ON-LINE CHARACTER RECOGNITION OF HANDPRINTED CHINESE CHARACTERS USING FUZZY MEASURING AND STRUCTURAL ANALYSIS

By

SONG-SHEN YEH

Bachelor of Science
National Taiwan College of Marine
Science and Technology
Taiwan, R. O. C.
1985

Master of Science
Oklahoma State University
Stillwater, Oklahoma
1990

Submitted to the Faculty of the
Graduate College of the
Oklahoma State University
in partial fulfillment of
the requirements for
the Degree of
DOCTOR OF PHILOSOPHY
May, 1995

ON-LINE CHARACTER RECOGNITION OF HANDPRINTED CHINESE CHARACTERS USING FUZZY MEASURING AND STRUCTURAL ANALYSIS

Thesis Approved:

Huizhu Lu
Thesis Adviser

J Chandler

Marilyn G. Kletke

Thomas C. Collins
Dean of the Graduate College

ACKNOWLEDGMENTS

I sincerely thank Dr. Huizhu Lu for her patience, inspiration and consistent support during my dissertation work. I am also grateful for her kindness and encouragement for hiring me as a research assistant during projects with the Oklahoma State Health Department.

I would like to express my deepest gratitude to my committee members, Dr. John P. Chandler, K. M. George, and Marylin G. Kletke for their critical reviews, valuable suggestions, and the time they invested in the materials. Without their assistant and patience, I would have never made it.

I wish to express my gratitude to the Department of Computer Science for providing me with a teaching assistantship during these years. I would like to extend my thanks to all of my friends, colleagues, and staff members, especially Anna Ventris and Janice Bryan, for their assistance.

Above all, I want to thank my parents, Hsiu-pin Yeh and Fu-mei Pen, for their support, confidence and love throughout these years. With gratitude, my love and thanks go to my wife, Tomoko Akao, and my son, Ningche K. Yeh, for their love and infinite patience during my graduate studies.

TABLE OF CONTENTS

Chapter Page

Chapter Page

LIST OF TABLES

LIST OF FIGURES

NOMENCLATURE

A	the smallest square that circumscribes an input character
W_A	width of the smallest square A
$S(x_s, y_s)$	start point of a stroke
$C(x_c, y_c)$	corner point of a stroke
$M(x_m, y_m)$	middle point of a stroke
$E(x_e, y_e)$	end point of a stroke
$S'(x'_s, y'_s)$	normalized start point of a stroke
$E'(x'_e, y'_e)$	normalized end point of a stroke
$M'(x'_m, y'_m)$	normalized middle point of a stroke
N	size of normalization
L_i	length classification of the ith stroke of a character
T_i	type classification of the ith stroke of a character
P_i	position classification of the ith stroke of a character
D_i	direction classification of the ith stroke of a character
l_i	physical measurement of length of the ith stroke of a character
θ_{ti}	physical measurement of type of the ith stroke of a character
p_i	physical measurement of position of the ith stroke of a character
θ_{di}	physical measurement of direction of the ith stroke of a character
$\overline{SE}$	a straight line segment that connect start point S and end point E of a

stroke

μ_{L_j}	membership function for length measure
μ_{T_j}	membership function for type measure
μ_{P_j}	membership function for position measure
μ_{D_j}	membership function for direction measure
$\mu_{L_j=k}$	membership function for length measure of k classification
$\mu_{T_j=k}$	membership function for type measure of k classification
$\mu_{P_j=k}$	membership function for position measure of k classification
$\mu_{D_j=k}$	membership function for direction measure of k classification
μ_{ij}	similarity measure of the ith input stroke to the jth template stroke concerning local features
v_i	primitive stroke type of the ith input stroke
v_j	primitive stroke type of the jth mask stroke
c_{ij}	degree of confidence corresponding to the primitive stroke types from the ith input stroke to the jth mask stroke
r_{ij}	similarity measure from the ith input stroke to the jth template stroke corresponding to local and structural features
R	rating matrix

CHAPTER I

INTRODUCTION

1.1 Background

Machine replication of human abilities has been of interest to researchers for decades. One of the major fast growing research areas is replication of human reading. This involves domain knowledge of character recognition and requires intensive study of visual perception to accomplish the task. The origin of character recognition can be traced back to 1870. It was first designed to aid the visually handicapped with the breakthrough of the sequential scanner. In 1900, it was then successfully applied by the Russian scientist Tyurin [6]. The current version of character recognition comes after the development of the digital computer in the middle 1940s. Thereafter, much work has been done by researchers from all over the world and for all different kinds of character sets.

1.2 Motivation

Over the years, researchers have tried every effort to find better solutions to handprinted character recognition. These solutions include searching for robust and fast algorithms, developing special purpose hardware, designing new feature extraction techniques, etc. These attempts do give us clues about the next generation of character

recognition systems, but they are not yet satisfactory. The recognition rates of handprinted character recognition systems need to be improved. Fast and improved feature extraction methodologies need to be not only developed but also simplified. The innovations of the hardware need to be combined with the recent advances in software technologies. Also, new aspects, such as quality factors [14], human knowledge [40], and dynamic information [41], need to be considered during the design of a recognition system. It may take another decade or even more to find solutions to all of these. Although the recent improvements are not yet satisfactory, they are very encouraging.

Among very different character sets, the recognition of handprinted Chinese characters is particularly challenging. This is due to the following reasons.

(1) The Chinese character set contains a large amount of characters. There are more than 42,000 characters in total and about 3,000 - 5,000 in daily use.

(2) The complexities of most of the Chinese characters are high.

(3) There is a large amount of similarities among different categories.

(4) There are great variations among different writers.

Another obvious motivation for constructing a character recognition system comes from the potential applications and their market share. Character recognition technology has enjoyed a very successful history of applications and has been applied in the practical world for decades. Many articles as well as researchers have covered various applications of this fast growing technology [3,10]. These applications can be discerned two main streams: on-line and off-line. On-line character recognition means that the recognition process is carried out while a symbol is drawn, while off-line

character recognition means that the recognition process is carried out after the symbols are drawn.

On-line recognition requires a transducer to capture the writing. The most often used interface is the tablet or digitizer. Real time and dynamic information are provided because of the nature of the on-line recognition. Moreover, the speed of recognition need only be fast enough to keep up with the writing speed. On-line recognition has several applications; e.g., computer-aided design (CAD), on-line data input, handwritten programs, mathematical symbols and formulas, and learning by computer. Syntactic or logic methods are more frequently used in on-line recognition due to the close relation of the shape of patterns and properties of features.

Off-line recognition requires an optical reader to convert images after drawing. A typical optical scanner provides resolution of 300-400 dpi. Since the process is carried out after the drawing of images, the dynamic information is no longer available for processing. Off-line character recognition is a subset of optical character recognition (OCR) by the utilization of the optical readers. The requirement of recognition speed is usually faster than on-line because the processing of a document may seem to take too long. Off-line recognition consists of applications such as reading tools for the blind, communication tools for the deaf, postal address reading, document or form conversion and analyzing, bar code reading and signature verification. Statistic or structural approach is often used due to the associated linguistic and contextual information.

Due to the restrictions of current methods, the challenge of handprinted Chinese characters, and the potential applications, we conclude that handprinted Chinese character recognition is a research field worth continuous study. The above reasons

have led us in investigating and in designing a Chinese handprinted character recognition system.

1.3 The Properties of Chinese Character

Chinese characters are neither alphabetical nor phonetic. They are monosyllabic words that form polysyllabic phrases. According to the Kangxi Dictionary, there are about 42,000 characters in total and about 3,000 to 5,400 in daily usage. Each character represents a word and may either be itself a radical or be decomposed into a radical plus a residue. Moreover, a character may be further decomposed into line segments called strokes (from pen-down to pen-up). Most strokes are written as horizontal, vertical, or diagonal; closed curves are not allowed. However, certain exceptions may apply as long as general writing rules are followed; these rules are top-to-bottom and left-to-right.

Although the writing of the Chinese character is taught formally in schools, the order of writing for each character may vary from writer to writer. This is owing to the various writing styles and habits adapted from the surroundings. Every Chinese character has a unique standard sequence of writing, but not every writer follows the standard sequence. This complicates the identification of a character based on stroke order. In the next section, a generalized character recognition system is presented.

1.4 A Generalized Character Recognition System

Research articles that survey recent improvements in optical character recognition technologies [1-13] are quite a few in number. Though the methodologies are applied to different symbols or characters, they are essentially the same. From this

point of view, we wish to sketch the hierarchy of a generalized character recognition system.

The central issue of character recognition is to emulate human visual perception. There is no doubt that two goals are set forward to be achieved -- speed and accuracy (though these are not always possible). With various writing styles and habits, even the best optical readers, eyes, have difficulties in recognition. It is thus very interesting to know how humans accomplish the task and why we sometimes make mistakes. The major challenge of handprinted character recognition is due to variations in handwriting. We may say that no one will be able to write a character or a word exactly the same twice. This is owing to the surroundings, mood, habit and other factors.

To accomplish certain tasks in the same way as humans do, a character recognition system must first tolerate variations among writers (theoretically). It then performs built-in methodologies to recognize input characters. A diagram in Figure 1.1 shows the hierarchy of a general purpose recognition system. This recognition system is based on the consideration of the basic requirements from various reviews [2,5,9,10]. The components of this generalized recognition system are described as follows.

1.4.1 Digitizer

A row image must be digitized through a scanner or a tablet before it can be further processed. In most cases, a row image consists of a matrix of at least 32x32 pixels. Each element in the matrix may be either multi-gray levels or binary levels. Usually, a scanned image is converted from multi-gray levels into binary levels to optimize the system performance. This conversion not only saves the memory

requirements during processing, but also speeds up the time for computation. Thereafter, the digitized image is sent to the preprocessor for further processing.

1.4.2 Preprocessor

To process the digitized image, the preprocessor plays an important role in a character recognition system. Without the preprocessing stage, a digitized image may contain noise from the sample itself or from the input device. These unwanted data may cause serious problems at a later stage. The objectives of preprocessing are threefold:

(A) to separate digitized characters,

(B) to eliminate wild noise, and

(C) to normalize an input pattern.

To separate digitized characters, a segmentation algorithm must be performed. It is system dependent whether a segmentation operation is applied or not. Some recognition systems require writers to write in a box. This approach does not require the segmentation operation with a simple constraint. However, this approach may be inconvenient to writers during writing. According to Nouboud [9], there are three ways to segment a character. The first method is to segment a character into elements, where an element is described by length, and angles concerning the horizon. The second approach to segment a character is to identify the curvature maxim or the local extreme in x and y axes. The last method is to separate a character into components where pen lifts occur. Thus, it is more suitable for on-line character recognition rather than for off-line character recognition. Notice that the process of character segmentation becomes

simple if a time constraint is defined. Therefore, most refined segmentation mechanisms are developed for off-line applications [25-29].

It is often the case that a digitized image contains wild noise or the data itself is distorted. To eliminate this kind of noise, some operations are required for individual recognition systems. These operations include noise reduction and image smoothing. The central issue of both operations is to reduce the amount of information to be stored and to remove wild noise generated by users or by the hardware itself. Two major smoothing processes need to be employed after filtering the data: filling and thinning. Filling is used to get rid of breaks, gaps and holes in patterns, and thinning is used to remove noise, bumps, and isolated bits. Recently, several researchers [30,31] have tried to implement some of the above-mentioned operations on parallel machines. Eventually, these new methods will be improved and applied to practical applications.

By normalizing an input pattern, we mean that data representing the character may be normalized in terms of size, orientation, or position. It is not required for every recognition system to perform normalization for every aspect, yet it is common to most character recognition systems to perform specific normalization for special needs. Pointers to references for normalization of size, orientation or position can be found in article [9]. In case of correlation matching, where a linear normalization is inadequate due to irregular variations in handprinted characters, non-linear normalization methods [32,33] can be used to produce more stable results.

Güdsen [16] presented a survey of preprocessing techniques. These techniques consist of filling in holes, skeletonization, centering, displacing, rotation, shearing rows and columns, thickening, and size normalization. The recognition rate of a character

recognition system highly depends on the applied preprocessing methodologies. This is because an unstable preprocessing operation may inadvertently remove the data we want. Hence, an input will no longer possess distinct properties that may be extracted by the feature extractor.

1.4.3 Feature Extractor

Each digitized character has its own distinctive properties. These properties must be extracted to match with those stored in the database. From the classification point of view, the extraction of features plays an important role in a recognition system. It also implies that a successful recognition system relies on both the feature extractor and the classification recognizer.

