensionality 40 of each vector.

After extracting pixel density features from all training sets in both horizontal and vertical directions, two types of training vector sets are generated: one is horizontal training vector set, another is vertical training vector set. In the following sections we will see how these training vector sets are used to create

code books.

2. Contour distance feature

   Similar to pixel density feature extraction, for the given binarized character image, the following operations are executed to extract contour distance features.

   - Normalization of character image (same as pixel density feature)
   - Contour extraction

     A contour point is a nonzero pixel which has at least one zero pixel in its four 4-connected pixels(up, down, left and right) [53]. Otherwise, it is a non-contour point. After getting all contour points and deleting all non-contour points, a contour of the image is extracted.
   - The same sliding window(height=40, width=3 and overlap=2) is also used to move from left to right and top to bottom, respectively. Each sliding window is divided into 5 zones, each of which has a height of 8 pixels and width of 3 pixels (=width of sliding window). From each sliding window a feature vector v is extracted using the following criteria:
     - In the vertical direction, for every 2 pixels of each zone i $(1 \leq i \leq 5)$, if there exists a left-most contour point x_{vl}, one distance $d_{vl}(ij)$ from this left-most contour point to the left border of image is produced (totally, 4=(8/2) left distances are produced in each zone, $1 \leq j \leq 4$).Otherwise, $d_{vl}(ij)$ is assigned to 0. Similarly, if there exists a right-most contour point x_{vr}, another distance $d_{vr}(ij)$ from this right-most contour point to the right border of image is produced (also, 4 right distances are produced in each zone). Totally, 8 distances are produced in the vertical direction of each zone and each distance is normalized to $0-1$ by the width of image.
     - In the horizontal direction, unlike the vertical one, we check every pixel of each zone if there exists a up-most contour point x_{hu} and calculate distance $d_{hu}(ik)$ from this up-most contour point to the up border of image. Otherwise, we also assign a value of 0 to $d_{hu}(ik)$ $(1 \leq k \leq 3)$. Then we do the same operation for the down-most contour points and get another 3 down distances ($d_{hd}(ik)$. Also, 6 distances are produced

in the horizontal direction and each distance is normalized to $0-1$ by the height of the image (40 in our 2-D HMMs).

- In each zone, 14 distances are produced and totally $70(5*14)$ distances will be produced in each sliding window moving from left to right. therefore, a feature vector v with dimensionality of 70 is extracted from each sliding window. After moving the sliding window from top to bottom, another feature vector will also be extracted in each sliding window.

Figure 10 shows the contour distance feature vectors extracted from the sliding window in both horizontal and vertical directions. This image has

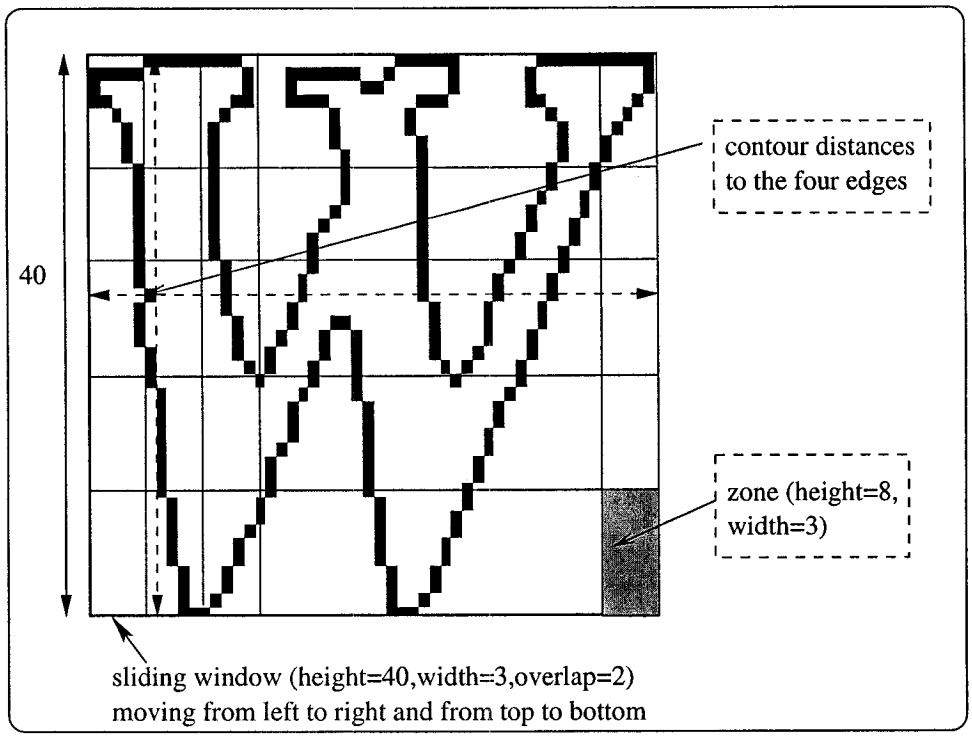


Figure 10: Contour distance feature extraction using sliding window in 2 Directions

resulted in 42 horizontal and 36 vertical sliding windows and therefore produced 42 horizontal and 36 vertical feature vectors with the dimensionality of 70 per vector.

After extracting contour distance feature from all training sets in both horizontal and vertical directions, like pixel density feature extraction, two types of training vector sets are also generated: one is horizontal training vector set,

another is vertical training vector set. It should be mentioned that the contour distance feature used in our 2-D HMMs is slightly better than the distance feature based on the original image because distance based on contour can reflect the inner contour information of the image.

5.2 Vector Quantization

The vector quantization (VQ) process generates output symbols by matching the input vector against each prototype vector, or codeword in the codebook using some distortion measure and assigning the index of the codeword having the smallest distortion. That is for the input vector $v = (v_1, v_2, ..., v_k)$ (k is dimensionality of v) and codeword $y_m = (y_{m1}, y_{m2}, ..., y_{mk}), 1 \leq m \leq M$ (M is codebook size), then the index, m^*, of the best codebook entry is

$$\begin{aligned} m^* &= \arg \min_{1 \leq m \leq M} d(v, y_m) \\ d(v, y_m) &= \sum_{i=1}^{k} (v_i - y_{mi})^2 \end{aligned}$$

Many kinds of codebook design algorithm have been proposed. Especially the conventional Linde-Buzo-Gray (LBG) algorithm [54] or modified K-means (MKM) algorithm [55] have been used to design codebooks in many HMM-based recognition systems. The VQ codebook design procedure of the conventional LBG or MKM algorithm is as follows:

1. Let all training vectors be the initial cluster and the centroid of the initial cluster be the initial codeword.

2. Increase the number of codewords by splitting the codewords of given codebook (LBG algorithm) or by replacing a codeword representing a cluster having the largest intra-cluster distance with the most distant vector pair in that cluster (MKM algorithm).

3. Perform clustering operation for all training vectors with a new codebook until the variation of the new codebook converges.

4. Stop if the desired codewords have been obtained or repeat (2)-(4).

Both LBG algorithm and MKM algorithm have the advantages of being simple in concept and implementation with low computational costs. However, the codebook size M of using LBG algorithm must be a power of 2 (i.e., 2, 4, 8, 16...). Therefore, in order to approximate the performance of system by adjusting codebook size, VQ based on MKM algorithm has been used in our system. The VQ training set (training vectors) is generated based on sliding window technique as mentioned in the feature extraction stage (section 5.1) for the features of all classes. Four codebooks have been generated for each 2-D HMM classifier corresponding to two horizontal features (pixel density and contour distance feature) and two vertical features (pixel density feature and contour distance feature. Figure 11 shows a block diagram of the VQ training and classification structure and is used for each 2-D HMM classifier which will be presented in section 5.3.

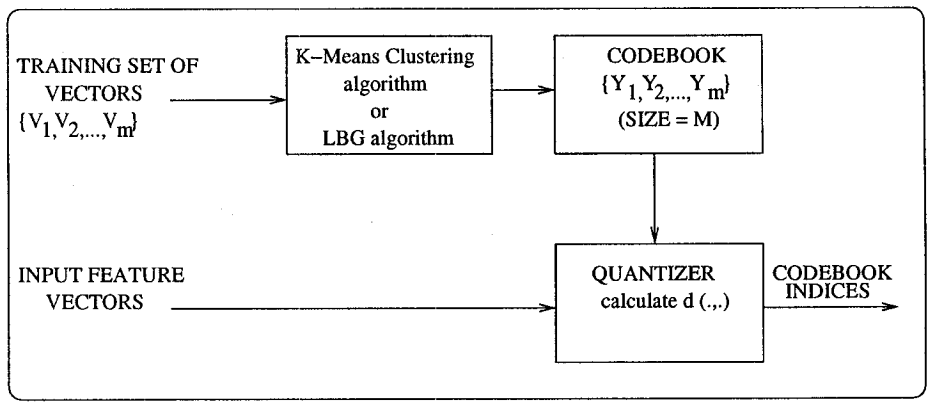


Figure 11: Block diagram of the VQ training and classification structure

To illustrate the effect of codebook size (i.e., number of codebook vectors) on the performance of system, Figure 12 shows experimentally the effect of codebook size ranging from 4 to 1024 on the recognition rate of isolated characters. It can be seen that a significant decrease in recognition rate starts from a codebook size of 64 to about 128 because a large codebook size means that each cluster will have worse training data and there is not enough good training data for HMM training. Also, a significant increase in recognition rate occurs around a codebook size of 32. This figure can also explain the correspondence between average training set distortion and codebook size. When the codebook size is much smaller, the distortion after VQ is much bigger. On the other hand, when the codebook size becomes extremely large, even reductions in distortion are much smaller, the recognition rate is still extremely

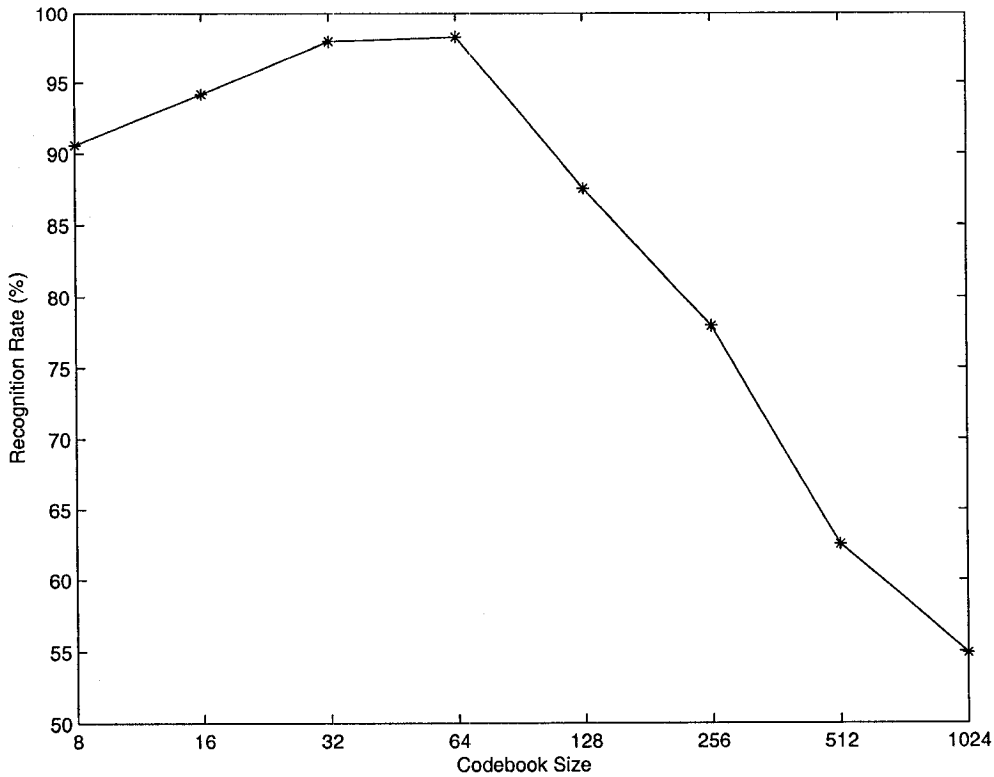


Figure 12: Recognition rate versus codebook size for isolated character recognition (# of states = 22)

low because the VQ procedure fails and the VQ procedure becomes a table look-up.

In our 2-D character based HMMs, the best codebook sizes of each 2-D character HMM classifier are generated based on testing performance.

5.3 2-D character based HMMs

5.3.1 Topology of character HMM

HMM is presented by two interrelated mechanisms. An underlying Markov chain having a finite number of states. and a set of random functions. One of which is associated with each state. An HMM is formally defined by the following elements [17, 16]:

- N: the number of states.
- M: the number of symbols.

- T: the number of observations.
- A set $Q = \{q_t\}$ of (hidden) states. $q_t \in \{1, 2, ..., N\}$.
  $t = 1, 2, ..., T$.
- A state transition probability distribution, also called transition matrix $A = \{a_{ij}\}$, representing the probability which goes from state S_i to state S_j.

$$a_{ij} = P[q_{t+1} = S_j | q_t = S_i] \quad 1 \leq i, j \leq N$$

  with $a_{ij} \geq 0$ and $\sum_{j=1}^{N} a_{ij} = 1$.
- A set $V = \{v_1, v_2, ..., v_M\}$ of observation symbols. M: the number of symbols, also called code book size in discrete HMMs.
- An observation symbol probability distribution, also called emission matrix $B = \{b_j(k)\}$, including the probability of emission of symbol v_k when the system state is S_j. For $1 \leq j \leq N, 1 \leq k \leq M$.

$$b_j(k) = P[v_k \ at \ time \ t | q_t = S_j]$$

  with $b_j(k) \geq 0$ and $\sum_{j=1}^{M} b_j(k) = 1$.
- An initial state probability distribution $\pi = \{\pi_i\}$, representing probabilities of initial states:

$$\pi = P[q_1 = S_i] \quad 1 \leq i \leq N$$

  with $\pi_i \geq 0$ and $\sum_{i=1}^{N} \pi_i = 1$.