1.4.4 Recognizer

A recognizer performs matching among input characters and masks. The central issue of a recognition system is the matching process. With certain information obtained from both the input and the mask, a recognizer is able to determine the degree of similarity between them. Intuitively, the variations between different characters are more important than the variations between the same character. This is particularly true for Chinese characters that contain complex characters and form a large character set. To simplify this matter, many researchers suggest implementing recognition systems for Chinese in two phases. The first phase is to preclassify input characters into different classes. The second phase is to apply an in-depth comparison technique to measure the similarity between input and masks within the same class. The major reason to subdivide

the recognition process into two phases is to reduce the comparison time by discarding irrelevant characters. This complicates the design of a recognition system for Chinese since a good preclassification methodology for a large set of characters is usually hard to find.

There are various matching techniques available for matching among characters; for example, dynamic programming, linear programming, relaxation, string and graph matching, neural networks, etc. These methodologies usually depend on the extracted features from both input and masks. A thorough evaluation of recognition methodologies as well as advantages and disadvantages of features is of great help in developing a recognition system.

1.4.5 Postprocessor

Though the recognition rates for some of the character recognition systems have reached up to 99%, it is still not satisfactory compared to the visual perception of human beings. Many researchers [118] have tried to improve with postprocessing techniques provided that contextual or linguistic information is known after recognition. A dictionary look-up with embedded search structure that provides matching of phrases is a general linguistic approach. Another approach is training; most of the off-line character recognition systems provide training capabilities. This technique is very flexible with the trade-off of time.

Figure 1.1: Diagram of the generalized character recognition system.

1.5 Problems and Strategies

From the above review of a generalized character recognition system and its background, it is clear that there still remain some problems. The problems of the recognition of handprinted Chinese characters stem from both static and dynamic points of view. By static, Chinese characters are multi-categorized, complex, and similar among different categories. By dynamic, the writing variations of a character may vary from writer to writer, and the order of writing may depend on the training received by the writer. Even a well-trained native Chinese does not guarantee that all characters are written in the correct sequence. In the following, we discuss some of the solutions that have been applied to such problems as these by researchers.

1.5.1 Multitude of Categories and Complexity of Characters

As mentioned earlier, Chinese characters constitute a large set. This character set can be classified into many categories. This property is the cause of one of the major problems in recognizing Chinese characters. A promising strategy to solve this problems is to preclassify or to cluster this large and complex character set into small groups. Some well-known preclassification schemes for off-line recognition, such as orthogonal expansion, stroke distribution, stroke analysis and background feature distribution can be found in [7,34-36]. As for on-line recognition, stroke based methods [119], where structure features are classified during dynamic writing, are also available.

The other problem is the complexity of Chinese characters. More than 10% of Chinese characters contain more than 20 strokes, such as 儷, 鑾, and 鬻. If the

representation of a character is complex, the size of the template database may be very large. As a result, it may affect the matching speed and consume disk spaces on the secondary storage.

1.5.2 Similarity among Different Categories

Some Chinese characters are very similar, such as 尤 and 犬, 土 and 士. If distinct features for each category can be extracted, the similarities among different categories will not cause too many problems. Notice that it is not always possible to extract distinct features to distinguish two similar characters from each other. Some researchers employed probability and statistical information to find the best candidate. Nevertheless, it is not always possible to obtain statistical information before hand in a practical application. In such case, post-processing techniques [118] that provide contextual or linguistic information are another approach.

1.5.3 Wide Variety of Writing Variations

The same problem existed among handprinted recognition in other character sets is the wide variety of writing variations among writers. There are too many factors that affect the writing. To name a few, they are culture, education, habit, mood, surroundings, writing devices, or even time. If an on-line application heavily relies on the order of strokes, the system performance becomes critical unless writing constraints are applied before recognition. In such case, adaptive learning may also be used to learn from experience as well. Notice that the recognition rate becomes very unstable when a

character is ill-written. How to manage the writing variations is still a challenging research topic.

1.6 Objectives of the Study

The major research objective is to develop a prototype of an on-line handprinted Chinese character recognition system. Figure 1.2 sketches the components of the proposed on-line handprinted Chinese recognition system; the dotted block diagram is the major concern of this research and is controlled by the software interface. A new approach to recognize the most frequently used Chinese characters (about 5,400) is proposed. This new approach includes character representation, primitive stroke extraction and fuzzy similarity measure from an input to the mask in the database. As a result, several goals are to be achieved and they are summarized as follows.

(1) To reduce the size of masks as well as to reduce the size of the database on the secondary storage.

(2) To tolerate variations of handprinted Chinese characters by different writers.

(3) To decrease the training time of the template database.

(4) To improve or at least maintain the recognition rate.

1.7 Outline of the Dissertation

This dissertation is divided into six chapters. Chapter 2 is a review of character recognition technologies. Chapter 3 presents the new feature representation scheme for a stroke based character set -- Chinese. Chapter 4 introduces the fuzzy similarity measure between an input and a template. Chapter 5 presents implementation and

evaluation of the system performance. The last chapter summarizes the work and future study.

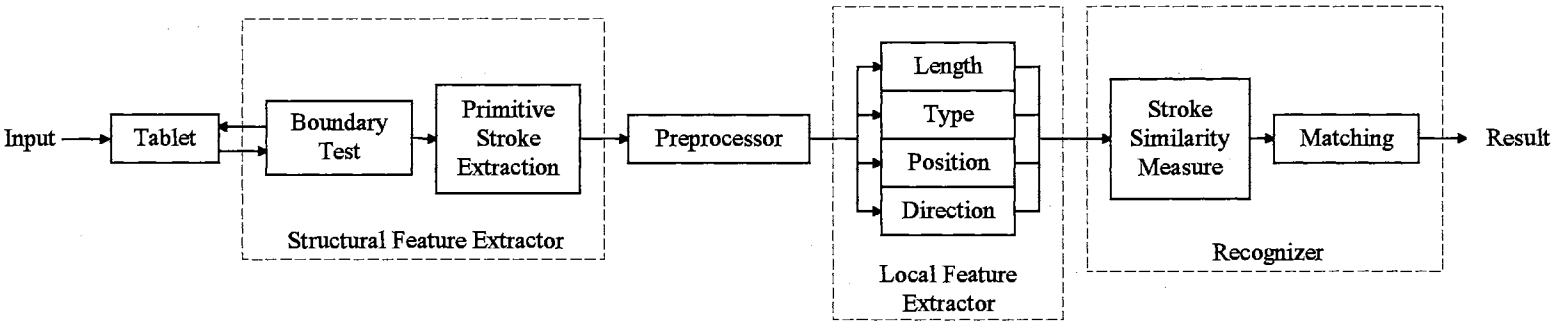


Figure 1.2: Diagram of the proposed on-line handprinted Chinese character recognition system.

CHAPTER II

LITERATURE REVIEW

2.1 Introduction

With the volume of articles, the review of this chapter mostly concerns the recognition of Chinese characters. This review summarizes the historical advancement of character recognition technologies. It is further organized as four groups: recognition of alphanumeric characters, recognition of printed Chinese characters, recognition of handprinted Chinese characters, and miscellaneous. Research publications of the first group are randomly selected to reflect their merits in the area of character recognition. The rest of the groups are mainly devoted to the recognition of Chinese characters. The second group reviews some techniques applied to fixed-font and multi-font printed Chinese characters. The third group reviews handprinted Chinese characters. The last group is a special collection of recognition processes that apply either fuzzy logic or neural networks.

2.2 Recognition of Alphanumeric Characters

Many successful applications for the recognition of alphanumeric characters have been investigated and implemented for years; such as the recognition of postal addresses [42,43], on-line run-on character recognition [47], document processing [61], and

signature verification [48]. Due to the importance of alphanumeric characters, this section is devoted to showing the current research trends rather than covering the historical advancement in this area.

Gader et al. [44] applied a pipeline strategy for handwritten numeral recognition. This pipeline strategy combines a two-stage template-based technique and a model-based technique. By the training on rejecting from the preceding stage, the system can yield higher reliability without a corresponding loss in recognition. Wang and Wang [46] proposed an approach for unconstrained handwritten numerals. Their approach derives stroke changing sequence and primitives from crossing number sequence and left-right profiles. Then a decision tree is established on the basis of these features for recognition.

Smagt [45] conducted experiments for the recognition of numerals based on the methodologies of feed-forward networks, Hopfield networks and competitive learning networks. Performance results from various network topologies were compared to the *k*-nearest-neighbor under the same condition. Another example with modified sparse distributed memory network model was conducted by Fan and Wang [60].

Since fuzzy logic caught the attention of the public, many practical research topics have been re-evaluated by using this methodology. Siy and Chen [62] interpreted numerals by membership functions. Kickert and Koppelaar [63] extended the syntactic recognition scheme by the idea of fuzzy sets to replace the probabilistic approach to recognizing capitals. Gary and Joe [109] proposed to represent structural relations between primitives by fuzzy attributed graphs. The similarity of two fuzzy attributed graphs of a known character and an unknown character is then estimated.

An interesting piece of research by Boccignone et al. [41] was to recover part of the lost script dynamics from static handwritten numerals. They reconstructed the order of writing from the possible conjunctions where a pen-up (pen-down) occurred; thus allowing a more effective description of characters for recognition purposes. Nadal and Suen [40] intended to differentiate confusing numerals by applying more human input. Clearly, these techniques have shown some of the future trends in this field.

Wang and Gupta [49] proposed an improved extended octal code that offers flexibility of size, orientation, and variation. Another structural approach from Nouboud and Plamondon [50] employs the chain code representation to encode each character. Thus, classification is a simple task of string comparisons.

2.3 Recognition of Printed Chinese Characters

The earliest report on machine printed Chinese character recognition was from Casey and Nagy [17] in 1966. Since then, many researchers have proposed the recognition of fixed-font [51,52] and multi-font [53-59,115] for printed Chinese characters. These approaches include correlation, transformation, global and local analysis, shape matching, structural analysis, projection profiles, etc. In general, it is more difficult to recognize multi-font than to recognize fixed-fonts of printed characters. However, it is still far more easy than recognizing handwritten characters. This is because differences between two fonts are predictable while differences between two writers vary from time to time. In the next two sections, the review of fixed-font and multi-font for printed Chinese characters are presented.

2.3.1 Fixed-Font

Nakata et al. [51] employed the correlation of frequency components from two projection profiles of a character. These features are invariant to the change of position of a pattern. Moreover, the dimensionality of the information is greatly reduced by the use of main components. Wang and Shiau [52] proposed the use of three transformations -- Fourier, Hadamard, and Rapid. They also proposed a three-stage search structure with a dictionary to optimize the search.

2.3.2 Multi-Font

The methods of recognizing multi-font printed Chinese characters fall into two groups; identifying the font before recognition and direct recognition. However, we humans do not require the font information to recognize a character. Therefore, direct matching of characters is a more general approach than the other.

Tsukumo and Asai [53] proposed three improved loci features; namely, four direction feature, four outer periphery feature, and four local corner feature.

Chen et al. [54] proposed to encode a printed character in terms of relative position and relative direction. To solve ambiguities among different fonts, a stroke merging method was employed to extract strokes more steadily.

For years, the statistical approach has been one of the main streams for pattern recognition. Jeng et al. [55] utilized a discrete-state Markov process to solve the input problem of Chinese characters. Another type of research based on the stability of feature points (nodes and end points) was proposed by Zhang et al. [56]. First, they measured

the complexity of feature points, then calculated the similarity on the dispersion of the selected font. Unfortunately, the complexity measure of feature points is unstable in some cases, particularly when the connections of line segments are irregular. Huang and Huang [57] proposed regression analysis to obtain either the value of slope angle or a dispersion code. They also preclassified a character by peripheral blocks and achieved an overall rejection rate of 1.25%.

Neural networks applied to character recognition have achieved a certain amount of success. Wu and Tsai [58] conducted an experiment on decision-tree approach by neocognitrons. Jeng et al. [115] evaluated the applicability of clustering and classification algorithms based on *k*-means clustering, neural net classifier, and a Hidden Markov matching scheme. They reached a 98% recognition rate on average for inside test, and 91% for outside tests. Another study of neural networks applied to printed Chinese was proposed by Zhang et al. [59]. They showed the effectiveness of associative memory networks by carefully selecting Hadamard vectors to represent the inner codes.

2.4 Recognition of Handprinted Chinese Characters

Research on the recognition of handwritten Chinese characters was previously done mostly by Japanese researchers. A survey based on evaluations for the recognition of Japanese Kanji characters can be found in [7]. Many researchers point out that the future research direction should be based on a combination of both topological feature analysis [65-69] and global feature analysis [70-73,29]. This is due to the consideration of the stability of features and ease of mask making mentioned in chapter I.

From the synthetic point of view, each Chinese character is simply a combination of strokes. A stroke is considered as consecutive short line segments from pen-down to pen-up. An intuitive approach of the recognition is stroke analysis. This analysis is further classified according to the complexity of line segments; that is, approaches are based on short line segments [74-81], strokes [82-96], and radicals (composition of strokes) [17,97-100]. Cheng and Hsu [101] proposed three stroke extraction methods based on pen movement, feature point, and Hough transformation for a digitized image.