For convenience, we denote an HMM as a triplet $\lambda = (A, B, \pi)$, which determines uniquely the model. HMM can model the set of observations using these probabilistic parameters as a probabilistic function of an underlying Markov chain whose state transitions are not directly observable. The structure of HMM has different forms: an constrained model (ergodic model) in which a transition from any state to any other state can be made, and a left-to-right model in which the time increases as the state index increases, that is, the states proceed from left to right. The latter model has the following properties, so it is appropriate for modelling sequential data as time goes on [16].

1. No transitions are allowed to state whose index are lower than the current state:

$$a_{ij} = 0. \quad j < i.$$

2. The state sequence must begin in state 1 and end in state N:

$$\pi_i = \begin{cases} 0 & \text{if} \quad i \neq 1; \\ 1 & \text{if} \quad i = 1; \end{cases}$$

3. Large changes in state index do not occur:

$$a_{ij} = 0. \quad j > i + \Delta.$$

especially when $\Delta = 1$, this model becomes strictly a left-to-right model in which only self transition and next transition are allowed and it is the simplest model and yet very efficient model. Our character based HMMs are based on the strictly left-to-right model (see Figure 13 (c)).

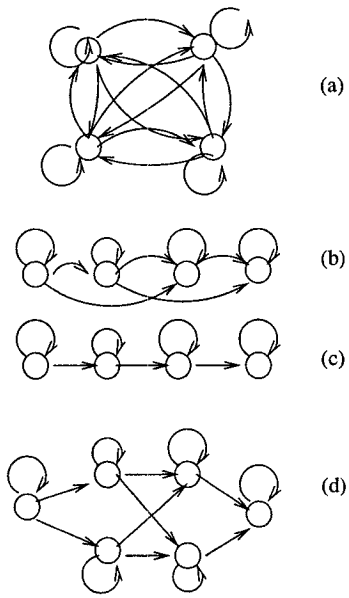


Figure 13: Illustration of four types of HMMs. (a) a 4-state ergodic model, (b) a 4-state left-right model, (c) a 4-state strict left-right model, and (d) a 6-state parallel path left-right model.

There are three main problems involved the use of HMM [17]:

1. Given the observation sequence $O = (o_1, o_2, ..., o_T)$, and a model HMM $\lambda = (A, B, \pi)$ (with $O_t \in V$ is generated by the model λ), we want to compute $P(O|\lambda)$, i.e. the probability of the observation sequence. Usually this is solved using the so called $forward - backward$ procedure [56] which will be presented in greater details in section 5.5.1.

2. Given the model $\lambda = (A, B, \pi)$, we want to determine the state sequence $I = \{i_1, i_2, ..., i_T\}$ $(1 \leq i \leq N)$ such that $P(O, I|\lambda)$ is maximum with respect to I. In other words, we want to compute the state sequence that most probably generates the observation sequence $O_1, O_2, ..., O_T$. This problem is resolved by the *Viterbi Algorithm* [17] which will be presented in section 5.5.3.

3. Given a set of L observation strings $\{O_t\}_l, 1 \leq t \leq T, 1 \leq l \leq L$, we want to determine $\lambda = (A, B, \pi)$ such that $P(\{O_t\}_l|\lambda)$ is maximized: this is the problem of training a HMM. The best-known method to perform this operation is the so called $Baum - Welch$ re-estimation technique [56, 17]. We will discuss this algorithm in section 5.5.2.

5.3.2 2-D character HMM

Our 2-*Direction* character HMM is different from the 2-Dimension HMM which is quite complicated and needs more computation as mentioned in [57]. Our 2-D HMMs have two different types but with similar performance as discussed below.

1. Each 2-D HMM is built based on concatenating two complementary observation sequences. First, we extracted one feature horizontally by using a sliding window to create one horizontal observation sequence O_h decoded from one code book CDB_h(code book size= M_h) created from all training sets for this horizontal feature. In the mean time, we extracted the same feature but vertically to create another vertical observation sequence O_v decoded from another code book CDB_v(codebook size= M_v) also created from all training set for the vertical feature. Then we concatenate observation sequence O_h and O_v to get another observation sequence O_{hv}. It should be pointed that when we concatenate O_h and O_v, we have added one constant ζ to all symbol values of CDB_v in order to differentiate O_h from O_v because some symbols of O_h could be the

same as symbols of O_v. Based on O_{hv}, we built one of our 2-D HMMs (see Figure 14).

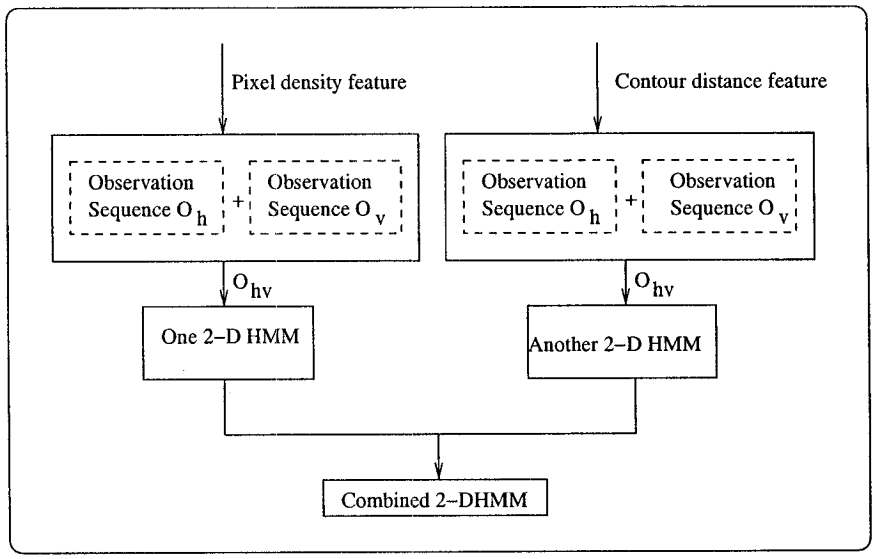


Figure 14: Building 2-D HMM by concatenating horizontal and vertical observation sequence O_h and O_v

The feature used in one 2-D HMM is the pixel density feature described in section 5.1. Another feature used in another 2-D HMM is the contour distance feature also mentioned in section 5.1. After building this two 2-D HMMs, we combine them together using sum rule.

2. Each 2-D HMM is the combination of two complementary 1-D HMMs using the same feature (pixel density or distance feature) mentioned above. After extracting the features horizontally and vertically, we also created two observation sequences O_h and O_v. Unlike method 1, we built two complementary $1-D$ HMMs instead of concatenating O_h and O_v. After combining these two 1-D HMMs together using sum rule, we build a 2-D HMM and the experimental results (section 7.1) have shown significant improvement of isolated character recognition comparing to single 1-D HMM. Another 2-D HMM consists of two 1-D HMMs which extracted contour distance feature horizontally and vertically and is also built similar to the first 2-D HMM. For the two 2-D HMMs, we also combine them using sum rule in the classification stage.

For each 2-D HMM, whether it is built from method 1 or method 2, the experimental results (section 7.1) have shown significant improvement of isolated character recognition comparing to single 1-D HMM. Another interesting aspect is that methods 1 and 2 have almost the same performance as indeicated in our experimental results.

In our OCR system, we have five 2-D HMM classifiers based on baseline information and frequency statistics of touching characters. Currently, all the five 2-D HMMs are built using method 2 and trained by using Baum-Welch algorithm which will be presented in section 5.5.2, The four 2-D HMM classifiers related to baseline are described below.

- Ascender 2-D HMM classifier (2-D ASCHMM) which is built for 57 ascender classes (10 numerals '012...9', 26 capital characters '$AB...Z$', 8 lower characters '*bdfhiklt*' and 13 punctuations described in section 4.2). In order to classify different font styles of character, 4 models corresponding to normal, bold, italic and bold italic font styles have been built for each class. This leads to a total of 218 models ($4 * 57$) for 2-D ASCHMM classifier.

- Descender 2-D HMM classifier (2-D DESHMM) which is designed for 5 descender characters ('g j p q y'). There are 20 models ($4 * 5$) in Descender 2-D HMM classifier.

- Center 2-D HMM classifier (2-D CENHMM) which is built for 13 center characters ('a c e m n o r s u v w x z'). Also, $52(4 * 13)$ models have been built in Center 2-D HMM classifier.

- General 2-D HMM classifier (2-D GENHMM) which is designed for all 75 classes (10 numerals, 52 letters, 13 punctuations). In case of isolated character training and recognition, we can not distinguish between each pair of 8 lower-case and upper-case character pairs $\{(Cc), (Oo), (Pp), (Ss), (Vv), (Ww), (Xx), (Zz)\}$ without baseline and contextual information. Therefore, we have grouped each of these 8 pairs of characters into one class. This reduced the total number of classes from 75 to 67 $(75 - 8)$, and totally 268 $(4 * 67)$ models were built in the General 2-D HMM classifier.

5.4 Touching Character Based HMM (TCHMM)

Touching character based HMM (TCHMM) is designed to recognize the most frequent touching characters without segmentation based on our statistic work for touching characters described in section 4.3 and will be presented in the next section. Also, a new method has been introduced to build some special touching character HMMs and 200 most frequent word HMMs (WHMM) which will be discussed in section 5.4.2

5.4.1 2-D touching character HMMs(2-D TCHMM)

Recognition of touching characters is difficult for any segmentation based OCR system. In order to recognize touching characters without segmentation, an intensive study has been conducted to determine the most common 2, 3, 4 and 5 touching characters that may occur in any general document as mentioned in section 4.3. In our current OCR system, the 57 most frequent occurring touching characters were selected for modelling by our 2-D HMM.

The following touching characters were found to have the highest frequencies, (an, th, ar, al, as, be, rm, in, fi, ed, es, pe, ec, am, Th, re, ri, te, ti, od, po, ce, ta, se, ty, co, ru, rs, rn, rv, ry, tw, ff, ca, ro, ct, ra, di, gn, wi, oc, ai, ee, hi, ts, to, ak, gy, ur, ul, tri, art, all, tru, are, the, ari) (section 4.3). Our 2-D TCHMM is similar to our 2-D character HMM. We see each touching characters as a single character and use the same features (pixel density and contour distance feature) and the same method (2-Directions) to build the 2-D TCHMM. The 2-D TCHMM classifier is formed by 57 models (1 model for each touching character) while 2-D character HMMs have 4 models per character. The experiments have shown a higher performance ($> 99.3\%$) for recognition of the 57 most frequent touching characters.

5.4.2 Automatically building TCHMM and WHMM

Section 5.4.1 has presented a 2-D TCHMM classifier which showed good performance. However, the 2-D TCHMM classifier is only designed for the 57 most frequent touching characters. We can not include all other touching characters because of the lack of training samples. Fortunately, the HMM has the advantage of building touching characters or word HMM by concatenating isolated character HMMs without touching character or word database. Similar to the idea of building word HMM, theoretically,

we can build any touching character HMM by concatenating the HMMs of isolated character. Figure 15 shows the simple procedure of this method.

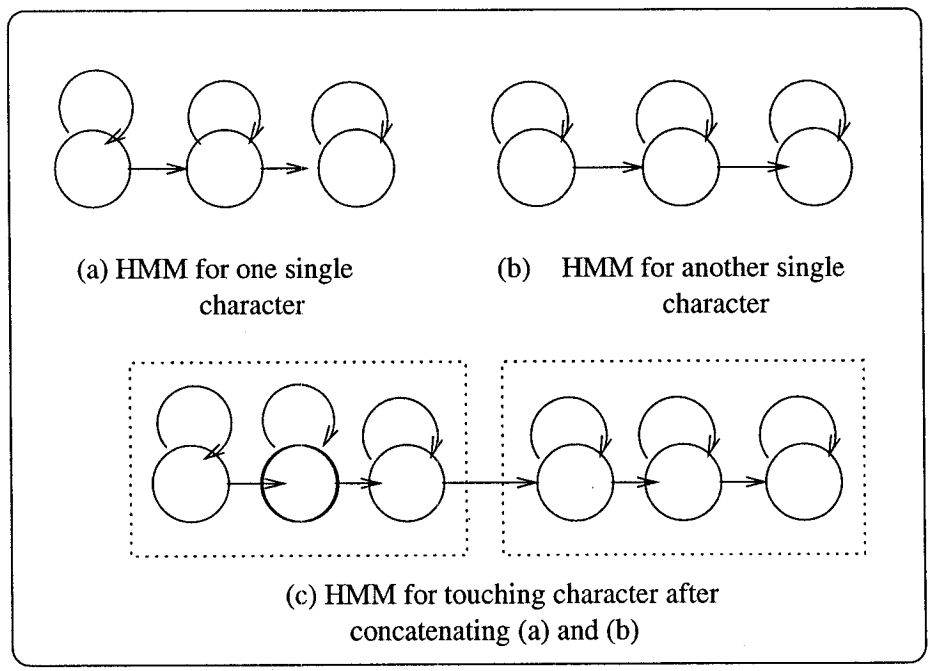


Figure 15: Procedure of building touching character HMM by concatenating two left-to-right single character HMMs

However, the performance of this method is lower than our current 2-D TCHMM classifier based on the experiments because real touching character database has been used in the current 2-D TCHMM classifier, and simply concatenating isolated characters can not reflect the real touching situation between adjacent isolated characters and it suffers from the following three problems:

- The first problem of this method is how to accurately estimate the transition probability from the last state of the first HMM model to the first state of the second HMM model.

- The second problem is that the features extracted from the real touching characters are different from these trained on the isolated character classifiers for some touching characters. There exists some white space for real touching characters while each isolated character does not (see Figure 16).