From the matching point of view, methodologies employed for recognition vary from system to system. To name a few, they are attributed grammar [102,103], attributed graph [69,80], decision rule [83], dynamic programming [29,65,68,71,76,81,86,96], fuzzy logic [104-108], knowledge-based [84], mutually best matching [90], neural networks [110-114], relaxation [67,74,75,78,84,93,94], segment matching [70,72,73], transformation invariant [100] and tree search [85,88,89,92,95,97].

On the basis of the above observations, our review focuses on two major parts: description ability of features (global feature or topological feature) and complexity of line segments (short line segment, stroke or radical). Different matching methods are discussed in each section.

2.4.1 Description Ability of Features

2.4.1.1 Methods Based on Global Features Global features are easy to implement and are unaffected by minor local changes. However, these features are very sensitive to style variations. Yamashita et al. [70] proposed extracting feature vectors based on the distribution of strokes, and employed segment matching with average

vectors in a dictionary. Chen et al. [71] employed a Hough transformation to map Chinese characters from the spatial domain into parametric space; then dynamic programming was applied to optimize the matching process. Some researchers [29,72,73] extracted stroke distribution features such as crossing counts, contour line length and profiles. They then applied different matching methods for discrimination.

2.4.1.2 Methods Based on Topological Features Topological features tolerate high degrees of distortion and style variations, but the complexity of the extraction process complicates the mask making. Ikeda et al. [65] and Jeng [68] both employed dynamic programming techniques to optimize matching probability. Ikeda et al. utilized stroke vector sequences and position vector sequences to represent the skeleton and local characteristics, while Jeng applied accumulated stroke features to reflect the topological properties of a character.

Liu and Kasvand [66] extracted peripheral features (end point, corner, hook, and intersection) to reflect the local property of a word. Another similar set of local features (hook, end point, T-shape, cross, and corner) was suggested by Xie and Suk [67]. They then applied a relaxation scheme for matching based on the above features. A paper from Chen and Lieh [69] employs relaxation both on the learning stage to synthesize different attributed graphs and on the recognition stage to match similar graphs.

2.4.2 Complexity of Line Segments

2.4.2.1 Methods Based on Short Line Segments Every Chinese character is a composition of short line segments. The complexity of matching is high since every

stroke is decomposed into short line segments. This may cause difficulties or prolong the matching process.

Cheung and Leung [77] proposed a chain-code transformation to represent line segments for Chinese characters. This method is similar to the Hough transformation but has an advantage in that background noise is eliminated. Lu et al. [80] used a hierarchical attributed graph representation; thus the recognition process becomes a simple task of graph matching. This approach can tolerate great variations among different writing variations.

Relaxation matching was first proposed by Yamamoto and Rosenfeld [74] for possible matching among polygonal approximation of characters. Several other publications [75,78] also employ this technique to reduce local ambiguities. Owing to its property of elastic matching, it can tolerate considerable degrees of distortion. Unfortunately, the performance of this technique is highly dependent upon the complexity of a character.

Compared to relaxation matching, algorithms of dynamic programming (DP) take less computation time. Dynamic programming is frequently adopted for string matching of an input pattern to reference patterns. Yamada [76] applied DP techniques hierarchically for the matching of both intra-contour and inter-contour. Lee and Chen [81] also employed a DP technique based on the accumulated chain codes for line approximation. To reduce the conventional complexity of DP algorithms, Tsay [79] proposed a model-guided attributed string matching by split and merge. This has proven to be very useful.

2.4.2.2 Methods Based on Strokes Stroke based algorithms are the most popular approach in the recent studies. Most elaborate algorithms use primitive strokes as the main features. However, if a matching algorithm is strongly based on the number of strokes, it may reduce the matching results unless a merge operation is applied.

Wang et al. [93] extracted features based on a full stroke while Cheng et al. [94] sought to obtain some local information from a sub-stroke. They both applied relaxation for elastic matching. Since features are stroke based rather than line segment based, the complexity of relaxation is greatly reduced. A modification of the relaxation matching was proposed by Leung et al. [84] on a stroke based analysis. The idea is to incorporate human knowledge about Chinese characters to remove inconsistencies; thus, speeding up the process. Yet, the training of the knowledge system is time consuming and requires an experienced supervisor.

Lin et al. [86] extracted primitive strokes based on a finite state mechanism. DP was applied for the matching among primitive strokes. In the second stage, the positions of strokes were considered to improve the recognition rate. Lin et al. [96] built a deviation-expansion model that contains the hypothetical knowledge for reference patterns. A matching graph based on DP scheme was employed to optimize the matching distance between an input and reference patterns.

Chou and Tsai [90,91] advocated an iteration matching scheme based on mutually best matching and at most one to one mapping. In their experiment, a match network was constructed and dynamically optimized to show the matching result between an input and template strokes.

Tree search algorithms for matching between an input and masks are very flexible for small sized character sets. A large character set as Chinese may suffer from the problem of maintenance. Therefore, tree search algorithms are mostly used in limited application such as candidate selection [95], model-guided matching [88,89], or knowledge based searching [92].

2.4.2.3 Methods Based on Radicals The problem confronted by radical based algorithms is the difficulty in correctly extracting radicals. According to the properties of Chinese characters, there are 214 distinct radicals. Some are left-right separable as 木 in 林; some are top-down separable as 木 in 宋; some may exist in different forms as 水 in 河 and 汞. These facts complicate the design of an extraction scheme.

Yhap and Greanias [97] extracted a reduced radical set of 72 alphabetic elements. During the extraction stage, they employed a tree-structured algorithm for the classification of multi-segmented strokes. Cheng and Hsu [98,99] proposed to extract radicals based on background thinning. This method consists of three steps: dividing by projection, dividing by background thinning and dividing by window extraction. Some results have been achieved, but special characters are not considered and errors still exist at the last stage. Liao and Huang [100] proposed a transformation invariant matching algorithm that is invulnerable to the defects of thinning algorithms with trade-off of more computation time.

2.5 Miscellaneous

2.5.1 Fuzzy Logic

Recognition mechanisms based on probability density functions usually have to estimate from a training set. In a practical application, a training set is often a subjective matter. Therefore, it may not be practical to deal with the style variations by probability approach. Fuzzy membership functions, on the contrary, provide the degree of membership of a pattern in a fuzzy set without any assuming function. This concept was first proposed by Zadeh in 1965 [21]. Since then, much attention has been caught by the public.

Cheng et al. [104,105] proposed to denote location measure and position measure of a stroke by the use of fuzzy membership functions. Then, a matching problem is reduced to a simple assignment problem. Chan and Cheung [106] extended the idea of attributed graph to Fuzzy-Attribute Graph (FAG) by making attributes fuzzy; thus equality of attributes is no longer appropriate for similarity measure. They then suggested a new measure for matching two FAGs. Another piece of research from Cheng and Tseng [108] concentrates on the extraction of invariant features for map recognition. They extracted six features that include fuzzy weighted ring data. In the matching stage, a fuzzy membership matching for similarity measure was applied.

2.5.2 Neural networks

A promising technology of pattern recognition is to apply neural networks. In a recent study, neural networks have proven to be very useful for many applications.

Advantages of this technology for character recognition are automatic training, robust performance, potential for parallelization, and reduced storage requirements. Whether or not the emulating of human behavior is achieved; the impact from neural networks is tremendous. A survey from Hildebrandt and Liu [4] has paid special attention to recent connectionist paradigms.

A Hopfield network-based content addressable associative memory was proposed by Yong [110] to perform the task of recognition. The author emphasized emulating biological function -- at times one thing may remind us of several others in our brains. To do so, a fast network was used for rough classification and some slow networks for detail matching.

A neural network model based on the neocognitron was introduced by Su and Tsai [111] to solve large scale Chinese characters. They designed a classification tree by the use of stroke junction features. In addition, each node in the tree is a four-stage neocognitron and recognizes a few features to discriminate a small set of Chinese words. They also proposed a new feature extraction scheme called distributed feature extraction, to extract correct corner features. Another study by Hildebrandt [113] tried to improve the discrimination and generalization of the neocognitron model. The author proposed to identify Kanji in five stages; i.e., edge pixels, oriented edge segments, oriented stroke segments, feature points, and Kanji characters. A better rejection rate has been reported; this has demonstrates that the generalization capability of the new model exceeds the original model. However, the discrimination capability suffers somewhat.

The paper from Lee et al. [112] tried to improve the learning speed and to reduce the architectural complexity. They applied a single layer perceptron classifier with no

hidden layer. A backpropagation learning algorithm was employed to iteratively adjust the weights of the perceptron during the learning stage. Recognition rates of 99% for multi-font and 91% for handprinted Chinese characters are reported. Another approach based on the backpropagation neural network was described by Tseng and Huang [114] to obtain the correct Mandarin phonetic transcription from the stroke sequence of a Chinese character. Their experiment successfully demonstrates the recognition of a subset of handprinted Chinese characters. For a large set where learning time increases fast, the modularity and scaling should be considered to improve the performance.

2.6 Summary

We have reviewed various recognition systems based on the extracted features and employed methodologies. Some restrictions to the systems are ignored during the review unless they were specifically mentioned in the articles. In general, most of the systems have removed one or partial constraints such as stroke order, stroke number, square writing, etc. Some algorithms have advantages over the others under special considerations, such as ease of implementation, possibility of extension, and storage requirement. Nevertheless, it is difficult to make a clear statement about whether one system is superior to another; this is partially because of the differences among environments and partially because of the lack of objective criteria. There is no doubt that an integrated, fast, accurate, and highly tolerant system is needed.

CHAPTER III

FEATURE SELECTION AND REPRESENTATION

3.1 Introduction

The central issue of pattern recognition is to extract distinct characteristics from an input and match them with those stored in the database. It is not always possible to extract distinct features, and yet it is crucial for every recognition system to discriminate one class from the others. Therefore, it is very important to know the properties of a feature. The basic criteria for the selection of features comply with the following rules. They must be distinctive; they must not be sensitive to variation in shape; they must be typical and present in many characters; they must be easy enough to detect and to extract. From these points of view, a review of features and comparisons of recognition techniques are very helpful for the design and implementation of a character recognition system.

3.2 Feature Detection

According to Suen [15], from the analysis point of view, there are two main streams of feature analysis: global and structural. Each form of analysis consists of three methodologies. For global analysis, a methodology for the distribution of points is

employed to provide position information, density, distance between certain points, and crossing counts. A transformation is used as the second methodology to convert a matrix into a series, a vector, or a spectrum. Finally, physical measurement obtains the physical width and height corresponding to the rows and columns of a matrix. For structural analysis, all three methodologies produce a line representation for a character. Each methodology has its own characteristics and merits. Figure 3.1 is a simplified version of feature detection techniques based on the work in [15].

3.3 Comparison of Recognition Techniques

In this section, we are looking at recognition techniques based on different feature properties. Though the central issue of a recognition system is classification, the success of a character recognition system is also built on how useful and distinctive the extracted features are. This also implies that differences between characters are more important than variations between different instances of the same character. Much work has been done on evaluating various recognition techniques based on feature properties [2-4,10]. With a better understanding of feature properties and the emergence of new technologies, the chances of developing improved recognition systems are greater. In the following, recognition techniques are briefly discussed and they are grouped by "global feature analysis" and "structural feature analysis". The details given below are mainly from the results presented in the research articles [2,4,10].

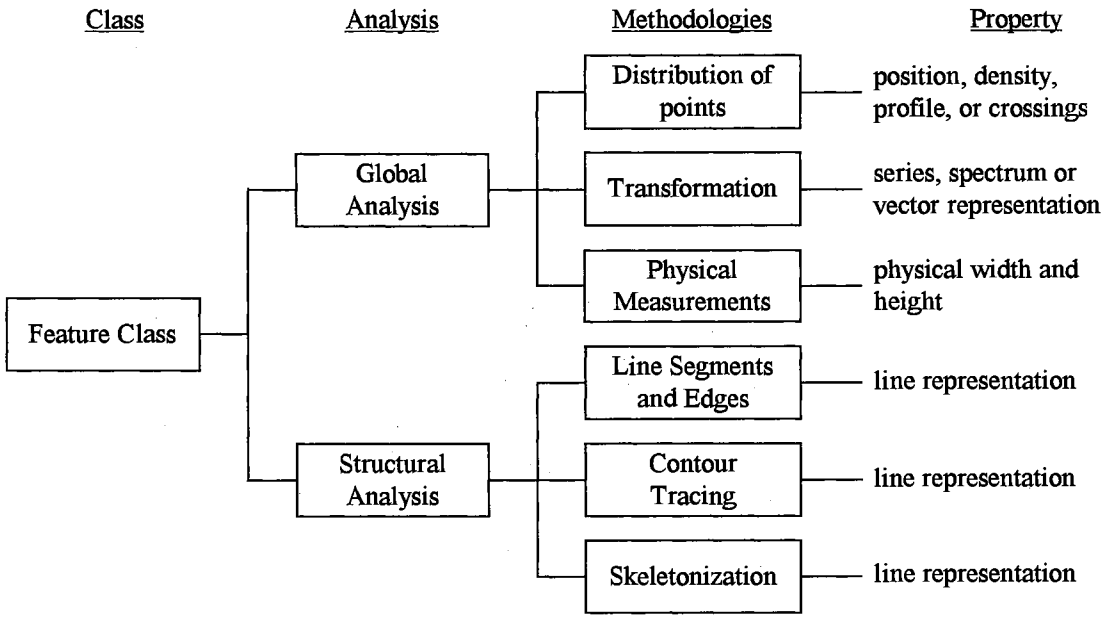


Figure 3.1: Hierarchy of feature detections.

3.3.1 Global Feature Analysis

This category contains techniques that extract global information from the input image. These methods are different from local methods in that an image is considered as a whole, rather than as limited regions. Therefore, features are extracted from every point inside a surrounding frame of a character.

Two major techniques for matching are "template matching and correlations" and "transformations and series expansions". The first technique usually results in high dimensionality of feature vectors. The second approach is to reduce the dimensionality of the first approach and to extract features that are invariant to global translation or rotation.

The implementations of extraction and mask making of global features are easy, and these processes are unaffected by minor local changes. However, dependence on position alignment and high sensitivity to distortion of style variations are the major drawbacks.

3.3.2 Structural Feature Analysis

This category of features represents the structural information of the input. They contain not only global but also local properties. These features tolerate high degrees of distortion and style variations while providing a certain amount of tolerance to rotation and translation. Owning to the afore-mentioned advantages, this category is the most popular feature method among researchers. Unfortunately, the complexity of the extraction process complicates mask making.

It is obvious that all methods suffer different disadvantages; yet they all have their merits from various points of view.

3.4 Normalization

As mentioned earlier, data representing a character may be normalized in terms of size, orientation, or position. It is not required for every recognition system to perform normalization for every aspect, yet it is common to most character recognition systems to perform specific normalization for special needs. In our experiment, a character is adjusted to the size of N by N (N = 64 pixels) before it can be used as an input or stored as a template in the database.

Recall that a Chinese character is a collection of strokes; a stroke is a collection of points. By normalization, every point associated with each stroke of a character is reallocated within a specific size. To transfer every point associated with a stroke, the smallest square that circumscribes the entire character is obtained during the writing. Assume A denotes the smallest square and W_A denotes the width of A. Normalization of start point $S(x_s, y_s)$ to $S'(x'_s, y'_s)$ of a stroke is given by:

$$x'_s = \frac{x_s}{W_A} \times N \quad \text{and} \quad y'_s = \frac{y_s}{W_A} \times N.$$

These equations are applied to every feature point that constitutes a character. By definition, a feature point is considered as either a start point S, an end point E, or any point between S and E of a stroke. For example, an end point $E(x_e, y_e)$ is normalized to $E'(x'_e, y'_e)$. After all feature points are properly adjusted, the resulting points are used to depict a character.

3.5 Stroke Representation

There exist many versions of the definition of a stroke. Due to the property of on-line recognition, it is more appealing to define a stroke as the composition of line segments from pen-down to pen-up; i.e., from a start point to an end point. Moreover, a stroke is not required to be a straight line.

Every Chinese character is a combination of strokes. From the syntactic point of view, it is straight forward to analyze individual strokes rather than the entire character. This approach has been adopted by many researchers and the progress is very encouraging.

Conventional schemes of stroke representation require detailed description of the selected features in order to portrait the characteristics of a stroke. This usually ends up creating a large database file to store the entire Chinese character set. To overcome this problem, we attempt to find a new representation scheme that will reduce the size of the database.

In our approach, local and structural features are provided to represent a stroke. Since local features are sensitive to style variations, structure features are obtained to compensate for these effects. Equally, due to the difficulty of extracting structural features, local features are provided to transfer stable stroke information. Another major distinction between both features is that structural features can be extracted during writing, while local features must wait until the normalization of an entire character. In the following, we intend to examine the following issues for both local features and structural features.

(1) What features do we want to extract?

(2) How to extract and what are their definitions?

(3) When do these features appear to be unstable?

3.5.1 Local Features

Features from this category are considered as limited regions from an input. These features reflect the local information of an image. Considering stroke based Chinese characters, there is specific local information embedded within each stroke. This information includes length, type, position, direction, curvature, crossing, etc. Individually, none of them is sufficient to represent a stroke, but a combination of some

of these local features can provide general information about a stroke. To show the idea, we propose to extract length, type, position, and direction of a stroke. Definitions and stabilities of these features are followed immediately.

3.5.1.1 Basic Definition A set of local features is defined for stroke representation. These features are the length, type, position and direction of a stroke. For the ith stroke of a character, they are denoted as L_i, T_i, P_i, and D_i accordingly. They are abstractions of the physical properties of the selected stroke. Each feature is extracted after the normalization of a character for consistency consideration. They are all introduced in both linguistic and quantitative representations. The purpose of the linguistic description is to provide the conceptual characteristic of a stroke. Visual perception usually contains fuzzy interpretation. Very often, there is no need to specify the exact information of each stroke since the relations between strokes usually provide enough information about a character. Quantitatively, classification of each feature takes the power of inexact interpretation and minimizes the size required to describe a stroke. With the combination of all these features, we have the advantage of enough information for stroke analysis, while still being able to perform structural analysis if it is necessary.

(A) Length of Stroke

The length of a stroke is defined as one of the four linguistic primitives; i.e., very long (VL), long (L), short (S), and very short (VS). Let L_i denote the length of the ith stroke of a character. L_i is defined as the function $f_L(l_i)$ for $0 \le l_i \le \sqrt{2}\mathrm{N}$ by

$$f_L(l_i)=\begin{cases}0 & \text{if } 0\le l_i < \text{N}/4 \text{ for VS}\\ 1 & \text{if N}/4\le l_i < \text{N}/2 \text{ for S}\\ 2 & \text{if N}/2\le l_i < 3\text{N}/4 \text{ for L}\\ 3 & \text{if } 3\text{N}/4\le l_i \le \sqrt{2}\text{N for VL}\end{cases}, \tag{3.1}$$

where l_i is the distance from the normalized start point $S'(x'_s, y'_s)$ to the normalized end point $E'(x'_e, y'_e)$ of the ith stroke. Thus, l_i is given by

$$l_i=\sqrt{(x'_s-x'_e)^2+(y'_s-y'_e)^2}\ . \tag{3.2}$$

(B) Type of Stroke

The type of a stroke is defined as one of the four linguistic primitives; i.e., vertical (V), upper-right to lower-left (RL), upper-left to lower-right (LR), and horizontal (H). The type of the ith stroke of a character, denoted as T_i, is defined as the function $f_T(\theta_{ti})$ for $0\le\theta_{ti}<\pi$ by

$$f_T(\theta_{ti})=\begin{cases}0 & \text{if } 7\pi/8<\theta_{ti}<\pi \text{ or } 0\le\theta_{ti}\le\pi/8 \text{ for H}\\ 1 & \text{if } 5\pi/8<\theta_{ti}\le 7\pi/8 \text{ for LR}\\ 2 & \text{if } \pi/8<\theta_{ti}\le 3\pi/8 \text{ for RL}\\ 3 & \text{if } 3\pi/8<\theta_{ti}\le 5\pi/8 \text{ for V}\end{cases}, \tag{3.3}$$

where θ_{ti} is measured counter-clockwise from the non-negative x-axis to the line that connects the normalized start point and the normalized end point of stroke i. Consequently, the equation for θ_{ti} is given as:

$$\theta_{ti}=\begin{cases}0 & \text{if } y'_s = y'_e\\ arc\cot\left[\left(x'_s - x'_e\right)/\left(y'_s - y'_e\right)\right] & \text{otherwise}\end{cases}. \tag{3.4}$$

(C) Position of Stroke

The position of a stroke is defined corresponding to the center of a normalized character (origin). There are four linguistic primitives: very far (VF), far (F), close (C),

and very close (VC). The position of the ith stroke of a character, denoted as P_i, is defined as the function $f_P(d_i)$ for $0 \le d_i \le \sqrt{2}\mathrm{N}/2$ by

$$f_P(d_i) = \begin{cases} 0 & \text{if } 0 \le d_i < \mathrm{N}/8 \text{ for VC} \\ 1 & \text{if } \mathrm{N}/8 \le d_i < \mathrm{N}/4 \text{ for C} \\ 2 & \text{if } \mathrm{N}/4 \le d_i < 3\mathrm{N}/8 \text{ for F} \\ 3 & \text{if } 3\mathrm{N}/8 \le d_i \le \sqrt{2}\mathrm{N}/2 \text{ for VF} \end{cases}, \tag{3.5}$$

where d_i is the distance from the normalized middle point (x'_m, y'_m) of the ith stroke to the origin and

$$x'_m = \frac{x'_s + x'_e}{2} \quad \text{and} \quad y'_m = \frac{y'_s + y'_e}{2}.$$

Therefore, d_i of the ith stroke is given by

$$d_i = \sqrt{x'^{\,2}_m + y'^{\,2}_m}\ . \tag{3.6}$$

(D) Direction of Stroke

There are four different primitive directions; i.e., East (E), South (S), West (W) and North (N). Let D_i denote the abstraction of direction of the ith stroke. D_i is defined as the function $f_D(\theta_{di})$ for $-\pi < \theta_{di} \le \pi$ by

$$f_D(\theta_{di}) = \begin{cases} 0 & \text{if } 3\pi/4 < \theta_{di} \le \pi \text{ or } -\pi < \theta_{di} \le -3\pi/4 \text{ for W} \\ 1 & \text{if } \pi/4 < \theta_{di} \le 3\pi/4 \text{ for N} \\ 2 & \text{if } -3\pi/4 < \theta_{di} \le -\pi/4 \text{ for S} \\ 3 & \text{if } -\pi/4 < \theta_{di} \le \pi/4 \text{ for E} \end{cases}, \tag{3.7}$$

where θ_{di} is the relative direction from a normalized middle point to the origin. Thus, we get

$$\theta_{di} = \begin{cases} 0 & \text{if } y'_m = 0 \\ arc\cot(x'_m / y'_m) & \text{if } y'_m > 0 \\ -arc\cot(x'_m / y'_m) & \text{if } y'_m < 0 \end{cases} . \qquad (3.8)$$

Specification of each category of local features is presented in Table 3.1 as well as the corresponding graphical classification of each local feature in Figure 3.2. With binary number system, VS, H, VC and W are coded as 00; S, LR, C and N are coded as 01; L, RL, F and S are coded as 10; VL, V, VF, E are coded as 11. Consequently, the representation of a stroke by local features is efficiently encoded into one physical byte. The bit allocation in Table 3.2 shows the mapping of local features to a physical byte.

TABLE 3.1

SPECIFICATION OF LOCAL FEATURES
FOR THE *i*th STROKE OF ANY CHARACTER

(a) Length

Linguistic Description	very short (VS)	short (S)	long (L)	very long (VL)
Quantitative Description	$0 \le l_i < N/4$	$N/4 \le l_i < N/2$	$N/2 \le l_i < 3N/4$	$3N/4 \le l_i \le \sqrt{2}N$
$L_i = f_L(l_i)$	0	1	2	3

(b) Type

Linguistic Description	horizontal (H)	upper-left to lower-right (LR)	upper-right to lower-left (RL)	vertical (V)
Quantitative Description	$0 \le \theta_{ti} \le \pi/8$ or $7\pi/8 < \theta_{ti} < \pi$	$5\pi/8 < \theta_{ti} \le 7\pi/8$	$\pi/8 < \theta_{ti} \le 3\pi/8$	$3\pi/8 < \theta_{ti} \le 5\pi/8$
$T_i = f_T(\theta_{ti})$	0	1	2	3

(c) Position

Linguistic Description	very close (VC)	close (C)	far (F)	very far (VF)
Quantitative Description	$0 \le d_i < N/8$	$N/8 \le d_i < N/4$	$N/4 \le d_i < 3N/8$	$3N/8 \le d_i \le \sqrt{2}N/2$
$P_i = f_P(d_i)$	0	1	2	3

(d) Direction

Linguistic Description	West (W)	North (N)	South (S)	East (E)
Quantitative Description	$-\pi < \theta_{di} \le -3\pi/4$ or $3/4\pi < \theta_{di} \le \pi$	$\pi/4 < \theta_{di} \le 3/4\pi$	$-3\pi/4 < \theta_{di} \le -\pi/4$	$-\pi/4 < \theta_{di} \le \pi/4$
$D_i = f_D(\theta_{di})$	0	1	2	3

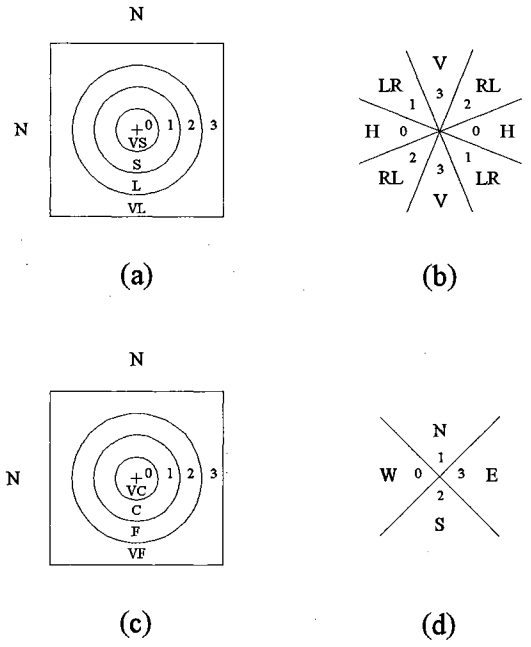


Figure 3.2: Graphical classification of local features. (a) Measure of stroke length where the cross represents the middle point of a stroke; (b) measure of stroke type where the crossing represents the middle point of a stroke; (c) measure of stroke position where the cross represents the center of a character; (d) measure of stroke direction where the crossing represents the center of a character.