- The last problem is that the 2-D HMM can not be applied to this method because the horizontal features extracted from real touching characters are quite

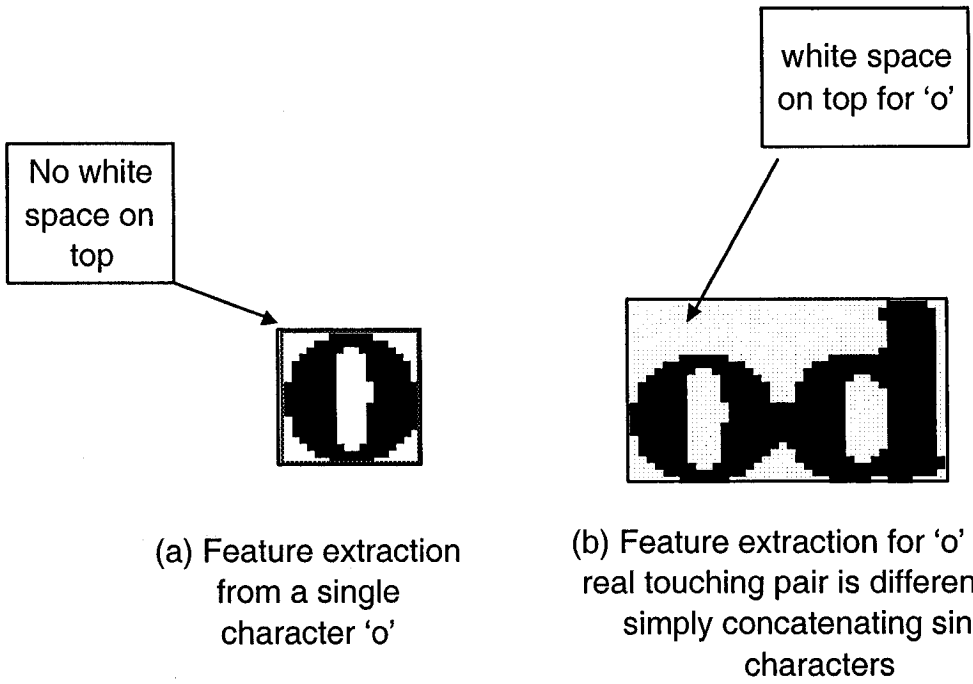


Figure 16: Feature extraction difference in single character and touching characters

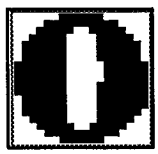
different from the horizontal features extracted from isolated characters. This will absolutely decrease the performance of this method. Therefore, only 1-D HMM can be used in this method.

For the problems mentioned above, we have introduced two steps to solve that:

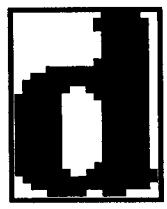
1. Concatenating single character images together. However, this is not a simple concatenation. First, we normalize each single character by fixing its height (40 in our system) while keeping the ratio of height and width. In the meantime, we consider the baseline information of each character (ascender, descender or center) and slightly resize their height (decrease or increase) according to each character's baseline to concatenate them together (see Figure 17).

2. Retraining touching character HMM classifier by using concatenated touching images while using the same code book of single character classifiers. Therefore, we can get relatively accurate observation sequences and transition probabilities.

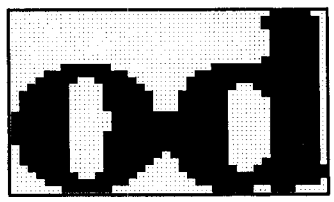
The experiment made about 10 percent improvement comparing to simple method after introducing the above solutions and is closer to the performance of the original



(a) 'o' is a center character



(b) 'd' is a ascender character



(c) Concatenating 'o' and 'd' together after considering each character's baseline

Figure 17: Building touching character image by concatenating two single character image considering baseline information

1-D TCHMM classifier which is trained using real touching database when testing on the 57 most frequent touching characters (see table 5).

The advantage of the modified method is that it can be used to build any touching character HMM automatically, especially for some special touching characters ('fi', 'fl', 'ft', etc.). In addition, this idea can be applied to building word HMM, and we have applied the modified method to build word HMM for the 200 most frequent words [58] which has occupied around 80 percent of all English words. The touching character HMM and word HMM will give us the potential capability to recognize

Table 5: A comparison of performance of four types of touching character HMMs tested on the same set

Type of Touching Character HMM	*no.* of models	Recognition rate %
Simple concatenating 1-D TCHMM	57	86.53
Original 1-D TCHMM	57	96.37
Suggested concatenating 1-D TCHMM	57	96.12
Original 2-D TCHMM	57	99.37

some special touching characters and words without segmentation.

5.5 Training and Testing

5.5.1 Forward-Backward algorithm

The forward-backward procedure is used to solve problem 1 and also help to solve problems 2 and 3 mentioned in 5.3, and also is used in the parameter estimation of an HMM. This method makes use of two inductively computed variables $\alpha_t(i)$ and $\beta_t(i)$, called respectively *forward* and *backward* variables, defined as [17]:

- $\alpha_t(i) = P(o_1 o_2 ... o_t, q_t = i|\lambda)$, that is, the probability of the partial observation sequence, $o_1 o_2 ... o_t$, (until time t) and state i at time t, given the model λ. we can solve for $\alpha_t(i)$ inductively, as follows:

  1. Initialization
  $$\alpha_1(i) = \pi_i b_i(o_1), \quad 1 \leq i \leq N.$$

  2. Induction
  $$\alpha_{t+1}(j) = \left[\sum_{i=1}^{N} \alpha_t(i) a_{ij}\right] b_j(o_{t+1}), \quad \begin{matrix} 1 \leq t \leq T-1 \\ 1 \leq j \leq N \end{matrix}$$

  3. Termination
  $$P(O|\lambda) = \sum_{i=1}^{N} \alpha_T(i).$$

  Step 1 initializes the forward probabilities as the joint probability of state i and initial observation o_1. Considering our strict left-to-right model, the state sequence must begin in state 1. The induction step, which is the heart of the forward calculation, iteratively calculate $\alpha_{t+1}(j)$ which is the probability of the observation sequence $o_1 o_2 ... o_{t+1}$ (until time $T-1$) and state j at time $t+1$. Again, our strict left-to-right model must be considered in each iterative step, to simplify the calculation. Finally, step 3 gives the desired calculation of $P(O|\lambda)$ as the sum of the terminal forward variables $\alpha_T(i)$. This is the case since, by definition,
  $$\alpha_T(i) = P(o_1 o_2 ... o_T, q_T = i|\lambda)$$

and hence $P(O|\lambda)$ is just the sum of the $\alpha_T(i)'s$. In our system, we have modified the calculation of $P(O|\lambda)$ as follows:

$$P(O|\lambda) = \alpha_T(N).$$

Since the state in the strict left-to-right model must end in N, the computation involved in the calculation of $\alpha_t(j), 1 \leq t \leq T, 1 \leq j \leq N$ requires an order of N^2T calculations.

- $\beta_t(i) = P(o_{t+1}o_{t+2}...o_T|q_t = i, \lambda)$ that is, the probability of the partial observation sequence from $t+1$ to the end, given state i at time t and the model λ. In a similar manner, we can solve for $\beta_t(i)$ inductively, as follows:

  1. Initialization

  $$\beta_T(i) = 1, \quad 1 \leq i \leq N.$$

  2. Induction

  $$\beta_t(i) = \sum_{j=1}^{N} a_{ij}b_j(o_{t+1})\beta_{t+1}(j), t = T-1, T-2, ..., 1, \quad 1 \leq i \leq N.$$

  The initialization step 1 arbitrarily defines $\beta_T(i)$ to be 1 for all i. Step 2, shows that in order to have been in state i at time t, and to account for the observation sequence from time $t+1$ on, we have to consider all possible states of j at time $t+1$, accounting for the transition from i to j, as well as the observation o_{t+1} in state j, then account for the remaining partial observation sequence from state j. Also, in the strict left-to-right model, the calculation is much easier than other models mentioned above.

  Again, the computation of $\beta_t(i), 1 \leq t \leq T, 1 \leq i \leq N$, requires an order of N^2T calculation, same as $\alpha_t(i)$.

5.5.2 Parameter Estimation using Baum-Welch algorithm

The most difficult problem of HMMs is to determine a method to adjust the model parameters (A, B, π) to satisfy a certain optimization criterion. This is the problem 3 associated to HMMs mentioned above. There is no known way to analytically solve for

the model parameter set that maximizes the probability of the observation sequence in a closed form. We can, however, choose $\lambda = (A, B, \pi)$ such that its likelihood, $P(O|\lambda)$, is locally maximized by using *Baum-Welch* method (also known as the EM (Expectation-Maximization) method [59]). In this section we simply discuss one iterative procedure, based primarily on the classic work of Baum and his colleagues, for choosing the maximum likelihood(ML) model parameters.

We first define variable $\xi_t(i, j)$, which is the probability of being in state i at time t, and state j at time $t + 1$, given the model and the observation sequence

$$\xi_t(i, j) = P(q_t = i, q_{t+1} = j|O, \lambda).$$

From the definitions of the forward and backward variables, we can write $\xi_t(i, j)$ in the form:

$$\begin{aligned} \xi_t(i,j) &= \frac{P(q_t=i, q_{t+1}=j, O|\lambda)}{P(O|\lambda)} \\ &= \frac{\alpha_t(i) a_{ij} b_j(o_{t+1}) \beta_{t+1}(j)}{P(O|\lambda)} \\ &= \frac{\alpha_t(i) a_{ij} b_j(o_{t+1}) \beta_{t+1}(j)}{\sum_{i=1}^{N} \sum_{j=1}^{N} \alpha_t(i) a_{ij} b_j(o_{t+1}) \beta_{t+1}(j)} \end{aligned}$$

Then we define another variable $\gamma_t(i)$ as the probability of being in state i at time t, given the entire observation sequence and the model; hence, we can relate $\gamma_t(i)$ to $\xi_t(i, j)$ by summing over j, giving

$$\gamma_t(i) = \sum_{j=1}^{N} \xi_t(i, j)$$

Using the above formula, if we define the current model as $\lambda = (A, B, \pi)$, we can re-estimate the parameters of an HMM using the re-estimated models as $\overline{\lambda} = (\overline{A}, \overline{B}, \overline{\pi})$ iteratively. The re-estimation formula is

$$\overline{\pi_j} = \gamma_1(i)$$

$$\overline{a_{ij}} = \frac{\sum_{t=1}^{T-1} \xi_t(i,j)}{\sum_{t=1}^{T-1} \gamma_t(i)}$$

$$\overline{b_j(k)} = \frac{\sum_{t=1, o_t=v_k}^{T} \gamma_t(j)}{\sum_{t=1}^{T} \gamma_t(j)}$$

Based on above procedure, it can be shown that either:

1. The initial model λ defines a critical point of the likelihood function, where new estimates equal the old ones, or

2. Model $\overline{\lambda}$ is more likely in the sense that $P(O|\overline{\lambda}) \geq P(O|\lambda)$, i.e. new model estimates are more likely to produce the given observation sequence.

Thus if we iteratively use $\overline{\lambda}$ in place of λ and repeat the re-estimation calculation, we then can improve the probability of O being observed from the model until some limiting point is reached. The final result of this re-estimation procedure is an ML estimate of the HMM. It should be pointed out that the *Baum-Welch* algorithm leads to a local maximum only. In addition, three main problems are related to the performance for re-estimation:

- Initial estimates of λ

  One interesting factor for the re-estimation of an HMM parameters is the choice of initial estimates for (A, B, π) so that the local maximum is equal to or as close as possible to global maximum of the likelihood function.

  Basically, there is no simple or straightforward answer. Instead, experience has shown that either random or uniform initial estimates of the π and A parameters are adequate for re-estimation of these parameters for almost all cases. Especially for our strict left-to-right model. However, for the B parameters, experience has shown that good initial estimates are helpful in the discrete symbol case. therefore, we define the initial estimates in our strict left-to-right model as follows:

A = non-zero random values, normalized to satisfy the constraint:
$\sum_{j=1}^{N} a_{ij} = 1,\ i = 1, 2, \ldots, N.$
$b_j(k) = \frac{1}{M} + \epsilon$ to satisfy: $\sum_{k=1}^{M} b_j(k) = 1,\ k = 1, 2, \ldots, M.$
ϵ is a uniformly distributed random variable which is much smaller than $\frac{1}{M}$

- Scaling

  Scaling is required for implementing the re-estimation procedure of HMMs when t starts to get big since each term of $\alpha_t(i)$ starts to head exponentially to zero and the underflow problem occurs when the dynamic range of the $\alpha_t(i)$ computation exceeds the precision range of any machine for sufficiently large t. Hence, the only reasonable way to perform the computation is to incorporate a scaling procedure.

  In our character based HMMs, we used the basic scaling procedure which is to multiply $\alpha_t(i)$ and $\beta_t(i)$ by the same coefficients c_t at time instant t while keeping the re-estimation result unchanged. In our HMM, c_t is defined as:

$$\begin{aligned} c_1 &= \frac{1}{\sum_{i=1}^{N} \alpha_1(i)} \\ c_t &= \frac{1}{\sum_{i=1}^{N} \hat{\alpha_t(i)}}, \quad 2 \leq t \leq T \end{aligned}$$

  giving $\hat{\alpha_1(i)} = c_1 \alpha_1(i)$, then replace each $\alpha_t(i)$ with a new one.
  Obviously, the scaling procedure need not be applied at every time instant t, but can be performed whenever desired. If scaling is not performed at some instant t, the scaling coefficients c_t are set to 1 at that time. The only real change to the HMM because of scaling is the final computation of $P(O|\lambda)$. $P(O|\lambda)$ should be divided by the multiplication of coefficients c_t in the final computation or by using *log* sum of c_t for $log\,[p(O|\lambda)]$.

- Effects of insufficient training data

  Another problem associated with training HMM parameters via re-estimation methods is that the observation sequence used for training is, of necessity, finite. Thus there is always an inadequate number of occurrences of low-probability events (e.g. $b_j(k)$) to give good estimates of the model parameters. If $b_j(k)$ (state=j and $o_t = v_k$) equal 0, it will stay 0 after re-estimation. The resultant model would produce a zero probability result for any observation sequence that

actually includes $b_j(k)$. Such a singular outcome is obviously a consequence of the unreliable estimate that $b_j(k) = 0$ due to insufficiency of training set.