TABLE 3.2

STROKE REPRESENTATION BY LOCAL FEATURES

Bit Position from the Most Significant Bit	Contents
0 - 1	stroke length
2 - 3	stroke type
4 - 5	stroke position
6 - 7	stroke direction

3.5.1.2 Feature Stability The representation of a stroke has tremendously reduced the size of the masks in the database. Now, the question is "Are they always reliable?" Unfortunately, the answer is no. Imagining a stroke written in a square block on a sheet of paper, which would be the most unreliable among length, type, position, and direction? All four features are strongly dependent on where and how they are written. Nevertheless, it appears that the existence of some features reflects the stability of the others. To clarify this, how features were obtained becomes very important. Length and type are determined by the start and the end point of a stroke, while position and direction are measured from the middle point of a stroke. Therefore, we divide features into two groups; i.e., length versus type and position versus direction.

(A) Length versus Type

If the length of a stroke is long enough, the type is reliable because minor changes at the end of the stroke will not affect much of the type classification unless it is close to the boundary. On the contrary, if a stroke is too short, then the type may become unreliable. Very often, the writing of a short stroke may result in a great amount of type deviation at the end point. Consequently, the type of a short stroke may easily fall into other categories and the slope of the stroke may appear to be different from usual.

At times, a short stroke may be ill written (dramatically changed in direction); yet the recognition process by human eyes is not affected. This is because most short strokes are simple and we humans are apt to ignore their direction by picking up the length and position instead. As a result, variations in the type of a short stroke become less important for the recognition process. Figure 3.3 demonstrates some possible

writings of the characters 我 and 永. As indicated by the arrows, short line segments may turn out to be different types; yet all characters in a row are identical.

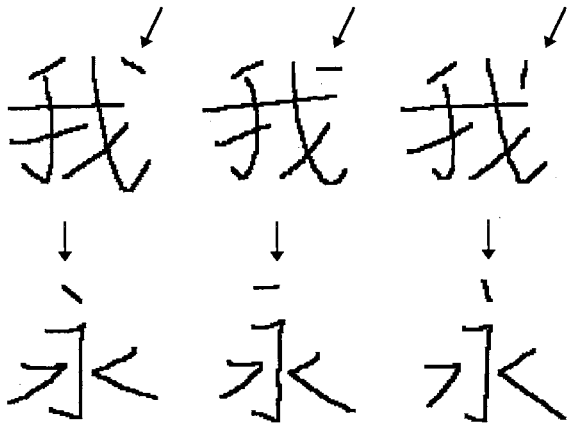


Figure 3.3: Possible writings of the characters 我 and 永. As indicated, types of short strokes are not significant and are usually ignored during recognition.

(B) Position versus Direction

The position of a stroke is another example that may affect the reliability of the direction of a stroke. When a stroke is classified as very close to the center, the direction becomes very ambiguous because the middle point could be in any direction closed to the origin. A little change of the position of the middle point could actually change the direction of a stroke from North to East, West, or even South. In such cases, the direction of a stroke is very unreliable. Whether or not human eyes capture the direction of a stroke is unknown, yet recognition is not affected by minor changes of the stroke position at the center.

If a recognition system does not consider the stability of features, it may degrade system performance. Under this premise, it is required to analyze feature stability to avoid the problem. From the above discussion, we neglect unstable information during the recognition process and will discuss this further in Chapter IV. In summary, the idea is to guide by template (mask) database to avoid unreliable information.

3.5.2 Structural Features

As mentioned earlier, structural approaches for mask making and feature extraction are usually difficult. Nevertheless, structural information is valuable for the recognition process. Recall that the definition of a stroke is from pen-down to pen-up; it is possible that a stroke is a composition of straight line segments. In such cases, stroke represented by local features (length, type, position, and direction) becomes insufficient. It is even possible that two characters could be encoded to the same code or at least very closely. Consequently, ambiguity may jeopardize system performance during comparison. To overcome this problem, we propose to extract stroke based structural features -- primitive strokes. A primitive stroke is a collection of short line segments that are written continuously in a standard or conventional way. From the definition of primitive strokes, only limited information is extracted to assist in distinguishing ambiguous characters represented by local features. Figure 3.4 shows some characters that have similar coding results corresponding to local features.

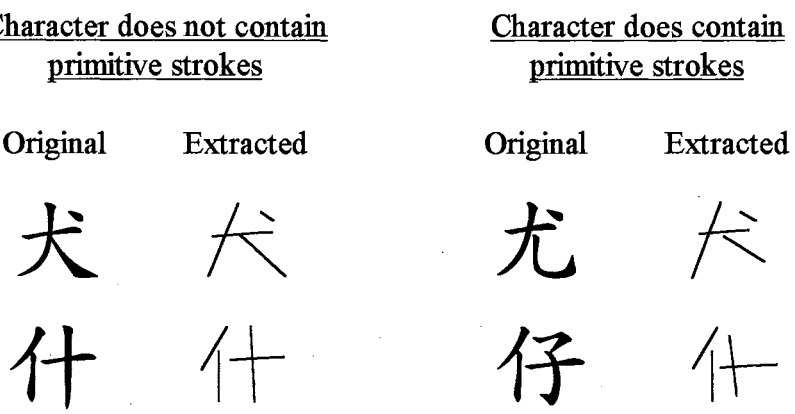


Figure 3.4: Ambiguous characters and their stroke representation by local features.

3.5.2.1 Collection of Primitive Strokes A set of primitive strokes is selected to provide partial structural information of a character. The selection of primitive strokes is done under the consideration that they are written in uniform, standard or conventional ways. This particular property allows us to extract such strokes and denote each of them by a stream of direction codes. There is a total of ten groups of primitive strokes in our experiment. Detailed shapes and direction sequences are presented in Table 3.3.

Using direction sequences to identify a particular primitive stroke has been applied by many researchers [35,67,85,86]. Nevertheless, the selection and grouping of primitive strokes, number of directions, and the algorithm to extract the direction sequence vary from system to system. We adopt an 8-neighborhoods direction code to represent the direction of a line segment; the definition of the codes is shown in Figure 3.5.

TABLE 3.3

PRIMITIVE STROKES AND THEIR DIRECTION SEQUENCES

Type	Shape	Possible Direction Sequences	Examples
1	㇀,㇇	01|020|0(1|2)(6|7)	氣,九
2	丿,㇒,㇔,㇏,㇕,㇆	(0|1|7)(2|3)|02(4|5)	夕,疑,欠,力,口,門
3	乙,㇈	0(2020|3(0(*e*|30)|130))	乙,近
4	𠂇,㇋,了	0(2(0(2|3)|13)|3(03|1(2|3)|2(*e*|3)))	建,乃,了
5	𠃌,㇂,㇎,㇄,㇃	(1|2)(0|7)	世,亂,已,心,良
6	𠂉	(2|3)1	女
7	亅	2(4|5)	利
8	㇉	(2|3)(0|1)(2|3)	弓
9	𠂆	30	矣
10	𠂋	32	齊

e - empty string
| - inclusive or

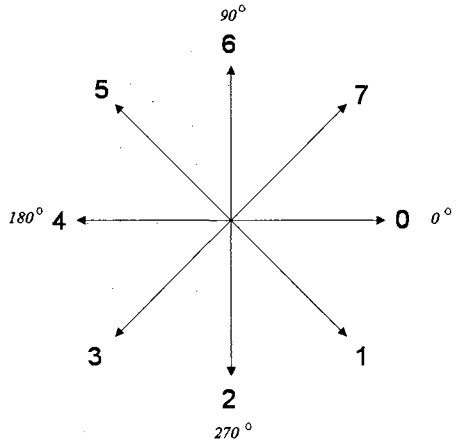


Figure 3.5: The 8-neighborhoods direction code.

Ideally, each primitive stroke should have a particular direction sequence. Unfortunately, this may not always be true, due to great variations among writers and inaccuracy of the normalization process. For example, some strokes may be simplified by different writers; ㄴ may be written as ㄱ, or ㄱ as ㄱ. As a result, the direction codes for ㄴ and ㄱ or ㄱ and ㄱ may appear to be the same. We gather these kinds of strokes and represent them by a single type such as type 1, 2, 4 and 5 in Table 3.3. This approach has an advantage that both writings (normal or simplified) can be classified into a same primitive type; a tradeoff is the possibility of shortening the distance to the other candidates that contain a similar stroke. Occasionally, a stroke may be ill written. As a result, the direction sequence of a primitive stroke may be very different. Therefore, we deliberately create possible direction sequences for types 1 to 8. For example, possible direction sequences for type 7 primitive stroke (丿) are either 24 or 25. By this approach, the system is able to tolerate some variations.

In addition to the start point (S) and end point (E), a corner point C (x_c, y_c) is introduced for the measure of direction sequences of a non-straight line stroke. The direction of a straight line segment is estimated from one feature point to the other feature point of a stroke, where a feature point can be a start point, a corner point, or an end point. The definition of a start point and an end point are the same, while a corner point is defined as a point that connects two other feature points as in Figure 3.6. The construction of a straight line segment may be from S to C, from C to C, or from C to E and no others. From the above definitions, it is straightforward that a critical criterion

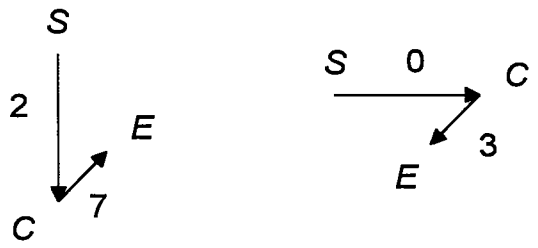


Figure 3.6: Examples of strokes and their corresponding direction codes.

for the extraction of a primitive stroke is the extraction of corner points. In the next section, an algorithm to extract critical corner points during writing is proposed.

3.5.2.2 Primitive Stroke Extraction The fundamental problem of on-line extraction comes from the instability of sampling points from an interface or from the writing itself. A person may write too fast on a slow sampling stylus or write too slowly on a fast sampling stylus. The writing may be ill-written by a person that does not follow the rules of writing. All these facts may result in great variations in sampling points. Consequently, a straight line may turn out to be a zigzag line. This zigzag line may be thought of as a set of successive short line segments as well. The connecting points of these short line segments are therefore considered as noisy points. Noisy points are defined as points that are created unexpectedly and their existence is hazardous to the structure of a stroke. To cope with this problem, the major concern of an on-line stroke extraction scheme is to eliminate these noisy points.

We propose an algorithm to extract primitive strokes in two phases. The first phase is to extract turning points (corner points) from an input stroke. The second phase is to remove false turning points created by a straight line stroke. After both operations

are completed, the left-over corner points are the base for the construction of the direction sequence of a stroke.

The central issue of the first phase is to extract the corner points of a stroke. This implies that all points between the start point and the end point require verification. To reduce the complexity of the operation in the first phase, it is possible to remove noisy and unnecessary points by a simple boundary test during the writing. The basic idea of the boundary test is to remove points that are too close to the previous candidate points. In other words, if the distance of a current point to a candidate point falls under a certain threshold, then the current point is said to be noisy and is qualified for removal. To accomplish the task, a fixed-size rectangle (11x11 pixels) is defined for the test. The selection of this fixed-size rectangle is under the consideration that a resolution of 640x480 is used. In essence, the test reduces the number of points created by a fast sampling interface or by slow writing. Figure 3.7 shows examples of removing noisy sampling points by the boundary test. As demonstrated, the point labeled as 3 is removed after the test.