In order to solve this possible problem in our system (even our system might have sufficient training set), we used a simple way to handle this problem. That is to add an extra threshold constraint (η) to the model parameters to ensure that no model parameter estimate falls below a specified level [60] as follows:

$$b_j(k) = \begin{cases} b_j(k), & if\ b_j(k) \geq \eta \\ \eta, & otherwise \end{cases}$$

5.5.3 Viterbi algorithm

The Viterbi algorithm is a formal technique for finding the single best state sequence, $q = (q_1 q_2 \ldots q_T)$ with the highest probability, for the given observation sequence $O = (o_1 o_2 \ldots o_T)$. To do this, we need to define the quantity

$$\delta_t(i) = \max_{q_1 q_2 \ldots q_{t-1}} \{P(q_1 q_2 \ldots q_{t-1}, q_t = i, o_1 o_2 \ldots o_t | \lambda)\}$$

that is, $\delta_t(i)$ is the best score (highest probability) along a single path, at time t, which accounts for the first observation and ends in state i. The complete procedure of this algorithm can be stated as follows:

1. Initialization. For all states i $(1 \leq i \leq N)$,

$$\begin{aligned} \delta_1(i) &= \pi_i b_i(o_1), \\ \phi_1(i) &= 0, \end{aligned}$$

2. Recursion. From time $t = 2$ to T, for all states j $(1 \leq j \leq N)$,

$$\begin{aligned} \delta_t(j) &= \max_i \{[\delta_{t-1}(i) a_{ij}]\}\, b_j(o_t), \\ \phi_t(j) &= \arg\max_i \{[\delta_{t-1}(i) a_{ij}]\}, \end{aligned}$$

3. Termination. (* indicates the optimized results).

$$\begin{aligned} P^* &= \max_i \{\delta_T(i)\} \\ s_T^* &= \arg\max_i \{\delta_T(i)\} \end{aligned}$$

4. Path (state sequence) backtracking. From time $T-1$ to 1

$$q_t^* = \phi_{t+1}(q_{t+1}^*)$$

It should be noted that the Viterbi algorithm is similar (except for the backtracking step) in implementation to the forward-backward algorithm explained above. The major difference is the maximization of $P(O|\lambda)$ over previous states, which is used in place of the summation of $P(O|\lambda)$ over all states. On the other hand, the Viterbi algorithm is extremely efficient since it can operate in the logarithm domain using only additions. Also it is possible to obtain the the state sequence at the same time. Because of its advantages, we used it in our system for HMM based classification. Given that the classifier has K models, an observation sequence O is classified to class L using the following:

$$L = \arg\max_k \{P(O|\lambda_k))\}$$

The diagram in Figure 18 explains this method.

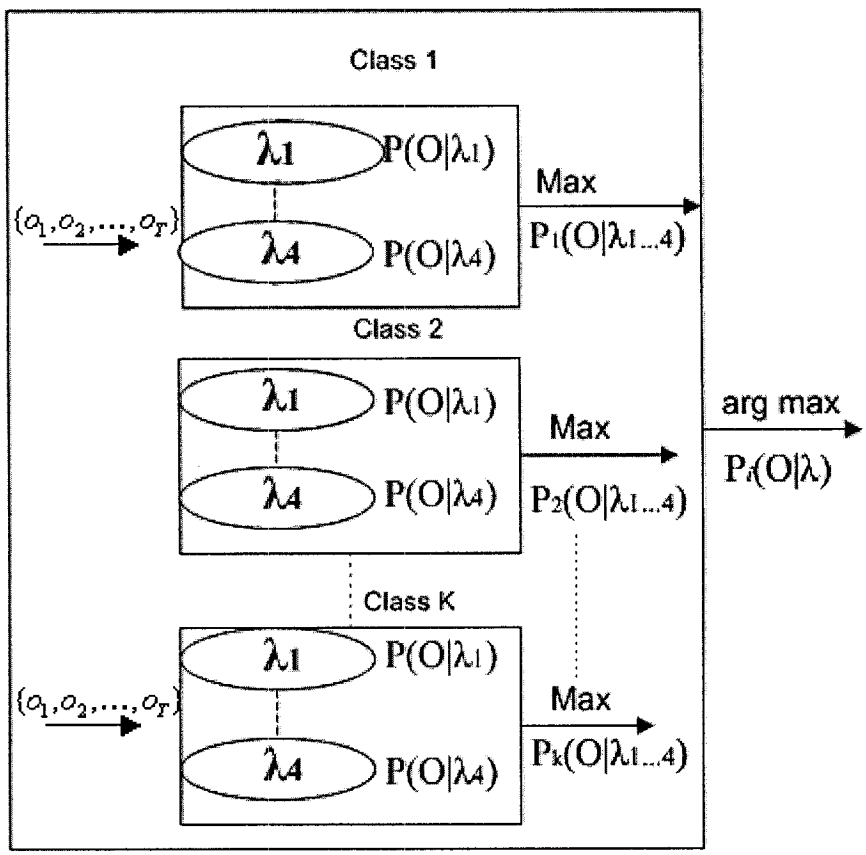


Figure 18: Diagram of Recognition Method

Chapter 6

Hybrid HMM and MLP Classifier for string recognition

6.1 System Architecture

The basic architecture of a hybrid 2-D HMM and MLP classifiers for the character string recognition has been designed as shown in Figure 19.

In the following sections, we will discuss the combination of an isolated character classifier and a touching character classifier.

6.2 Combination of 2-D HMM and MLP classifier

The combination of multiple classifiers is receiving increasing the attention of researchers and practitioners of pattern recognition. Classifier combination is expected to outperform the individual classifiers, whose performance is limited by the imperfection of feature extraction and learning/classification algorithms, and the inadequacy of training data. In our system, MLP classifier can achieve a higher performance for the recognition of isolated characters and it has been widely used in many OCR systems based on segmentation technique for many years [26, 27, 28, 61]. In addition, our 2-D HMM classifier can also achieve good performance for the recognition of isolated characters. However, in string recognition, only single classifier can not achieve a better performance comparing with combining multiple classifiers. Therefore, we have suggested the method of combining 2-D HMM and MLP classifiers to improve

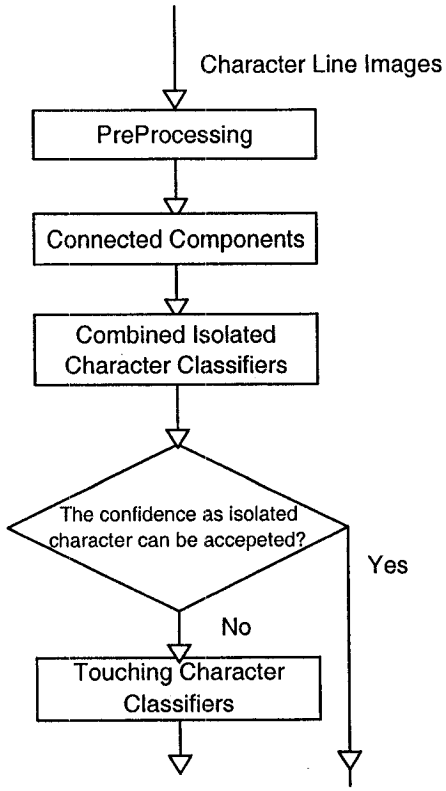


Figure 19: System architecture of Hybrid 2-D HMM and MLP character string recognition

the performance of our OCR system. In the following sections, we will first give a simple introduction to MLP classifier and then discuss the combination of 2-D HMM and MLP classifiers.

6.2.1 MLP classifier for the recognition of isolated characters

1. Feature Extraction

   The feature extraction in MLP classifier is different from 2-D HMM classifier. For a given character image, no normalization is used and three feature sets are extracted in the MLP classifier:

   - Feature Set 1 (160D) consists of a mesh feature (160D), where the number of local regions is 256 (16x16) and the dimension of the feature vector is reduced from 256D to 160D by using Principal Component Analysis (PCA) (see [62] for details).
   - Feature set 2 (164D) consists of gradient feature (96D) and distance feature

(68D). For the gradient feature, 4 partitions of 3600 gradient direction are considered, and the number of local regions is 24, that is 4 vertical divisions and 6 horizontal divisions. For the distance feature, 17 positions are considered for each of the sides (left, right, up, bottom) (see [62] for details).

- Feature set 3 consists of outline layer features. It calculates the distances from the minimum boundary rectangle of the character image to the first black pixels of outline layers on projection lines of 4 directions. These distance values are divided by the width or the height of the image for normalization purpose. The projection lines are equally spaced, and 17 lines are used in each direction. In addition, the maximum number of outline layers in each direction has been set at 5 empirically.

2. Implementation of MLPs

   In our system, two separate MLPs are implemented by using different input feature sets (feature sets 1 and 2), and the outputs of the MLPs are combined according to sum rule.

   The Neural Network (NN) structures for these two MLPs:

   - The first one is: 161 (input neurons) 90 (hidden neurons) 62 (output neurons)
   - The second one is: 165 (input neurons) 90 (hidden neurons) 62 (output neurons)

   Similar to our 2-D HMM classifiers, four isolated MLP classifiers are designed to recognize character candidates according to their base line information (ascender, center, descender and general) obtained in the preprocessing stage. A touching character MLP classifier is also designed to classify the 57 most frequent touching characters mentioned in section 4.3. Each of these five MLP classifiers is a combination of two separate MLP networks mentioned above and the experimental results have shown a higher performance which will be presented in section 7.1.

3. MLP classifiers trained with outliers

   Though the MLP classifiers have shown a higher performance for recognition

of isolated characters, we still found that our MLP classifiers sometimes are sensitive to some out-of-class patterns (outliers). In order to make the MLP classifier more accurate in character classification and improve its resistance to outliers (NN classifiers were shown to be weak in outlier rejection according to some publications), some modifications in the training stage of the NN classifier have been made.

We have trained the MLPs by using both isolated characters and outlier samples. Two types of outliers have been generated. The outliers with the first type are generated from a set of touching character pairs (in CENPARMI database). Here the touching character pairs contain all possible combinations of letters (small or capital letters). The 1/4 parts of the pairs on the right and left sides and 1/3 parts of the pairs in the middle are used as the candidates of the outliers. If the confidence value of a candidate (from a classifier trained without outliers) is higher than a threshold, this candidate is selected as an outlier for the training. For the outliers with the second type, we generate them from isolated characters. Currently 1/3, 1/2, 2/3 left parts of characters (6 fonts) are used as candidates to produce the outliers with the second type.

In the training stage of our system, when the input pattern is an outlier, the target outputs of all classes are set to 0. This effect guides the classifier to give low outputs for all target classes on outlier patterns. The experimental result has shown that training with outlier samples can largely improve the resistance of NN classifiers to outliers. This is very useful in integrating segmentation and recognition of character strings.

6.2.2 The Combination

As mentioned above, in order to obtain a better performance for machine-printed character string recognition, a combination system of 2-D HMM classifier and MLP classifier is being developed. Combining multiple classifiers has a different way and different combination produces a different performance. Therefore, the methodology of integrating the results of a number of different classification algorithms has been studied by many researchers [31] and it gives us a good guide to combine our 2-D HMM and MLP classifiers.

To achieve high combination performance, there are some requirements related to the constituent classifiers and the combination method. An essential condition is that the individual classifiers are complementary like HMM and MLP classifiers. On a set of complementary classifiers, also a good combination method is needed to achieve a higher performance. The variety of classifier combination methods can be divided into three groups according to the level of classifier outputs: abstract (crisp class) level, rank level, and measurement level [31]. Combination at the measurement level is advantageous in that the output measurements contain rich information of class measures. Currently, the combination in our system is implemented at the measurement level which can be divided into two subtasks: confidence evaluation and combination rule. Figure 20 shows the combination topology:

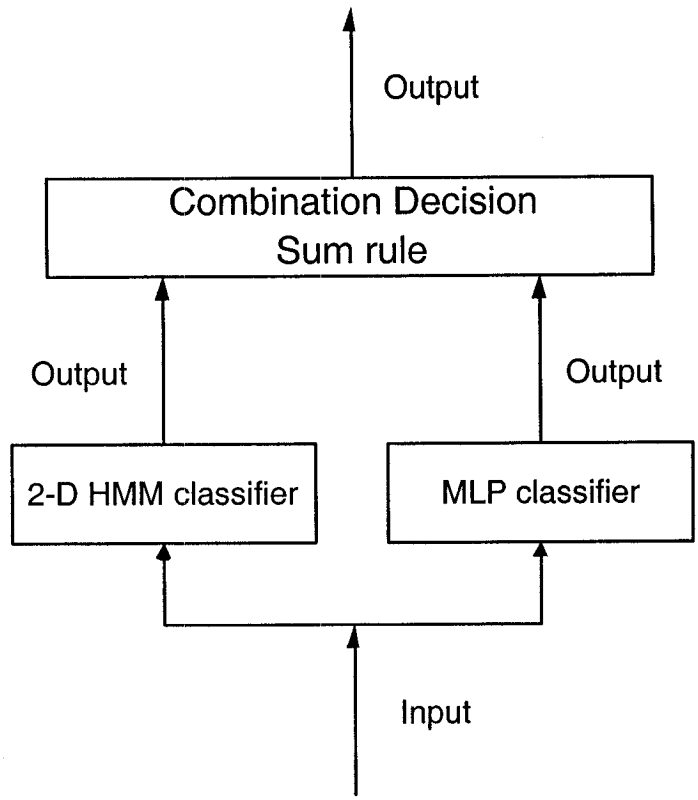


Figure 20: Diagram of Combination topology

- Confidence evaluation

  For combining the classifiers with diverse outputs, confidence evaluation is particularly important. A confidence evaluation method can be decomposed into two sub-functions: a scaling function and an activation function [63]. Basically,

there are three activation functions corresponding to three types of confidence: log-likelihood, exponential, and sigmoid. The scaling function shifts and re-scales the classifier outputs into moderate ranges.

We denote 2-D HMM and MLP classifiers as $E_k(k = 1, 2)$, each classifying an input pattern into the same set of M classes, with the outputs of classifier E_k denoted as d_k. For generating confidence measures, first the classifier outputs are shifted and scaled by a scaling function $f_i(d_k)$ $(1 \leq i \leq M)$ to a moderate range. The re-scaled outputs are transformed to confidence measures by an activation function

$$g_i(d_k) = g_i\left[f_i(d_k)\right]$$

As prevalently used in neural networks, the sigmoid function behaves well in squashing neuronal outputs to approximate probability measures and itself has been a good activation function for confidence transformation and the outputs from our MLP classifiers have been automatically transformed to $(0, 1)$. Therefore, we keep the outputs of the MLP classifier unchanged.

Considering the outputs of our 2-D HMM classifier, we used the negative log-likelihood as the confidence value which is different from the confidence value of MLP classifier and should be normalized to the same range $(0, 1)$. We defined the activation function simply the linear form of the scaling function:

$$g_i(d_k) = \alpha f_i(d_k) + \beta$$

where α and β are coefficients.
Therefore, we first need to define a scaling function $f_i(d_k)$ to re-scale the output values of HMM to zero mean and standard deviation 1 (*GlobalNormalization*):

$$f_i(d) = \frac{d_i - \mu_0}{\sigma_0}$$

where μ_0 and σ_0^2 are the mean and variance of the pooled classifier outputs, respectively. Then we normalize the confidence values to the range of (0,1) by $g_i(d_k)$. Finally, the outputs of both 2-D HMM and MLP classifiers have been transformed to confidence values in the range of $(0, 1)$ and combination rule will make the final decision of combination of 2-D HMM and MLP classifiers.

- Combination rule

  The combination rule used in the decision combination step (as shown in figure 20) can be Sum, Product, Maximum, Median etc. Some researchers have proven that on transforming the classifier outputs to confidence measures, high performance can be obtained by simple combination rules such as the sum rule or product-rule [35]. Currently, Sum rule has been used in our system and other rules will also be experimented in the future. The Sum rule can be described as follows:

  Let x be an input sample, and $C_k(x)$, where $k = 1, 2$ be the decision on x by 2-D HMM and MLP classifiers, respectively. $D_{k,w_j}(x)$, where $k = 1, 2$ and $j = 1, 2, ..., M$ represents the confidence value assigned to class w_j by 2-D HMM and MLP. Here M is the total number of classes in a classifier, e.g. M=13 for the center classifier. By using Sum rule, the final combined result $D(x)$ and confidence value $C_{w_j}(x)$ are decided by the following equation:

$$\begin{aligned} C_{w_j}(X) &= \frac{\sum_{k=1}^{2} C_{k,w_j}(x)}{2} & j = 1, 2, .., M \\ D(x) &= w_j & if\, C_{w_j} = \max_{i=1}^{M}(C_{w_i}(x)) \end{aligned}$$

After combining 2-D HMM and MLP classifiers, a group of isolated character classifiers are used based on different baseline information of connected components as shown in the following digram (Figure 21):

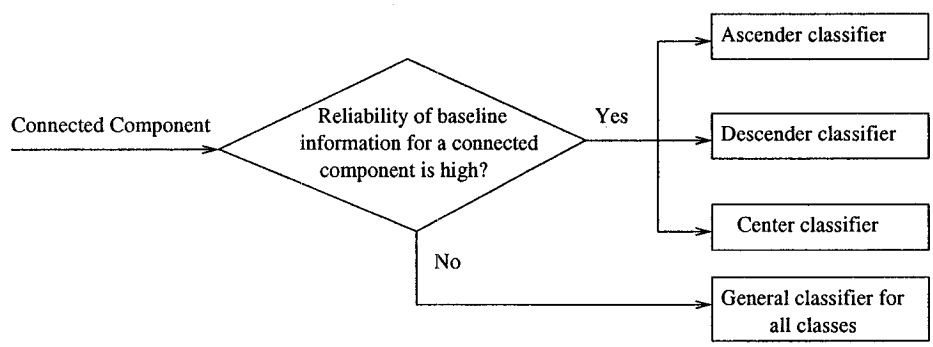


Figure 21: Diagram of isolated character recognition

From this structure we know that the reliability of baseline information is very important. Figure 22 shows an example of reference lines detected. The reliability of the baseline information is calculated by the ratio of the distance between the top two reference lines to the x-height.

Figure 22: Image with reference lines detected

6.3 Touching Character String Recognition

As described above, if the confidence value of a connected component recognized by the combined isolated character classifier is not high, then this connected component will be sent to our touching character classifier or Integrated Segmentation and Recognition (ISR) module for further classification.

The touching character classifier

The touching character classifier is a combination of 2-D TCHMM and MLP classifiers for the 57 most frequent touching characters or our special touching character HMM classifier described in section 5.4.2. If the confidence value from the touching character classifier is not acceptable, the connected component will be sent to the ISR module for further classification. Figure 23 shows the diagram of the touching character classifier.

Integrated Segmentation and Recognition of Character String

Generally speaking, in the ISR module, character string is recognized by using heuristic pre-segmentation and dynamic programming (DP) search. A pre-segmentation module is first used to separate the character string image into primitive segments and generate candidate character patterns. A combined character classifier is used to assign character likelihood and class scores to each candidate pattern. A segmentation path is formed by a pattern sequence from the start segment to the last segment such that each segment is used exactly once. The optimal path in the sense of maximum score found in DP search corresponds to the segmentation result of the touching character image.

- Pre-segmentation

  In this pre-segmentation module, our segmentation method can achieve accurate

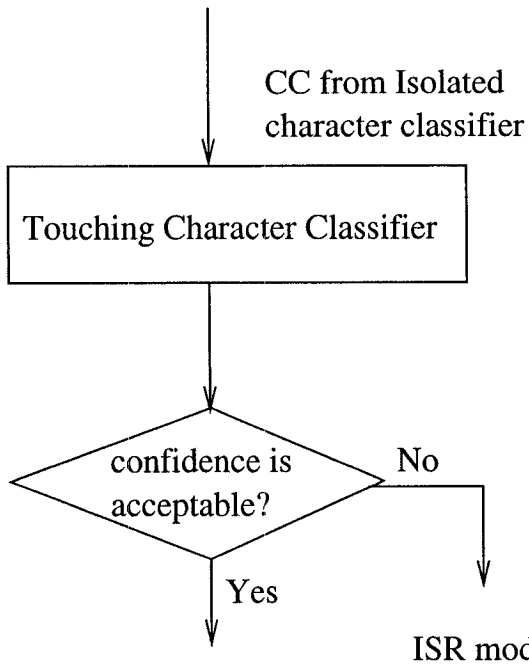


Figure 23: Diagram of touching character classifier

segmentation for both touching normal characters and touching italic characters. It is free of slant correction, so no extra noises will be introduced. The input of this module is a character line image. We first detect the slant angle of the line image, and then with this information, a slant projection is made. From the smoothed histogram of the slant projection, local minimal points are considered as segmentation candidates, and then some candidates are removed by heuristic rules. In addition, we also detect other probable segmentation points by the feature points on the contour. Based on these segmentation points, the shortest path approach is adopted for accurately locating the cut path of each candidate segmentation point. Finally, a confidence value will be assigned to each segmentation point presenting the likelihood of each segmentation point. Figure 24 shows an example of segmenting touching characters of italic font.

Figure 24: Segmentation candidates of touching characters of Italic font

- DP search

The basic DP search is implemented as follows. Each node in the candidate

lattice corresponds to a state in the search space. The path from the start node to an intermediate node corresponds to a partial segmentation-recognition solution of touching character. The accumulated scores of the partial paths are updated according to the preceding nodes of the intermediate nodes. As an illustrative example, Figure 25 shows a graph of a candidate lattice containing four image segments and nine candidate patterns. The node $n3$ is linked to three preceding nodes by edges $e2$, $e5$, and $e7$. Denote the accumulated score of optimal partial path at node n as $D(n)$, the average score is

$$ND(n) = \frac{D(n)}{L(n)}$$

where $L(n)$ is the string length of the partial path, and the maximum class similarity of edge e as $d(e)$. Under the accumulated score criterion, the score of node $n3$ is updated by:

$$D(n3) = \max[D(n0) + d(e2), D(n1) + d(e5), D(n2) + d(e7)]$$

where $D(n0) = 0$. While under the average score criterion, the score is updated by:

$$ND(n3) = \max\left[\frac{D(n0) + d(e2)}{L(n0) + 1}, \frac{D(n1) + d(e5)}{L(n1) + 1}, \frac{D(n2) + d(e7)}{L(n2) + 1}\right]$$

The optimal path at the terminal node $n4$ gives the result of segmentation-recognition. It should be mentioned that for each result from each node a

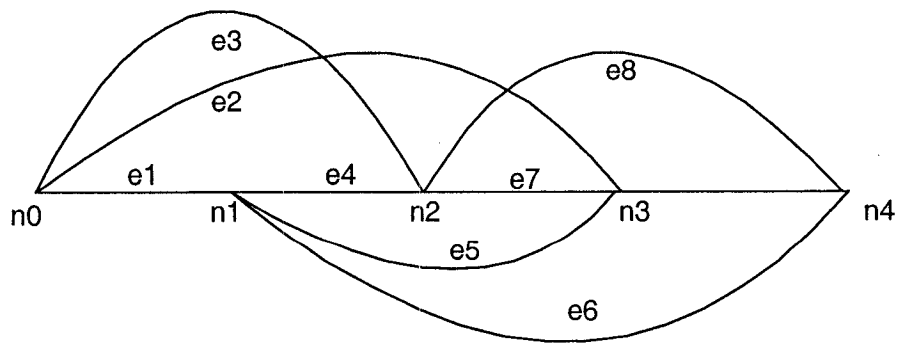


Figure 25: Diagram of DP search

verification module has been used to approximate the best path search and

we will discuss the verification module later in section 6.4. Some preliminary experiments have been carried out to test this method, and very promising results have been obtained.

6.4 Verification

Since in the current system, many errors come from the touching character recognition based on DP search, and some errors come from isolated character recognition for some similar characters (e.g. 'i', 'l', and 't', 'c' and 'e' etc.), two verification modules have been developed to reduce these errors:

1. The first verification module is designed for all classes by using K-Nearest Neighbour (K-NN). This verification module includes a set of character verification model. one for each class. Each character verification model consists of several (5 in our experiment) D_v-dimensional prototype vectors. These prototypes are trained by using the K-means clustering approach from the set of training feature vectors generated from our isolated CENPARMI database (CENPARMI-ISODB1 and CENPARMI-ISODB2) for each character class. Currently, we are using 20 stroke crossing features and 10 profile features (calculating the first differences of profile horizontally and vertically) for the verification model (i.e. $D_v = 30$). After the prototypes are trained, we can estimate threshold parameters for the t-th prototype of the character class C_m as follows:

   - $T_x^v(m,t)$ is the maximum distance of the training feature vectors from the prototype concerned
   - $T_{min}^v(m,t,i)$ and $T_{max}^v(m,t,i)$ are the minimum and maximum values of the i-th feature among the training feature vectors concerned.

   Suppose that the hypothesized character image has been recognized as character C_m. In order to verify whether this recognition result is reliable, a D_v verification feature vector, $F_v = (f_1, f_2, ..., f_{D_v})^t$, is first extracted from the hypothesized character image, and then compared with all the verification prototypes for character C_m to identify the nearest neighbor, say t-th prototype. Let us use $S^v(m,t)$ to denote the Euclidean distance between F_v and the t-th prototype.

Then, we can define a confidence measure for the recognition result C_m as follows:

$$CM(m) = \begin{cases} 0, & \text{if } S^v(m,t) > T_x^v(m,t) \quad or \\ & N_e^m > \frac{D_v}{10} \\ 1, & otherwise \end{cases}$$

where $N_e^m = number$ of $f_i < T_{min}^v(m,t,i)$ or $f_i > T_{max}^v(m,t,i)$ for $i = 1, 2, ..., D_v$. Then $CM(m)$ can be used to determine whether the recognition result for a hypothesized image is reliable. $CM(m) = 1$ means the recognition result is reliable, otherwise it is not.

2. The second verification module is designed and used in the DP search for each segment. To enhance the recognition performance, besides the results from the character classifiers, we have tried to use other sources of information in this verification module to help the DP method to make a better decision. In our current system, the following information has been used:

   - Segmentation cost

     As mentioned above, the confidence values related to the potential segmentation points from the pre-segmentation stage have been added in the DP search stage to evaluate each segment. For hypotheses that generated from segmentation points with lower confidence value, a penalty is applied.

   - Size and Shape information

     We extract some information related to the aspect ratios of different characters from our isolated character training set, and height and width values of some characters in the string line being recognized when these characters can be recognized with a high confidence. For hypotheses on characters that are very short, compared to the average thickness, a penalty is added, with a greater penalty value for classes that are ascenders or descenders.

   - Position information

     For hypotheses on characters that have a large proportion of their height above the upper-line, a penalty is added for classes that are not ascenders. For hypotheses that have a large proportion of their height below the baseline, a penalty is added only for classes that are not descenders [64].

Actually, more information can be integrated into the system. Frequency statistics of touching characters mentioned in section 4.3, frequency analysis of words [58] and knowledge from our online lexicon which will be presented in section 6.5 are some possible sources of useful information.

6.5 Online Lexicon

Our online lexicon is being developed to improve the accuracy of our OCR systems. After the classifier has recognized each text line image, the final recognition result will be refined further with our online lexicon to correct possible errors from classification. It may take care of some pairs of characters having the same shape such as English letters O and zero (0), English letters l and one (1), etc.

The lexicon is in the form of a DLL in which other programs may use to determine the existence of a word (or phrase) in the English language, or to find the closest matches. Our lexicon consists of 207845 words and 4892 word bigrams. The lexicon was reduced from an original size of 600000 words. Rarely used words, archaic/obsolete words, and non-English words were filtered out. The online lexicon can also identify some of the most common proper names. Additionally, it can correct numerals, correct the case of a word, and determine if the word is formed by a merge of two words.

If the input word exists in the lexicon, the program will correct the case (capital or lower) of the word and return success value to the calling program, along with the modified word. If the word does not exist in the lexicon, the program will then find the closest matching words.

In order to determine the closest matches of a word, we send the word through several heuristic functions. Each heuristic adds to a global list, the closest words it was able to find, along with a probability for each word. If a word already exists in the return list, the word is not added, but rather the probability of that word is increased.

Currently, we have a total of thirteen functions. However, we do not always use all of them when processing a word. We stop processing when we have a sufficient list of words, with high enough confidence values. The numeral heuristic will not be applied if there are not enough digits in the word. For example, the word "2i0ns" contains

two digits (2 and 0) less than half the word length, so this word is ignored by this heuristic. The word "3i9" would be changed to "319". An example of the case-fix has is the word "ConCORdia". This word will be changed to "Concordia" and the word "cENParMI" will be changed to "CENPARMI". This case fix applies to all words, not only capitalized words. This heuristic was added when it was discovered that some OCR systems had trouble distinguishing between capital and lower case characters (i.e. "C" with "c", "O" with "o", and "S" with "s").