The extraction of turning points is initiated after the writing of a stroke. The idea is to verify whether a point represents an abrupt protrusion on a stroke. From the geometrical point of view, this property can be estimated by the change of distances from continuous points to $\overline{SE}$, where $\overline{SE}$ is a straight line segment that connects start point S and end point E. In analytical geometry [120], the distance between an arbitrary point (x_c, y_c) and the line $ax + by = c$ is given by the expression

$$\frac{|ax_c + by_c - c|}{\sqrt{a^2 + b^2}},$$

where a, b, and c can be easily determined by the following observation. Given $S(x_s, y_s)$ and $E(x_e, y_e)$, if $x_s = x_e$, then $a = 1$, $b = 0$, and $c = x_s$, since the slope of a vertical line is undefined; otherwise, $a = -m$, $b = 1$, and $c = y_s - mx_s$ where the slope m of the line through S and E is given by

$$m = \frac{y_s - y_e}{x_s - x_e}.$$

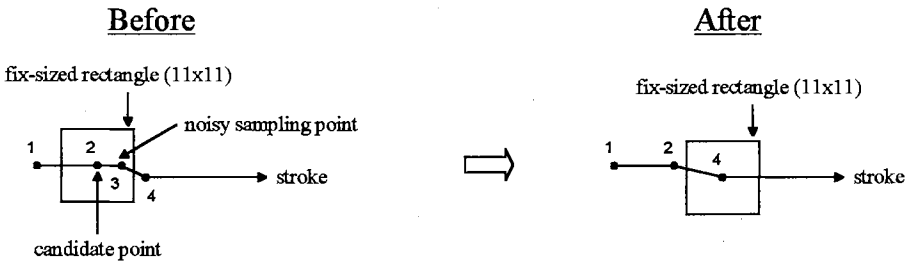


Figure 3.7: The removal of noisy and unnecessary sampling points by the boundary test.

By observing the change of distances from corner points to the line, we can remove unnecessary points from the list that depicts a stroke. The verification of corner points of a stroke repeats until the end point is reached. Finally, the list of candidate turning points (include a start point and an end point) is sent to the second procedure for further processing. Without loss of generality, $\overline{SE}$ is drawn vertically in Figure 3.8. We define four kinds of detectable turning points; other than these cases, a point is qualified for removal from the candidate list. A diagram of the first phase of the primitive stroke

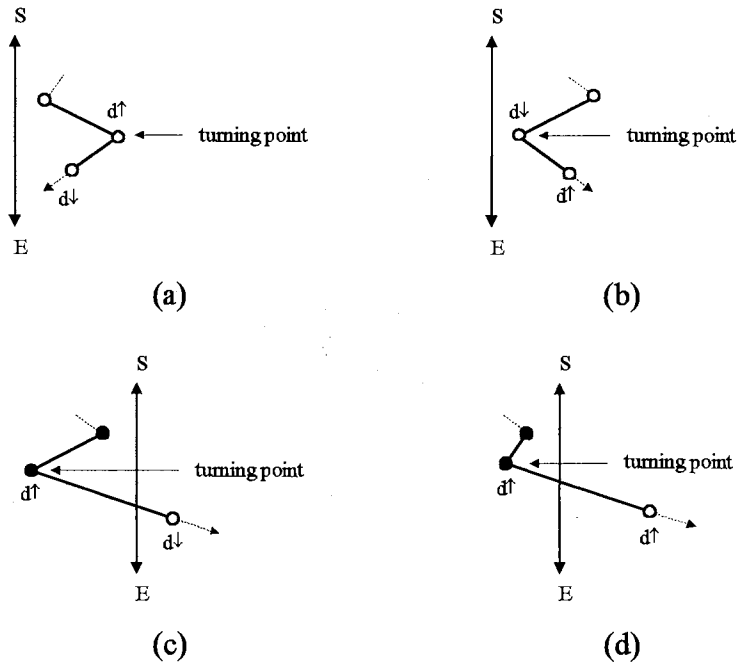


Figure 3.8: Detectable turning points. 'd↑' ('d↓') represents the distance from the current point to $\overline{SE}$ that is longer (shorter) than the distance from the previous point to $\overline{SE}$. '●' denotes a point that is on the left-hand side of $\overline{SE}$ and '○' denotes a point that is on the right-hand side of $\overline{SE}$.

extraction scheme is presented in Figure 3.9. Definitions and functions are given as follows.

Variables:

`a, b, c` - coefficients of line $ax + by = c$ through S and E for measure of distance

`curPt` - current point

`prePt` - previous point

`i` - index to the position of a point in a stroke, where a stroke is defined as a collection of points

`nPt` - total number of points that constructs a stroke

`stk` - current input stroke

Functions:

`CompatiblePt(curPt,prePt)` - checks the compatibility of `curPt` and `prePt`. The compatible criteria include a check of distance from both points to $\overline{SE}$ and a check of relative position from both points to $\overline{SE}$. If the relationship between `curPt` and `prePt` is one of the cases in Figure 3.8, they are not compatible, otherwise they are compatible.

`GetPoint(stk,i)` - gets a point at position `i` from stroke `stk`.

`RemovePt(stk,i-1)` - removes a point at position `i-1` from stroke `stk`.

`SizeOfStroke(stk)` - returns the total number of points that constructs stroke `stk`.

The algorithm in the first phase cannot detect the corner point illustrated in Figure 3.10. This is because the check of distance and the check of position corresponding to $\overline{SE}$ do not provide enough information for identifying such cases. Other information, such as the angle between two line segments, is required to determine such turning points. Nevertheless, it may not be worth the trouble to define another criterion when the problem can be solved easily by the grouping between different strokes. Therefore, end hooks of ㄱ, ㄱ, ㄴ, ㄱ, ㄱ, ㄴ, ㄴ, ㄴ, and ㄴ are usually undetectable as depicted in Figure 3.10.

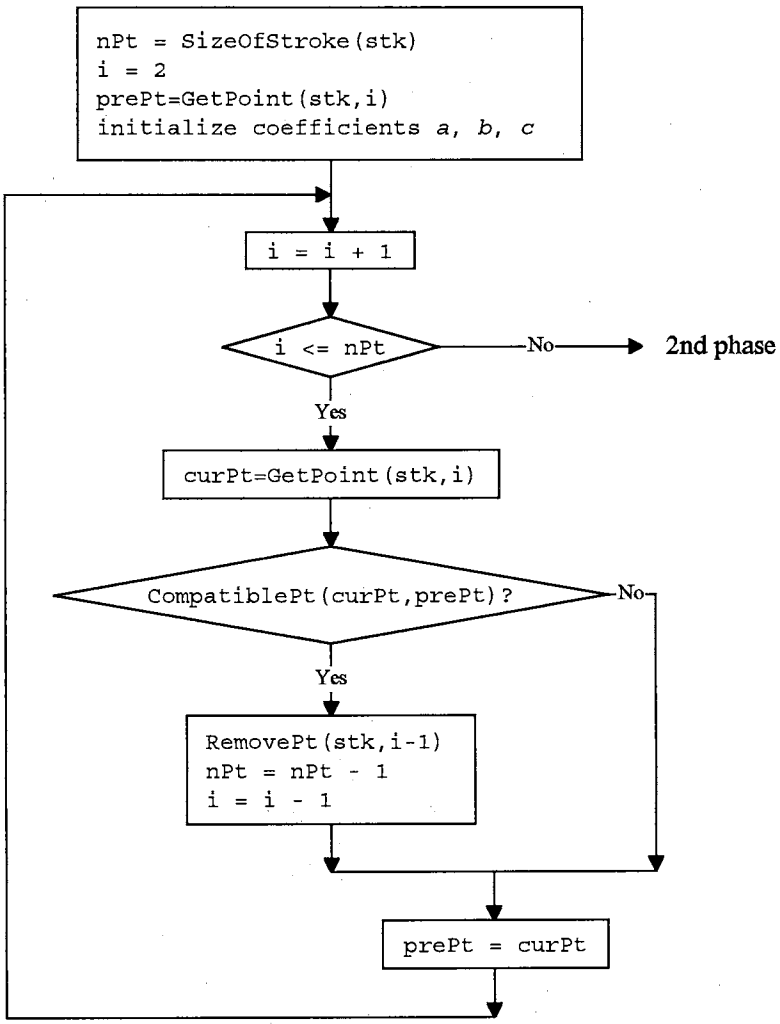


Figure 3.9: Primitive stroke extraction scheme -- first phase.

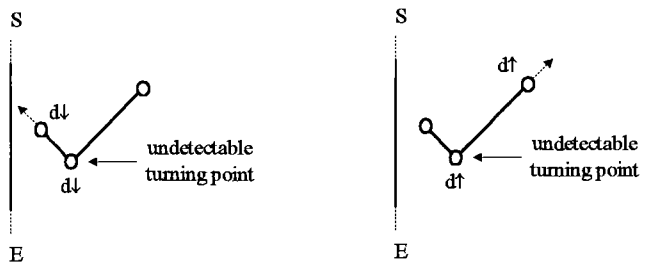


Figure 3.10: Undetectable turning points.

Recall that first phase of the extraction creates a list of candidate turning points from an input stroke. Unfortunately, the algorithm does not take care of strokes that are purely straight line segments (or nearly straight line segments). This is because most of the straight line segments are actually represented as zigzag lines by a set of points. In such cases, some of these points tend to be detected during the first phase, resulting in multiple false turning points. To avoid the translation of these false turning points into direction sequences, we remove these false turning points in the second phase.

The second phase of the primitive stroke extraction scheme has two major objectives: remove false turning points and translate paired stable points into a direction code. The process of the second phase checks each candidate turning point until the list is exhausted. The checks include length and direction tests. The length test is to make sure that the length from one point to the next point is long enough to represent the direction; the direction test is to check whether consecutive directions are the same. To systematize both operations, a stroke is normalized to the size of NxN before processing. In other words, each point is transformed to a new point in the new coordinate system at origin (0,0) such that every x and y satisfy the condition $-N/2 \leq x,y \leq N/2$. This

transformation of points assures consistency of length and direction tests. Nevertheless, it is not immune to minor distortions. The threshold for the length test is selected by N/10; straight line segments that are shorter than this threshold are considered unstable and are qualified for removal. At the end of the entire operation, the resulting direction sequence is used to search the database. A diagram of the second phase for the extraction of primitive strokes is presented in Figure 3.11. Definitions and functions are given as follows.

Variables:

`pDir` - previous direction

`cDir` - current direction

`i` - index to the position of a point in a stroke

`nPt` - total number of points that constructs a stroke

`spt` - current start point

`ept` - current end point

`DirSeq[]` - a stream of direction codes

Functions:

`SizeOfStroke(stk)` - returns the total number of points that constructs stroke `stk`.

`GetPoint(stk,i)` - gets a point at position `i` from stroke `stk`.

`Transformation(spt)` - transforms point `spt` to a new point in the new coordinate system at origin (0,0) such that every x and y satisfy the condition $-N/2 \leq x,y \leq N/2$.

`LengthTooShort(spt,ept)` - checks the length between point `spt` and point `ept`. If the length between these two points is less than N/10, the length is said to be too short.

`CreateDir(spt,ept)` - creates the direction code from point `spt` to point `ept`.

`RemovePt(stk,i-1)` - removes a point at position `i-1` from stroke `stk`.

Although the extraction of primitive strokes can be achieved, there are constraints that must be followed during writing to ensure the performance of the extraction.

(A) The writing speed should not be too slow. Since the tablet is a very sensitive pointing device, it may get noisy points around the candidate feature point. These noisy points may result in the misclassification of primitive strokes.

(B) The writing speed should not be too fast. This is under the consideration that a slow sampling tablet may not be fast enough to capture a corner point of fast writing.

(C) General rules of writing must be followed; a straight line segment should never be written as two short line segments with great difference in angles. If such a case exists, the stroke is considered ill-written and may again cause the misclassification

of the stroke and enlargement of the distance to the correct character during matching. Moreover, the standard writing order of a stroke must be followed.

(D) Last but not least is that a straight line segment should never be written as a closed curve. If the curvature of a stroke exceeds the threshold, a noisy critical point will be generated. This side-effect will jeopardize the classification of the correct primitive stroke.

3.5.3 Summary of Stroke Representation

We have introduced the representation of a stroke in both local and structural analyses. If a stroke is a purely straight line segment, local description is enough to represent it. This is under the assumption that distinct properties of a stroke are provided. On the contrary, if a stroke is not a straight line segment, local description may not be sufficient. In such cases, structural description is required to distinguish one stroke from another. Therefore, the representation of a stroke consists of two types: a local description or a local description plus a structural description.

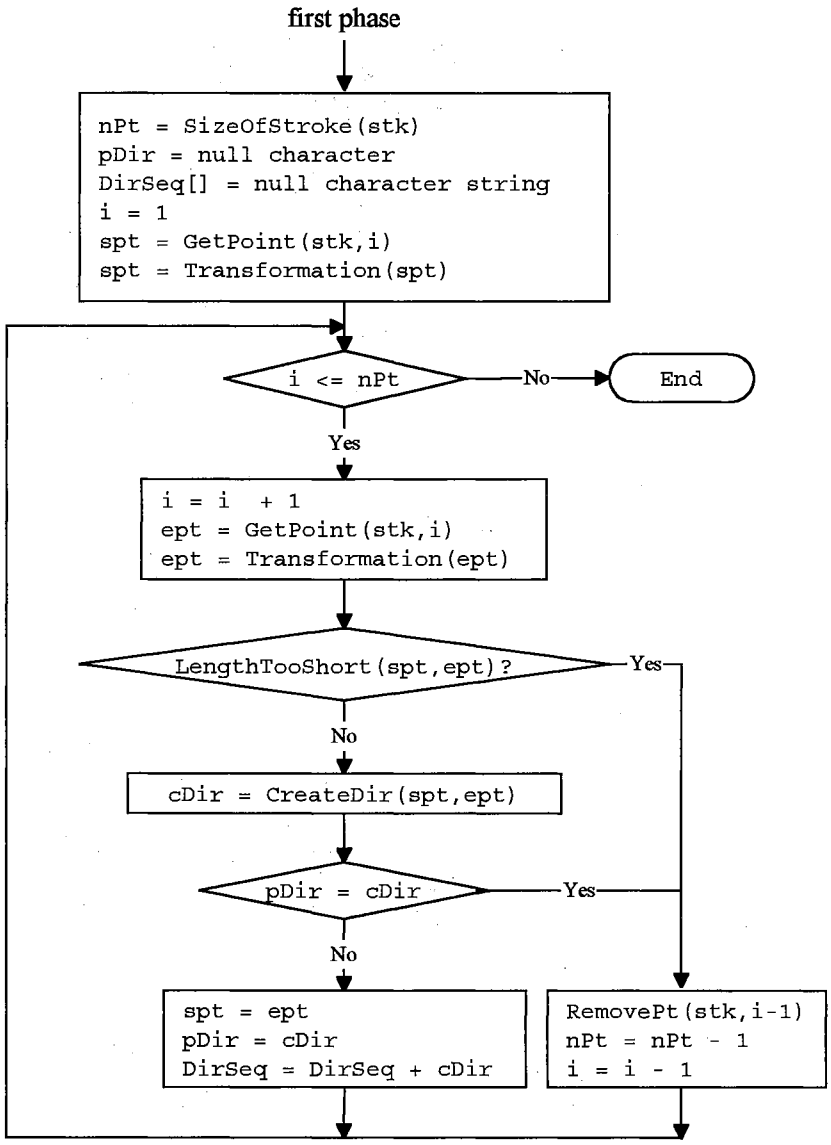


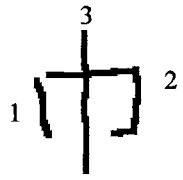
Figure 3.11: Primitive stroke extraction scheme -- second phase.

3.6 Character Representation

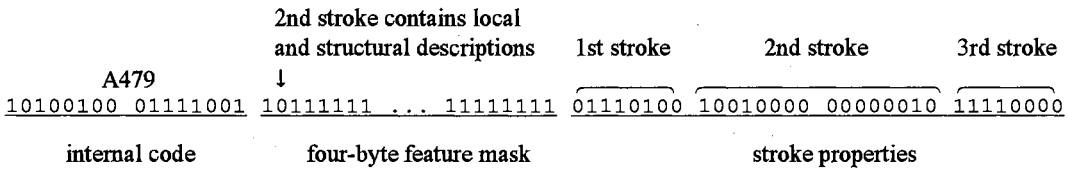
In the dissertation, the physical representation of a Chinese character consists of four parts: internal character code, stroke count, feature mask, and stroke properties. The internal character code, feature mask, and stroke properties are stored in the main database file while stroke count is stored in the higher level index file. In other words, characters with the same stroke count are stored together for searching.

Due to the size of the Chinese character set, a character is represented by two bytes. The most popular two-byte internal code in Taiwan is the Big-5 code. It was published by the Institute of Information Industry of Taiwan in May 1984. There are 5,401 most frequently used Chinese characters. The range of the Big-5 code is from A440 to C67E in hex where the first byte is from A4 to C6 and the second byte is from 40 to 7E or from A1 to FE.

By the definition of local and structural features, it appears that local features of a stroke cannot be obtained until the end of the writing of a character. This is because the normalization of the entire character is required for consistency in encoding. Yet, the extraction of a primitive stroke can be achieved right after the input of a stroke (from pen-down to pen-up). To make connections between both local and structural features for each stroke, we reserve a four-byte feature mask to represent the links from local descriptions to structural descriptions of a character. This four-byte feature mask gives a maximum of 32 links to strokes in order started from the least significant bit. If a bit is set to 0, then the corresponding stroke contains a primitive stroke. Otherwise, it



(a) standard writing sequence



(b) binary representation

Figure 3.12: Standard writing sequence for character 中 and its binary representation in the template database.

contains a solely local description. The last part is the stroke properties, where a stroke is represented either by a local description or a local description plus a structural description.

Figure 3.12 shows an example of the standard writing sequence for character 中 and its binary representation. Notice that the second stroke is a type-2 primitive stroke; therefore, the second bit started from the least significant bit in the four-byte feature mask is set to zero. This represents that stroke property of the second stroke consists of both local description and structural description (see Figure 3.12(b)). As for the other strokes, only local descriptions are stored in the database.

3.7 Summary of Feature Selection and Representation

The representation of a stroke as well as the representation of a character is presented. With the combination of local and structural features, differences among similar characters can be found. This is under the assumption that the combinations of local features are distinct and specific primitive strokes are detectable.

It would be interesting to see how effective this new encoding scheme is when it is used in a particular application. It will require experiments to see whether the representation of a stroke is sufficient for discrimination. These questions are left for the similarity measure and experiments.

CHAPTER IV

SIMILARITY MEASURE

4.1 Introduction

In this chapter, a new measure of similarity between an input and a template is proposed based on the encoding method presented in chapter III. Recall that every template stroke consists of local description, which itself is further divided into different categories. Occasionally, a stroke may be classified as one of the structural primitives. With the information from the template, the major problem is how to estimate the resemblance of a physical input to a template. To deal with this, we wish to apply fuzzy membership functions to the encoded template strokes for the local similarity measure. Then, the structural property is considered for the overall similarity measure between two strokes. Finally, the optimal matching result among strokes from an input character to a template character is processed based on the solution to the maximum assignment problems in linear programming.

4.2 Review of Fuzzy Sets

This section reviews fuzzy set theory, in which the answer is no longer simple "yes" or "no." This implies that most real-word classes are fuzzy in nature. This

uncertainty introduces inexactness and imprecision that are all too common in real life. Clearly there is no unique way to build a fuzzy theory to represent something between "yes" and "no." Many alternative approaches have been proposed to deal with this. The most intuitive one is the fuzzy set theory proposed by Zadeh in 1965 [21]. Thereafter, a great amount of publications [22-24] related to this area has been published.

The use of fuzzy sets in pattern recognition originated in 1966 with Bellman, Kalaba, and Zadeh [18]. Many research articles can be found about application of fuzzy techniques to pattern recognition [19,20]. The following review is based on their results.

Let X be a classical (crisp) set of collection of elements x and $x \subset X$. Membership of element x in a classical subset A of X is either belong to or not belong to the A. This property is described by a characteristic function (or membership function) $\mu_A(x)$ from X to $\{0,1\}$ such that

$$\mu_A(x) = \begin{cases} 1 & \text{iff} \quad x \in A \\ 0 & \text{iff} \quad x \notin A \end{cases}. \tag{4.1}$$

If the membership is represented by real interval [0,1], A is called a fuzzy set. From this point of view, a characteristic function is also called grade of membership, degree of compatibility, or degree of truth. This property implies that the closer the value of $\mu_A(x)$ is to 1 the more the x belongs to A.

There are different ways of denoting a fuzzy set. Two major ways are described as follows. A fuzzy set A of X can be completely characterized by the set of ordered pairs:

$$A = \{(x, \mu_A(x)) \mid x \in X\}. \tag{4.2}$$

The first element denotes element or object and the second element denotes the degree of membership. A fuzzy set A is represented solely by stating its membership function:

$$A = \sum_{i=1}^{n} \mu_A(x_i)/x_i \tag{4.3}$$

or

$$A = \int_X \mu_A(x_i)/x_i \ . \tag{4.4}$$

The essential issue of fuzzy sets is the membership function. Researchers have proposed many ways to define operations based on membership functions. One of the major trends is to extend the set-theoretical operations from classical sets to fuzzy sets. An example of one of the basic set-theoretic operations, *intersection*, is given as follows. The *intersection* of fuzzy sets A and B ($A \cap B$) is defined by

$$\forall x \in X, \ \ \mu_{A \cap B}(x) = \min\left(\mu_A(x), \mu_B(x)\right) \ . \tag{4.5}$$

4.3 Similarity Measure between Strokes

In Chapter II, we mentioned that the differences between characters are important for the discrimination because a classification strongly depends on the distinctness of features. This implies that the differences between strokes are crucial for discrimination between one stroke and other similar ones. The set-theoretical operation, *intersection*, is capable of describing the characteristics of a stroke from various features. Nevertheless, a common scheme is required to represent the degree of similarity for different features before the *intersection* can be applied. Recall that local features of every template stroke are encoded, while an input stroke contains a physical information.

To tie the information from both sides, we propose to apply membership functions for the measure of local similarity between an input stroke and a template stroke. The overall similarity between two strokes is then based on the initial local measure with the consideration of structural resemblance.

Without loss of generality, we will consider the *i*th input stroke and the *j*th template stroke throughout the discussion. Meanwhile, the stroke counts for both input and template are presumed to be equal (say n). Therefore, local properties of the *i*th input stroke are represented by the quantitative descriptors; i.e., l_i, θ_{ti}, d_i and θ_{di}. On the other hand, local properties of the *j*th mask stroke are denoted by L_j, T_j, P_j and D_j where the physical quantitative representations are not available after encoding.

The degree of similarity corresponding to length, type, position and direction is characterized by μ_{L_j}, μ_{T_j}, μ_{P_j} and μ_{D_j} individually. Owing to the method of encoding, various membership functions are defined to depict individual characteristics. Thus, the activation of a membership function is guided by the classification of the local feature from a template stroke. For example, μ_{L_j} is further described by $\mu_{L_j=0}$, $\mu_{L_j=1}$, $\mu_{L_j=2}$ and $\mu_{L_j=3}$ where the encoded stroke length L_j is equal to 0, 1, 2 and 3 correspondingly. This rule is applied to trigger membership functions μ_{T_j}, μ_{P_j} and μ_{D_j} as well. From this point of view, we attempt to define membership functions as sketched in Figure 4.1.