The online lexicon was able to increase the accuracy of some OCR systems. One system was increased from 91.49% to 94.56%, a second OCR went from 92.90% to 95.19%, and a third OCR increased from 98.26% to 98.86%. The OCRs were tested on 200, 500, and 351 lines respectively.

6.6 Improvement of Hybrid System

Our OCR system for printed English document recognition is a hybrid system which has been presented above. However, it still needs to be improved further. In order to make our hybrid OCR system more robust. We have proposed some improvements:

1. Modified structure of combining 2-D HMM and MLP classifiers
   Since the performance of MLP classifier for isolated character recognition is good and the processing time is shorter than 2-D HMM, the topology of combining 2-D HMM and MLP classifiers has been modified in order to get a better performance in terms of effectiveness and efficiency as shown in Figure 26.

   According to Figure 26, we first try to recognize a character component by the MLP classifier. If the confidence value returned from the classifier is higher than a threshold, the corresponding recognition result is accepted. Otherwise, the combination system of 2-D HMM and MLP is called to recognize the character. In addition, we are trying a weighted Sum rule to combine the NN and HMM classifiers. Some parameters need to be adjusted to get better results.

2. Interaction between Preprocessing and Classification
   As mentioned in section 6.3, the DP search is performed to find the shortest path based on presegmentation points provided by the preprocessing module. However, sometimes the presegmentation points provided by the preprocessing

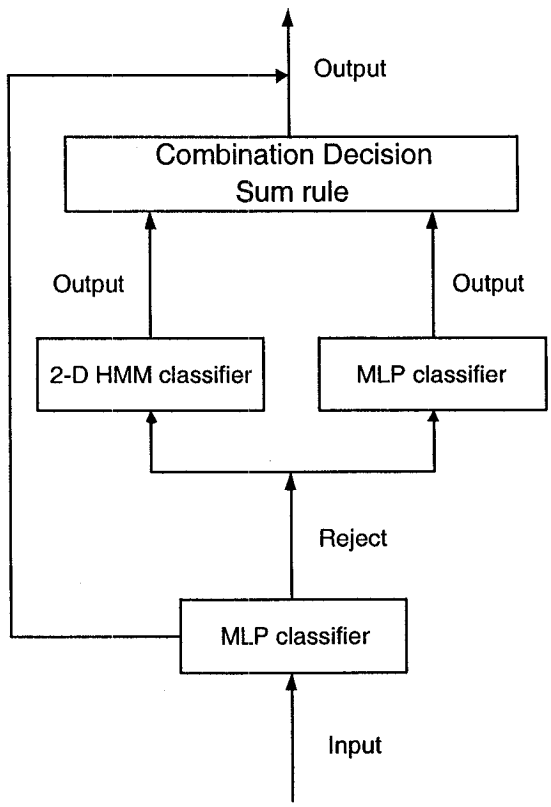


Figure 26: Architecture of the modified system of Hybrid HMM and MLP classifiers

module are not enough such that the DP search can not get the best result in terms of a higher confidence value. To solve this problem, interaction between preprocessing and classification module has been added to our system:

First, the preprocessing module provides presegmentation points to the DP module. If the confidence value from DP search is acceptable, the classification module will output the result. Otherwise, the ISR module will call the preprocessing module again to provide more segmentation points (over-segmentation points) and the DP search will continue to find the best result.

Second, if the preprocessing module detects considerable broken parts which can be acquired during image quality assessment and they dominate the text line image [48], this means the input line image has more broken characters. Then the preprocessing module will send this information to the classification module which will call the preprocessing module to group connected components of each word or the whole line as needed, and the ISR module will continue to recognize each word or whole line based on DP search. Therefore, our improved hybrid

system is capable of handling some degraded documents with broken characters. Figure 27 shows the diagram of the interaction between preprocessing other modules.

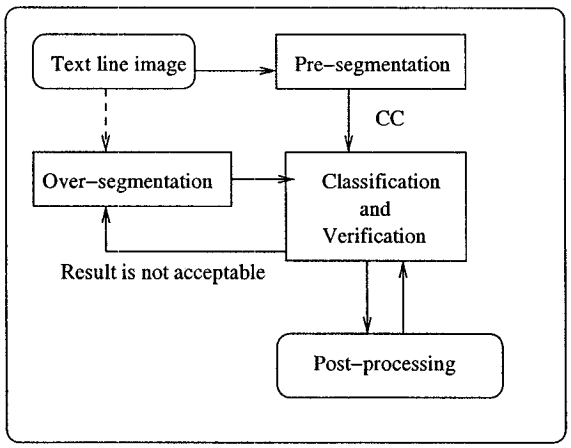


Figure 27: Interaction among preprocessing, classification and post-processing units

3. Interaction between Classification and Post-processing

   Post-processing is a very important stage for all OCR systems. The online lexicon is the main model of post-processing module in our OCR system and it plays a vital role in improving the accuracy of our OCR system. However, if the recognition result has been output to an online lexicon from the classification module, the online lexicon will try its best to correct errors and it still can not correct all errors from the classification module, especially for the recognition of degraded printed documents. Therefore, interaction between classification and online lexicon is important and necessary in our OCR system.

   As mentioned in section 6.3, the confidence values corresponding to each character candidate have been sent simultaneously to the online lexicon while recognition result is output to the online lexicon. When the online lexicon tries to correct the recognition result from the classification module, it also considers the confidence value of each character in some confused words in order to avoid mis-correcting some correctly recognized characters. In addition, the classification module can also make use of the lexicon in order to improve ISR module and it will be included in our future work (Figure 27).

Chapter 7

Experimental Results

In order to verify the efficiency of our 2-D HMM classifiers, a series of comparative experiments for isolated printed character recognition are conducted. Also, the performance of MLP for isolated printed character recognition and the performance of the whole system for text line recognition will be given in the following sections. All the experimental results were tested on a Dell Pentium 4, 2.4G HZ, 1G RAM and 67.6G Hard Drive machine, and we assume all the input images are binary images.

7.1 Isolated character recognition

The data sets for training both HMM and MLP classifiers on isolated character recognition are derived from CENPARMI-ISODB1 and CENPARMI-ISODB2 databases, and one training set (Iso-TrainSet 1) consists of 13650 character samples excluding punctuation, and another training set (Iso-TrainSet 2) contains 25839 samples including punctuation as mentioned in section 4.2. Both Iso-TrainSet 1 and Iso-TrainSet 2 contain 20 fonts, 4 styles (normal, bold, italic and bold italic), and different sizes. The testing sets for isolated character recognition are also derived from the same database. One (Iso-TestSet 1) contains 9672 character samples without punctuation, the other (Iso-TestSet 2) consists of 15218 samples including punctuation (see section 4.2). The testing sets have the same fonts, styles and sizes as the training sets. The training and testing sets for touching character HMM classifier(TCHMM) includes 3936 and 954 touching samples, respectively (see section 4.3). The target of TCHMM is to recognize the 57 most frequent touching characters mentioned in section 5.4.1. The

testing results and error analysis for isolated character recognition are given below. TOP1, TOP2, ..., TOP5 listed in the tables are defined as: TOP1 means input pattern belongs to the first choice of the outputs, TOP2 means input pattern belongs to one of the first two choices of outputs, ..., TOP5 means input pattern belongs to one of the first five choices of outputs. Then we measure the corresponding recognition rates.

1. Testing without punctuation training

Table 6: A Comparison of 5 1-D HMM and 2-D HMM classifiers for the recognition of isolated character and 57 most frequent touching characters without punctuation training

HMM Classifier	No. of Models	TOP1	TOP2	TOP3	TOP4	TOP5
1-D ASCHMM	176	95.18%	97.42%	98.12%	98.40%	98.65%
1-D CENHMM	52	96.01%	97.83%	98.87%	99.21%	99.41%
1-D DESHMM	20	99.10%	99.62%	99.62%	99.62%	99.74%
1-D GENHMM	216	93.43%	96.10%	97.23%	97.75%	98.00%
1-D TCHMM	57	96.37%	97.32%	98.16%	98.69%	99.13%
2-D ASCHMM	156	98.69%	99.63%	99.99%	99.99%	100 %
2-D CENHMM	52	99.75%	100%	100%	100%	100 %
2-D DESHMM	20	100 %	100%	100%	100%	100 %
2-D GENHMM	216	98.42%	99.48%	99.77%	99.88%	99.98%
2-D TCHMM	57	99.37%	99.69%	99.90%	100%	100 %

Table 7: The Top 5 results of of 5 MLP classifiers for the recognition of isolated and 57 most frequent touching characters without punctuation training

MLP Classifier	TOP1	TOP2	TOP3	TOP4	TOP5
ASC MLP	98.25%	99.43%	99.69%	99.79%	99.93 %
CEN MLP	99.90%	100%	100%	100%	100 %
DES MLP	100 %	100%	100%	100%	100 %
GEN MLP	98.59%	99.38%	99.67%	99.89%	99.95%
TC MLP	99.27%	99.59%	99.87%	99.95%	100 %

In order to verify the improvement of our proposed 2-D HMM classifiers compared with 1-D HMM classifier, a series of comparative experiments are conducted. The training and testing sets for four baseline 2-D HMM classifiers used here are Iso-TrainSet 1 and Iso-TestSet 1, respectively. If baseline information can be obtained with a high confidence value from the preprocessing stage, the

9672 isolated character samples can be recognized by three 2-D HMM classifiers (2-D ASCHMM, 2-D CENHMM and 2-D DESHMM) corresponding to Ascender, Center and Descender baseline, respectively. Otherwise, all the 9672 samples are recognized by 2-D GENHMM classifiers. The detailed statistics of the Top 5 testing results on Iso-TestSet 1 for both 1-D HMM and 2-D HMM classifiers are summarized in Table 6. The same experiments for MLP classifiers are shown in Table 7. The error analysis will be given in the error analysis of testing with punctuation training.

- Performance

  From Tables 6 and 7, we can see that the performances of the 5 2-D HMM classifiers are much better than the 5 1-D HMM classifiers. Particularly, the performances of 2-D ASCHMM, 2-D CENHMM, 2-D GENHMM and 2-D TCHMM classifiers. On the other hand, the performances of the 5 2-D HMM classifiers are closer to the 5 MLP classifiers and both the 5 2-D HMM and MLP classifiers have a higher recognition rate on isolated characters. Obviously, Center, Descender and Touching character HMM and MLP classifier have a higher recognition rate than Ascender and General classifiers because the target classes of the former classifiers are much smaller than the latter classifiers. Basically, increasing recognized classes decreases the recognition rate.

2. Testing with punctuation training

Table 8: The TOP 5 results of Ascender and General 2-D HMM classifiers for isolated character recognition with punctuation training

HMM Classifier	No. of Models	TOP1	TOP2	TOP3	TOP4	TOP5
2-D ASCHMM	228	98.45%	99.23%	99.69%	9989%	99.99 %
2-D GENHMM	268	98.32%	99.48%	99.77%	99.88%	99.98%

Table 9: The TOP 5 results of Ascender and General MLP classifiers for isolated character recognition with punctuation training

MLP Classifier	TOP1	TOP2	TOP3	TOP4	TOP5
ASC MLP	98.24%	99.32%	99.67%	99.86%	99.95 %
GEN MLP	98.45%	99.38%	99.68%	99.89%	99.96%

As mentioned in sections 5.3 and 6.2.1, 13 punctuations have been treated as recognition targets of both Ascender and General 2-D HMM and MLP classifiers. For the 13 punctuations, we have collected similar training and testing sets which include 20 fonts, 4 styles and different sizes like the isolated characters (see section 4.2). The training and testing sets are Iso-TrainSet 2 and Iso-TestSet 2 as described above. The testing results of 2 2-D HMM and MLP classifiers on Iso-TestSet 2 are shown in Tables 8 and 9. Here, we only show results of Ascender and General 2-D HMM and MLP classifiers because the punctuations will only affect the performance of these two classifiers and the results of the other 3 2-D HMM and MLP classifiers will remain the same.

- Performance

  In Tables 8 and 9, we can notice that the performances of Ascender and General 2-D HMM classifiers with punctuation training are slightly lower than that of 2-D HMM classifiers without punctuation training. However, the performances of Ascender and General MLP classifiers with punctuation training are almost the same as shown in Table 7. This is because we have added 52 (13x4) more punctuation models to our 2-D ASCHMM and GENHMM and decreased the recognition rate a little bit while only 13 punctuation classes have been added to the ASCMLP and GENMLP classifiers. However, the overall performances of both 2-D HMM and MLP classifiers are still good and similar to each other after training with punctuations.

- Error analysis

  Even though our 2-D HMM and MLP classifiers have achieved a higher performance for isolated character recognition, some errors exist. Based on our experiments on Iso-TestSets 1 and 2, most errors come from similar character image groups for both the HMM and MLP classifiers, such as ('G','C'), ('B','8'), ('i','I'), ('/','l'), (numeral '1' and character 'l'), etc. Some examples of such cases are shown in Figure 28.

  In Figure 28, characters 'B', 'Q', 'i', '/', 'G', '$', 'i' and 'l' have been misclassified as '8', 'a', 'I', 'l', 'C', 'S', 'j' and numeral 'l' respectively because of their similarities. Most of these errors can be corrected by our verification module and online lexicon in string recognition. Very few

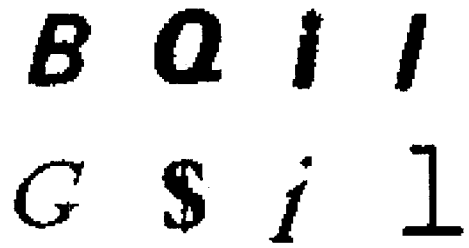


Figure 28: Examples of characters misclassified as 8,a,I,l,C,S,j

errors come from the degraded images (see Figure 29).

Figure 29: Examples of degraded characters misclassified as h,8,F,L

In Figure 29, characters 'H', 's', 'r' and 't' have been misclassified as characters 'h', '8', 'F' and 'L' respectively because original images are degraded. These errors can also be corrected by online lexicon in string recognition.

7.2 Text line recognition

7.2.1 Experiments on Bigram characters

Table 10: A comparison of character recognition accuracies (%) for four recognition systems tested on bigram set(CENPARMI-SPDB)

Systems	Lines	Bigrams	Characters	Accuracy %
FineReader	936	24336	72072	86.89
OmniPage12	936	24336	72072	97.12
TextBridge	936	24336	72072	97.44
Our system	936	24336	72072	98.87

The experiments presented in section 7.1 have shown good recognition rates of our hybrid 2-D HMM and MLP classifiers on isolated characters. However, we need to further verify the performance of our hybrid 2-D HMM and MLP classifier and our segmentation method on text line recognition. Therefore, we have conducted the

Table 11: A comparison of character recognition accuracies (%) for 2 recognition systems tested on bigram set(CENPARMI-SPDB) without lexicon

Systems	Lines	Bigrams	Characters	% Accuracy
TextBridge	936	24336	72072	90.32
Our system	936	24336	72072	98.87

first experiment on Bigrams which contains 24245 (33.4%) touching characters in all text lines. We also compare the testing result of our system on Bigrams with three other commercial OCR software packages in order to evaluate the performance of our classification module. The testing set (TestSetBI) used in this experiment is CENPARMI-SPDB database (see section 4.1), and the detailed testing results of four OCR system under the same condition are summarized in Table 10. Table 11 shows a comparison of our system and TextBridge on the same testing set (Bigrams) but without the lexicon. It should be pointed out that we can not disable the lexicons of the other two commercial OCR systems (FineReader and OmniPage12) in Table 10. Therefore, we can not carry out the same experiments for the other two commercial OCR systems without their lexicon, and we will only give the comparison of our system and Textbridge in subsequent experiments with and without the lexicon. OCR performance is measured using the *percentageAccuracy* defined as

$$(1 - \frac{substitutions + deletions + insertions}{\text{total number of characters in reference}}) * 100\%$$

where the reference is the ground truth.

- Performance

  From Table 10, we can see that the character accuracy of our system without the lexicon is 98.87% which is better than three other commercial OCR systems. This proves that the performance of our hybrid 2-D HMM and MLP classifiers are quite promising, and our ISR module also works well for segmenting touching characters. In Table 11, when we disable the lexicon of Textbridge, its performance is worse (only 90.32%) than using the lexicon. However, our system without lexicon can achieve a much better performance than Textbridge. This also shows that the lexicon plays a vital role in OCR system. In addition, even thought we can not test the other two commercial OCR systems without the lexicon, we can expect that the performance of the other commercial

OCRs without the lexicon should be worse than that with the lexicon, and the performance of our system is still better than theirs.

- Error analysis

  After analyzing the experiment results, we observed that most errors (80%) come from baseline detection errors; some errors come from similar characters such as ('t', 'l' and numeral '1'), ('o' and numeral '0'), ('r' and 'f'), ('I' and 'T'), etc. Very few errors come from segmentation, such as ('fl' to 'n'), ('w' to 'vv'), ('vr' to 'w'), etc. For the baseline detection errors, the reason is because each text line is composed of bigram characters and it is difficult to detect baseline information for those text lines containing all lower case or all capital case bigrams. Therefore, we can only use the General classifier to recognize them. As we mentioned in section 5.3, we have considered each pair of $\{(Cc), (Oo), (Pp), (Ss), (Vv), (Ww), (Xx), (Zz)\}$ as the same class. Usually, these errors can be corrected in real documents. The detailed errors are summarized in Table 12.

Table 12: Distribution (%) of common errors from our OCR system on Bigram test

Error Type	Actual Text	CENPARMI OCR output	% Error
Baseline Error	p	P	80.69
	o	O	
	x	X	
	w	W	
	c	C	
	s	X	
Similar Characters	t	l	11.09
	I	T	
	l	1	
	f	t	
Segmentation	fl	n	4.58
	w	vv	
	rn	m	
	rv	w	
other errors			3.64

Table 13: A Comparison of character recognition accuracies (%) for two recognition systems tested on different sets of text lines

Test Set	Systems	Lines	Chars.	% Acc.
TestSet 1	TextBridge	351	19822	97.12
	TextBridge+Lexicon	351	19822	99.40
from	Our System	351	19822	98.79
CENPARMI-DOCDB	Our System+Lexicon	351	19822	99.53
TestSet 2	TextBridge	100	2380	78.82
	TextBridge+Lexicon	100	2380	82.52
from	Our System	100	2380	88.03
FUJITSU-DOCDB	Our System+Lexicon	100	2380	88.03

7.2.2 Experiments on real documents

In order to further verify the effectiveness of our OCR system for recognition of real English documents of Multi-font and varying qualities, two different comparative experiments are conducted. Two testing sets are used in the experiments. The first test set, TestSet 1, is derived from CENPARMI-DOCDB database (see section 4.1), and 351 text line images are randomly selected from CENPARMI-DOCDB database. 19822 characters are included in TestSet 1. The second test set, TestSet 2 is derived from FUJITSU-DOCDB database (see section 4.1), consisting of 100 degraded text line images, yielding 2380 characters. Since our online lexicon is being developed, we list the testing results of our system with and without the lexicon in order to see the performance of our classification module and the improvement of our online lexicon. In addition, we also compare the results of our system with a commercial OCR (TextBridge). Table 13 shows the detailed testing results. OCR performance is also measured using the *PercentageAccuracy* as described above.

- Performance

  From Table 13, we observed that the performance of commercial OCR with lexicon is better than that of our system on TestSet 1 which is relatively clean. However, the performance of our system without lexicon is better than that of commercial OCR on TestSet 1. In the meantime, the performance of our system is better than that of Textbridge on TestSet 2 which is degraded whether lexicon is used or not. Most importantly, on both TestSets 1 and 2, the performance of our OCR without lexicon is much better than that of Textbridge and much

Table 14: Distribution (%) of common errors from our OCR system on Test Sets 1 and 2

Error Type	Actual Text	CENPARMI OCR output	% Error
Similar Characters	t	l	57.62
	l	i	
	i	l	
	i	j	
Punctuations	l	1	22.29
	h	b	
	.	'	
	,	.	
	I	[	
	/	l	
	l	]	
Segmentation	rn	m	18.07
	fr	k	
	ri	d	
Other errors			2.02

more improvement can be expected from our lexicon in the future because the lexicon of commercial OCRs has improved the classification result (2%) more than our lexicon does (less than 1%). It further proves that our hybrid 2-D HMM and MLP classifier and ISR module are reliable and good for both clean and degraded documents. Finally, If we do not consider the lexicon, our proposed hybrid 2-D HMM and MLP classifier and ISR module can be a good approach for unlimited-vocabulary OCR. Obviously, making use of lexicon and some contextual information can improve the overall performance of OCR.

- Error Analysis

  Based on the experiments on TestSet 1 and TestSet 2 (see section 7.2.2), it has been found that most errors come from similar characters, punctuations and segmentation as shown in Table 14. There, we can also see that more than 57% errors come from similar characters because sometimes it is difficult to distinguish them if the quality of document is not good. In addition, since we did not train our classifiers on some small punctuations, such as '. , ; : -', we only classify these small punctuations according to their structures, positions, sizes, etc, and it is likely to make some mistakes, especially for degraded documents.

In the meantime, for punctuations '[', ']' and '/', it is hard to distinguish them from 'l', 'I' and numeral '1' without contextual information. About 22% errors belong to this type of error. About 18% errors come from segmentation and most of them can be corrected by improving our ISR module and the lexicon. Figures 30 and 31 show some misclassified sample images by our OCR (without lexicon) on TestSet 1 and TestSet 2, respectively.

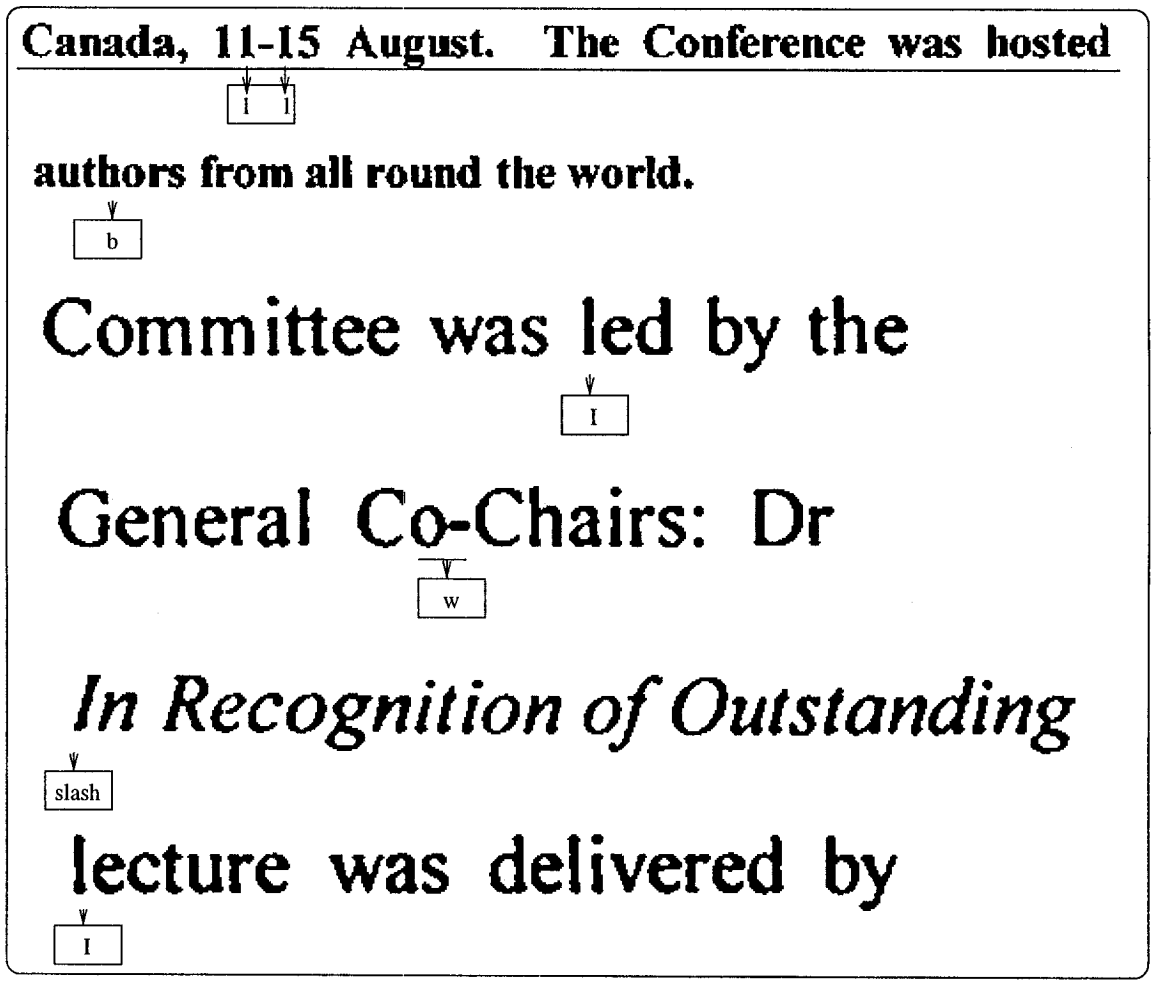


Figure 30: Misclassified error sample images on Cenparmi database

Figure 31: Misclassified error sample images on Fujitsu database

Chapter 8

Conclusion

8.1 Contribution

In this thesis, a hybrid 2-D HMM and MLP based OCR system has been proposed to recognize printed English documents of multi-font and varying qualities. Several new methods have been proposed in the recognition stage in order to improve the performance of character recognition. From the experimental results, the effectiveness of the proposed methods has been proven. The main contribution of this thesis can be summarized as follows:

1. Statistics of frequency of touching characters

   A detailed statistical analysis has been done on analysis of frequency of touching characters and the corresponding touching character database is built and used in our OCR system. To the best of our knowledge, this is the first formal research on analyzing frequency of touching characters and the statistics presented in this thesis can also be used in other research and OCR systems related to the recognition of printed English documents.

2. Touching Character Classifiers

   A segmentation free touching character classifier for both 2-D HMM and MLP on the 57 most frequent touching characters has been built successfully and applied to OCR system based on our statistics of frequency of touching characters. In addition, we have suggested a new method to automatically build special touching character HMMs and some word HMMs to recognize some special touching characters and words.

3. 2-*Direction*(D) HMM.

   A 2-D HMM with multi-model corresponding to different character styles (normal, bold, italic and bold italic) has been designed and applied to improve the character recognition rate, and a significant improvement has been achieved compared with 1-D HMM.

4. Impove the Hybrid HMM and MLP classifier

   The 2-D HMM has been combined with MLP classifier to improve the original version [65]. At the same time, different 2-D HMM and MLP classifiers based on baseline information (Ascender, Descender, Center and General) and trained on multi-font database have been built and combined together to improve the recognition rate. The hybrid 2-D HMM and MLP classifier has shown stable and good classification performance.

5. Improve the Integrated Segmentation and Recognition module.

   In order to recognize both clean and degraded printed documents, an image quality assessment technique [48] and a combination of segmentation-free (touching character classifier) and segmentation-based method has been used to improve performance of our OCR system. For the segmentation problem, an integrated segmentation (pre-segmentation and over-segmentation) and recognition module combined with verification has been improved based on the original ISR version [65]. The overall performance of our OCR is quite impressive and promising.

6. Database establishment

   A touching character database has been built for the recognition of the 57 most frequent touching characters.

8.2 Future Work

Though our OCR system has achieved promising performance, it is still far from real application, and more work related to the following aspects may be done in the future.

- Improve the preprocessing module

  The baseline reliability should be further improved when degraded documents

occur. Currently, the baseline detection uses very strict threshold and may fail to detect some baselines in degraded documents, leading to a decrease of the performance of our system. In addition, image quality assessment and filtering selection should be further improved to handle degraded documents, especially for broken characters.

- Improve punctuation engine
  Currently, some errors coming from punctuations still exist and further improvement should be made to strengthen the punctuation engine. In addition, our current classifiers can only recognize 20 punctuations, and the remaining 10 punctuations, such as '#', '+', '{', '}', etc. should be added to our classifiers and more experiments should be conducted to support all punctuations.

- Improve ISR module
  Even our current ISR module works well, further improvement can be made by integrating more knowledge, e.g. frequency of touching characters, frequency of bigrams, trigrams and words could be two aspects of improvements. In addition, making use of lexicon in ISR module could be another aspect of improvements.

- Improve online lexicon
  Based on the experimental results, our lexicon only improves the classification result a little while commercial OCR system has made much contribution to the overall performance. Therefore, more works should be done to improve our lexicon and much better performances can be expected in the future.

- Study other strategies for printed document recognition
  Some other techniques, e.g. the fully HMM-based approach used in [19, 20] and Segmentation and MCE negative training based approach used in [14] to recognize printed English documents, should be investigated and compared with our current method. Some ideas from these methods can be employed to improve our system.

- Preparing for real application
  The final goal of the main modules of this system is for real application. As a result, our hybrid OCR has to be further trained and tested on larger databases, and more real-life data need to be collected in the future. In addition, the

processing time should be considered as another important factor for practical applications, and therefore refinement in each stage should be made to speed up the whole system.

Bibliography

[1] Nagy. G. Twenty years of document image analysis in pami. *Pattern Analysis and Machine Intelligence*, 22:38–63, January 2000.

[2] Mori. S., Suen. C.Y., and al. Historical review of OCR research and development. In *Proc. of the IEEE*, pages 1029–1058, July 1992.

[3] T. Pavlidis. Recognition of printed text under realistic conditions. *Pattern Recognition Letters*, 14:317–326, 1993.

[4] N. Arica and F. Yarman-Vural. An overview of character recognition focused on off-line handwriting. *IEEE transaction on Systems,Man and Cybernetics*, 31:Part C, May 2001.

[5] C.Y. Suen, M. Berthod, and al. Automatic recognition of handprinted characters-the state of the art. In *Proc. of the IEEE*, volume 68, pages 469–487, 1980.

[6] Luis R. Blando, Junichi Kanai, and Thomas A. Nartker. Prediction of OCR accuracy using simple image features. In *Proc. of the Third International Conference on Document Analysis and Recognition*, volume 1, pages 319–322, August 1995.

[7] M. Bokser. Omnidocument technologies. In *Proc. of the IEEE*, volume 80, pages 1066–1078, 1992.

[8] T.A. Naster and et. al. A prelimiary report on OCR problems in LSS Document Conversion,Presented in TR 92-04,ISRI. Technical report, University of Nevada, April 1992.

[9] R.G. Casey and G. Nagy. Decision tree design using a probabilistic model. In *IEEE Transactions on Information theory*, volume 30, pages 93–99, January 1984.

[10] Marcel Brun, Junior Barrera, and Nina S.T. Hirata. Multi-resolution classification trees in OCR design. In *Proc. of the XIV Brazilian Symposium on Computer Graphic and Image Processing*, pages 59–66, October 2001.

[11] Fang Yang, Xue-Dong Tian, and Bao-Lan Guo. An improved font recognition method based on texture analysis. In *Proc. of 2002 International Conference On Machine Learning and Cybernetics*, volume 4(1), pages 1726–1729, November 2002.

[12] Y. Ishitani. Model-based information extraction method tolerate OCR errors for document images. In *Proc. of the Sixth International Conference on Document Analysis and Recognition*, pages 908–915, September 2001.

[13] Jonathan J. Hull, Siamak Khoubyari, and Tin Kam Ho. Word image matching as a technique for degraded text recognition. In *Proc. of the 11th International Conference on Pattern Recognition Methodology and Systems*, volume 2, pages 665–668, August 1992.

[14] Qiang HUO and Zhi-Dan FENG. Improving Chinese/English OCR Performance by using MCE-based Character-Pair Modeling and Negative Training. In *Proc. of the Seventh International Conference on Document Analysis and Recognition*, pages 364–368, August 2003.

[15] Zhi-Dan FENG and Qiang HUO. Confidence guided progressive search and fast match techniques for high performance Chinese/English OCR. In *Proc. of the International Conference on Pattern Recognition*, volume 3, pages 89–92, August 2002.

[16] L.R. Rabiner. Turorial on hidden markov model and seletected application in speech recognition. In *Proc. IEEE 77(2)*, volume 2, pages 257–285, 1989.

[17] Lawrence Rabiner and Biing-Hwang Juang. *Fundamentals of Speech Recognition.* PTR Prentice Hall, 1993.

[18] V. Margnet and M. Pechwitz. Synthetic data for Arabic OCR system development. In *Proc. of the Sixth International Conference on Document Analysis and Recognition*, pages 1159–1163, September 2001.

[19] Issam Bazzi, Richard Schwartz, and John Makhoul. An Omnifont Open-Vocabulary OCR System for English and Arabic. In *IEEE Transactions on Pattern Analysis and Machine Intelligence*, volume 21, pages 495–504, June 1999.

[20] Premkumar Natarajan, Issam Bazzi, Zhidong Lu, John Makhoul, and Richard Scwhartz. Robust OCR of degraded documents. In *Proc. of the Fifth International Conference on Document Analysis and Recognition*, pages 357–361, September 1999.

[21] Issam Bazzi, Chris Lapre, John Makhoul, and Richard Schwartz. Omnifont and unlimited-vocabulary OCR for English and Arabic. In *Proc. of the Fifth International Conference on Document Analysis and Recognition*, volume 2, pages 842–846, August 1997.

[22] Datong Chen, Jean-Marc Odobez, and Herve Bourlard. Text segmentation and recognition in complex background based on Markov random field. In *Proc. of the 16th International Conference on Pattern Recognition*, volume 4, pages 227–230, 2002.

[23] Prem Natarajan, Baback Elmieh, Richard Schwartz, and John Makhoul. Video-text OCR using Hidden Markov Models. In *Proc. of the Sixth International Conference on Document Analysis and Recognition*, pages 947–951, September 2001.

[24] Jonathan J. Hull. A hidden markov model for language syntax in text recognition. In *Proc. of the 11th International Conference on Pattern Recognition Methodology and Systems*, volume 2, pages 124–127, August 1992.

[25] Chinmoy B. Bose and Shyh-Shiaw Kuo. Connected and degraded text recognition using Hidden Markov Model. In *Proc. of the 11th International Conference on Pattern Recognition Methodology and Systems*, volume 2, pages 116–119, August 1992.

[26] Michael D. Garris, Charles L. Wilson, and James L. Blue. Neural network-based systems for handprint OCR applications. In *IEEE Transactions on Image Processing*, volume 7, pages 1097–1112, August 1998.

[27] Tai-Wen Yue, Yu-Jen Chang, and Chien-Wu Tsai. Equip a Gaussian-vector-field feature extracting mechanism to an MLP for optical character recognition. In *IEEE International Conference on Systems, Man and Cybernetics*, volume 3, pages 2295–2300, October 1995.

[28] E. Filippi, M. Costs, and E. Pasero. Multi-layer perceptron ensembles for increased performance and fault-tolerance in pattern recognition tasks. In *IEEE International Conference on Neural Networks*, volume 5, pages 2901–2906, July 1994.

[29] H.D. Block, B.W. Knight, and F. Rosenblatt. Analysis of a four layer serious coupled perceptron. *Rev. Mod. Phys.*, 34:135–152, 1962.

[30] M.L. Minsky and S. Papert. Perceptrons: An introduction to computational geometry. *Cambridge,MA:MIT Press.*

[31] L. Xu, A. Krzyzak, and Ching Y. Suen. Methods of combining multiple classifiers and their applications to handwriting recognition. *IEEE Transactions on System, Man and Cybernetics*, 22:418–435, 1992.

[32] A.F.R. Rahman and M.C. Fairhurst. An evaluation of multi-expert configurations for the recognition of handwritten numerals. *Pattern Recognition*, 31(9):1255–1273, 1998.

[33] C.Y. Suen and L. Lam. Multiple classifier combination methodologies for different output levels. In *Proc. of the first Int. Workshop on Multiple Classifier Systems*, pages 52–66, Cagliari,Italy, June 2000.

[34] L. Lam. Classifier combinations: Implementations and theoretical issues. In *Proc. of the first Int. Workshop on Multiple Classifier Systems*, pages 77–86, Cagliari,Italy, June 2000.

[35] J. Kittler. On combining classifiers. *IEEE Trans. on Pattern Anal. and Mach. Intell*, 20(3):226–239, March 1998.

[36] Y.S. Huang and C.Y. Suen. A method of combining multiple experts for the recognition of unconstrained handwritten numerals. *IEEE Trans. on Pattern Anal. and Mach. Intell*, 17(1):90–94, 1995.

[37] S. Procter and J. Illingworth. Combining HMM classifiers in a handwritten text recognition system. In *Proc. of the International Conference on Image Processing*, volume 2, pages 934–938, October 1998.

[38] Akihiro Nomura, Kazuyuki Michishita, Seiichi Uchida, and Masakazu Suzuki. Detection and segmentation of touching characters in mathematical expressions. In *Proc. of the Seventh International Conference on Document Analysis and Recognition*, pages 126–130, August 2003.

[39] Utpal Garain and Bidyut B. Chaudhuri. Segmentation of touching characters in printed Devnagari and Bangla scripts using fuzzy multifactorial analysis. In *IEEE Transactions on System, Man and Cybernetics-Part C*, volume 32, pages 449–459, November 2002.

[40] Richard G. Casey and Eric Lecolinet. A survey of methods and strategies in character segmentation. In *IEEE Transactions on Pattern Analysis and Machine Intelligence*, volume 18, pages 690–706, July 1996.

[41] Jin-Ho Kim, Kye-Kyung Kim, Sung-II Chien, and Heong-Moon Choi. Segmentation of touching characters in printed Korean/English document recognition. In *IEEE Transactions on System, Man and Cybernetics*, volume 1, pages 438–443, October 1996.

[42] Su Liang, M. Ahmadi, and M. Shridhar. Segmentation of touching characters in printed document recognition. In *Proc. of the Second International Conference on Document Analysis and Recognition*, pages 569–572, October 1993.

[43] Yi Lu. On the segmentation of touching characters. In *Proc. of the Second International Conference on Document Analysis and Recognition*, pages 440–443, October 1993.

[44] R.L. Hoffman and J.W. Mccullough. Segmentation methods for recognition of machine-printed characters. *IBM journal of Research and Development*, pages 153–165, March 1971.

[45] Tsujimoto and H. Asada. Resolving ambiguity in segmenting touching characters. In *ICDAR*, pages 701–709, 1991.

[46] C. Vasantha Lakshmi and C. Patvardhan. A Multi-font OCR System for Printed Telugu Text. In *Proc. of the Language Engineering Conference*, pages 7–17, December 2002.

[47] J. Barrera and R. Terada et al. An OCR based on mathematical morphology. In *Proc. of SPIE in Nonliner Image Processing IX*, volume 3304, pages 197–208, San Jose,CA, January 1998.

[48] Andrea Barretto de Souza. Automatic filter selection using image quality assessment. Master's thesis, department of computer science at Concordia University, May 2003.

[49] A. Souza, M. Cheriet, and C.Y. Suen. Automatic filter selection using image quality assessment. In *Proc. of the Seventh International Conference on Document Analysis and Recognition*, pages 508–512, August 2003.

[50] J. Wang and J. Jeans. Segmentation of merged characters by neural networks and shortest path. *Pattern Recognition*, 27:649–658, May 1994.

[51] S. Liang, M. Shridhar, and M. Ahmadi. Segmentation of touching characters in printed document recognition. *Pattern Recognition*, 27(6):825–840., 1994.

[52] Nenghong Fu. Statistics of touching character frequency. Technical Report at CENPARMI Presented in Fujitsu project for Fujistu Laboratories Ltd., August 2002.

[53] Dahai Cheng and Hong Yan. Recognition of handwritten digits based on contour information. *Pattern Recognition Society*, 31:235–255, 1998.

[54] Y.Linde, A.Buzo, and R.M Gray. An algorithm for vector quantiser design. *IEEE Transaction. Commun.*, COM-28:84–95, 1980.

[55] J. G. Wilpon and L.R. Rabiner. A modified K-means clustering algorithm for use in isolated word recognition. *IEEE Transaction. on Acoustics, Speech, and Signal Processing*, ASSP-33:587–594, 1985.

[56] L.E. Baum. An inequality and associated maximization technique in statistical estimation for probabilistic functions of Markov processes. Inequality 3(1970) 1-8, 1970.

[57] H. Othman and T. Aboulnasr. A separable low complexity 2D HMM with application to face recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 25:1229–1238, October 2003.

[58] John B.Carroll, Peter Davies, and Barry Richman. *Word Frequency Book.* American Heritage Publishing Co., Inc., 1971.

[59] A.P. Dempster, N.M. Laird, and D.B. Rubin. Maximum likelihood from incomplete data via the EM algorithm. *J. Roy. Stat. Soc*, 39:1–38, 1977.

[60] Claudio Becchetti and Lucio Prina Ricotti. *Speech Recognition Theory and C++ Implementation.* John Wiley and Sons, 2002.

[61] Mohammad B. Menhaj and Mahdi Adab. Simultaneous segmentation and recognition of Farsi/Latin printed texts with MLP. In *Proc. of the 2002 International Joint Conference on Neural Networks*, volume 2, pages 1534–1539, May 1993.

[62] Qizhi Xu. *Automatic Segmentation and Recognition system for Handwritten Dates and Cheques.* PhD thesis, department of computer science at Concordia University, 2003.

[63] Hongwei Hao, Cheng-Lin Liu, and Hiroshi Sako. Confidence evaluation for combining diverse classifiers. In *Proc. of the Seventh International Conference on Document Analysis and Recognition*, pages 760–765, August 2003.

[64] C. Scagliola, G. Nicchiotti, and F. Camastra. Enhancing cursive word recognition performance by the integration of all the available information. In *Proceedings of the Seventh International Workshop on Frontiers in Handwriting Recognition*, pages 363–372., Amsterdam, September 2000.

[65] Qizhi Xu. Progress report for neural network classifiers and a hybrid nn and hmm classifier for machine-printed character string recognition. Technical Report at CENPARMI Presented in Fujitsu project for Fujistu Laboratories Ltd., August 2003.