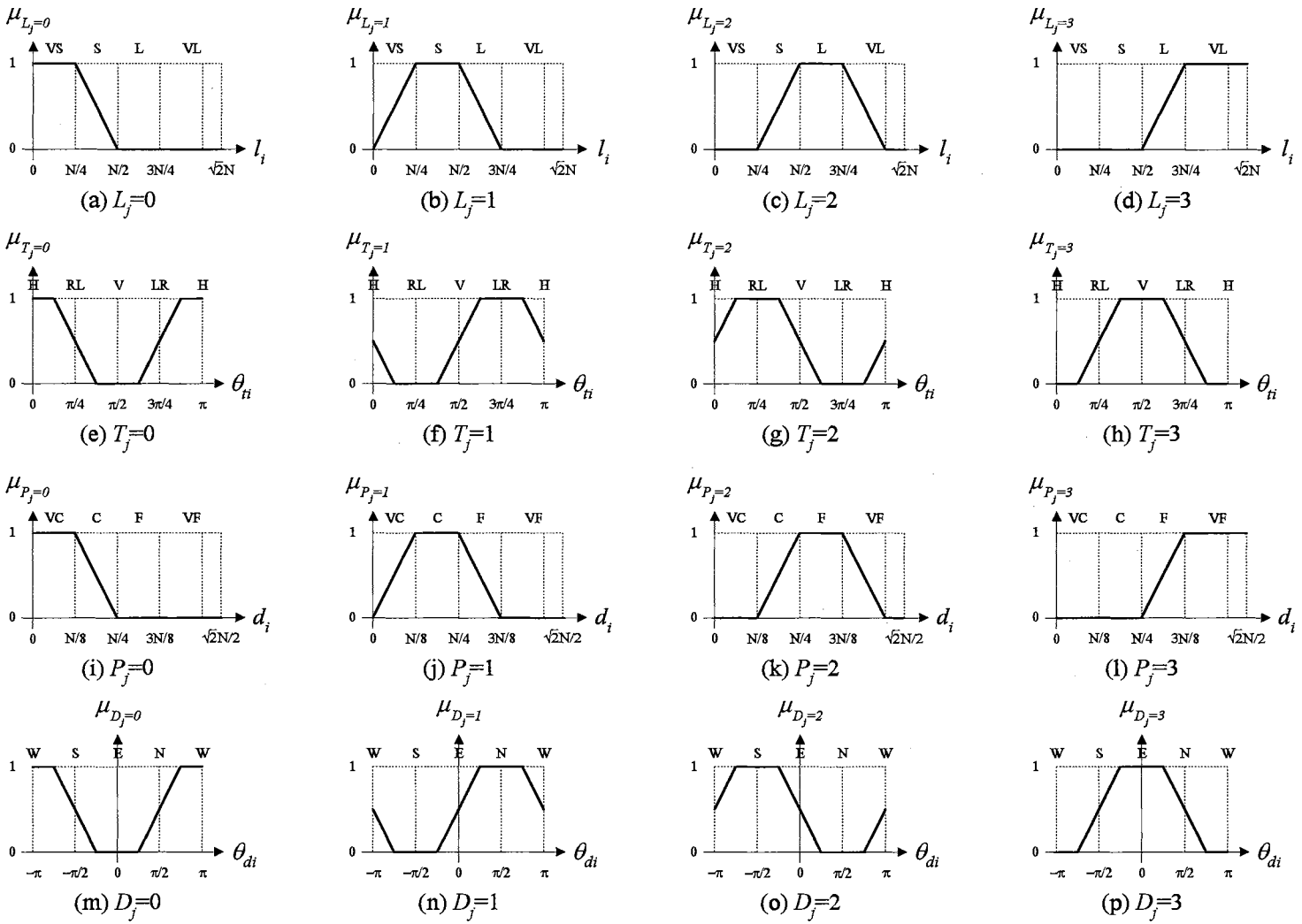


Figure 4.1: Membership functions for the classification of local features.

Therefore, membership functions for length measure corresponding to each classification are defined as

$$\mu_{L_j=0}(l_i)=\begin{cases}1 & \text{if } l_i < \mathrm{N}/4 \\ (-4/\mathrm{N})(l_i-\mathrm{N}/2) & \text{if } \mathrm{N}/4 \le l_i < \mathrm{N}/2 \\ 0 & \text{otherwise}\end{cases}, \tag{4.6a}$$

$$\mu_{L_j=1}(l_i)=\begin{cases}1 & \text{if } \mathrm{N}/4 \le l_i < \mathrm{N}/2 \\ (4/\mathrm{N})l_i & \text{if } l_i < \mathrm{N}/4 \\ (-4/\mathrm{N})(l_i-3\mathrm{N}/4) & \text{if } \mathrm{N}/2 \le l_i < 3\mathrm{N}/4 \\ 0 & \text{otherwise}\end{cases}, \tag{4.6b}$$

$$\mu_{L_j=2}(l_i)=\begin{cases}1 & \text{if } \mathrm{N}/2 \le l_i < 3\mathrm{N}/4 \\ (4/\mathrm{N})(l_i-\mathrm{N}/4) & \text{if } \mathrm{N}/4 \le l_i < \mathrm{N}/2 \\ (-4/\mathrm{N})(l_i-\mathrm{N}) & \text{if } 3\mathrm{N}/4 \le l_i < \mathrm{N} \\ 0 & \text{otherwise}\end{cases} \tag{4.6c}$$

and

$$\mu_{L_j=3}(l_i)=\begin{cases}1 & \text{if } 3\mathrm{N}/4 \le l_i \\ (4/\mathrm{N})(l_i-\mathrm{N}/2) & \text{if } \mathrm{N}/2 \le l_i < 3\mathrm{N}/4 \\ 0 & \text{otherwise}\end{cases}. \tag{4.6d}$$

To activate the correct membership function, let

$$\delta_{\mathrm{VS}}(L_j)=\begin{cases}1 & \text{if } L_j = 0 \\ 0 & \text{otherwise}\end{cases}, \tag{4.7a}$$

$$\delta_{\mathrm{S}}(L_j)=\begin{cases}1 & \text{if } L_j = 1 \\ 0 & \text{otherwise}\end{cases}, \tag{4.7b}$$

$$\delta_{\mathrm{L}}(L_j)=\begin{cases}1 & \text{if } L_j = 2 \\ 0 & \text{otherwise}\end{cases} \tag{4.7c}$$

and

$$\delta_{\mathrm{VL}}(L_j)=\begin{cases}1 & \text{if } L_j=3\\ 0 & \text{otherwise}\end{cases} \tag{4.7d}$$

where δ_{VS}, δ_{S}, δ_{L} and δ_{VL} are activation functions corresponding to every encoded template stroke length. Thus, from Eqs 4.6 to 4.7, the similarity of length measure can be represented by the canonical form as follows:

$$\begin{aligned}\mu_{L_j}(l_i)=\delta_{\mathrm{VS}}(L_j)\mu_{L_j=0}(l_i)+\delta_{\mathrm{S}}(L_j)\mu_{L_j=1}(l_i)+\\ \delta_{\mathrm{L}}(L_j)\mu_{L_j=2}(l_i)+\delta_{\mathrm{VL}}(L_j)\mu_{L_j=3}(l_i)\end{aligned} \tag{4.8}$$

Similarly, membership functions, activation functions and canonical forms for type measure, position measure and direction measure are defined accordingly. For type measure, membership functions are defined as

$$\mu_{T_j=0}(\theta_{ti})=\begin{cases}1 & \text{if } \theta_{ti}\le\pi/8\\ & \text{or } 7\pi/8<\theta_{ti} \text{ or } L_j=0\\ (4/\pi)(\theta_{ti}-5\pi/8) & \text{if } 5\pi/8<\theta_{ti}\le 7\pi/8\\ (-4/\pi)(\theta_{ti}-3\pi/8) & \text{if } \pi/8<\theta_{ti}\le 3\pi/8\\ 0 & \text{otherwise}\end{cases}, \tag{4.9a}$$

$$\mu_{T_j=1}(\theta_{ti})=\begin{cases}1 & \text{if } 5\pi/8<\theta_{ti}\le 7\pi/8 \text{ or } L_j=0\\ (4/\pi)(\theta_{ti}-3\pi/8) & \text{if } 3\pi/8<\theta_{ti}\le 5\pi/8\\ (-4/\pi)(\theta_{ti}-\pi/8) & \text{if } \theta_{ti}\le\pi/8\\ (-4/\pi)(\theta_{ti}-9\pi/8) & \text{if } 7\pi/8<\theta_{ti}\\ 0 & \text{otherwise}\end{cases}, \tag{4.9b}$$

$$\mu_{T_j=2}(\theta_{ti})=\begin{cases}1 & \text{if } \pi/8<\theta_{ti}\le 3\pi/8 \text{ or } L_j=0\\ (4/\pi)(\theta_{ti}+\pi/8) & \text{if } \theta_{ti}\le\pi/8\\ (4/\pi)(\theta_{ti}-7\pi/8) & \text{if } 7\pi/8<\theta_{ti}\\ (-4/\pi)(\theta_{ti}-5\pi/8) & \text{if } 3\pi/8<\theta_{ti}\le 5\pi/8\\ 0 & \text{otherwise}\end{cases} \tag{4.9c}$$

and

$$\mu_{T_j=3}(\theta_{ti}) = \begin{cases} 1 & \text{if } 3\pi/8 < \theta_{ti} \le 5\pi/8 \text{ or } L_j = 0 \\ (4/\pi)(\theta_{ti} - \pi/8) & \text{if } \pi/8 < \theta_{ti} \le 3\pi/8 \\ (-4/\pi)(\theta_{ti} - 7\pi/8) & \text{if } 5\pi/8 < \theta_{ti} \le 7\pi/8 \\ 0 & \text{otherwise} \end{cases} . \quad (4.9d)$$

Activation functions for relative classifications are given as

$$\delta_{\mathrm{H}}(T_j) = \begin{cases} 1 & \text{if } T_j = 0 \\ 0 & \text{otherwise} \end{cases}, \quad (4.10a)$$

$$\delta_{\mathrm{LR}}(T_j) = \begin{cases} 1 & \text{if } T_j = 1 \\ 0 & \text{otherwise} \end{cases}, \quad (4.10b)$$

$$\delta_{\mathrm{RL}}(T_j) = \begin{cases} 1 & \text{if } T_j = 2 \\ 0 & \text{otherwise} \end{cases} \quad (4.10c)$$

and

$$\delta_{\mathrm{V}}(T_j) = \begin{cases} 1 & \text{if } T_j = 3 \\ 0 & \text{otherwise} \end{cases} . \quad (4.10d)$$

From Eqs 4.9 to 4.10, the similarity for type measure can be written in the following canonical form; i.e.,

$$\begin{aligned} \mu_{T_j}(\theta_{ti}) = {} & \delta_{\mathrm{H}}(T_j)\,\mu_{T_j=0}(\theta_{ti}) + \delta_{\mathrm{LR}}(T_j)\,\mu_{T_j=1}(\theta_{ti}) + \\ & \delta_{\mathrm{RL}}(T_j)\,\mu_{T_j=2}(\theta_{ti}) + \delta_{\mathrm{V}}(T_j)\,\mu_{T_j=3}(\theta_{ti}) \end{aligned} . \quad (4.11)$$

For position measure, membership functions are defined by

$$\mu_{P_j=0}(d_i) = \begin{cases} 1 & \text{if } d_i < \mathrm{N}/8 \\ (-8/\mathrm{N})(d_i - \mathrm{N}/4) & \text{if } \mathrm{N}/8 \le d_i < \mathrm{N}/4 \\ 0 & \text{otherwise} \end{cases}, \quad (4.12a)$$

$$\mu_{P_j=1}(d_i) = \begin{cases} 1 & \text{if } \mathrm{N}/8 \le d_i < \mathrm{N}/4 \\ (8/\mathrm{N})d_i & \text{if } d_i < \mathrm{N}/8 \\ (-8/\mathrm{N})(d_i - 3\mathrm{N}/8) & \text{if } \mathrm{N}/4 \le d_i < 3\mathrm{N}/8 \\ 0 & \text{otherwise} \end{cases}, \quad (4.12b)$$

$$\mu_{P_j=2}(d_i)=\begin{cases}1 & \text{if } \mathrm{N}/4 \le d_i < 3\mathrm{N}/8 \\ (8/\mathrm{N})(d_i-\mathrm{N}/8) & \text{if } \mathrm{N}/8 \le d_i < \mathrm{N}/4 \\ (-8/\mathrm{N})(d_i-\mathrm{N}/2) & \text{if } 3\mathrm{N}/8 \le d_i < \mathrm{N}/2 \\ 0 & \text{otherwise}\end{cases} \tag{4.12c}$$

and

$$\mu_{P_j=3}(d_i)=\begin{cases}1 & \text{if } 3\mathrm{N}/8 \le d_i \\ (8/\mathrm{N})(d_i-\mathrm{N}/4) & \text{if } \mathrm{N}/4 \le d_i < 3\mathrm{N}/8 \\ 0 & \text{otherwise}\end{cases}. \tag{4.12d}$$

Activation functions for position measure are given as

$$\delta_{\mathrm{VC}}(P_j)=\begin{cases}1 & \text{if } P_j=0 \\ 0 & \text{otherwise}\end{cases}, \tag{4.13a}$$

$$\delta_{\mathrm{C}}(P_j)=\begin{cases}1 & \text{if } P_j=1 \\ 0 & \text{otherwise}\end{cases}, \tag{4.13b}$$

$$\delta_{\mathrm{F}}(P_j)=\begin{cases}1 & \text{if } P_j=2 \\ 0 & \text{otherwise}\end{cases} \tag{4.13c}$$

and

$$\delta_{\mathrm{VF}}(P_j)=\begin{cases}1 & \text{if } P_j=3 \\ 0 & \text{otherwise}\end{cases}. \tag{4.13d}$$

Therefore, the similarity for position measure can be written in the following canonical form by Eqs 4.12 and 4.13:

$$\begin{aligned}\mu_{P_j}(d_i)=\ & \delta_{\mathrm{VC}}(P_j)\mu_{P_j=0}(d_i)+\delta_{\mathrm{C}}(P_j)\mu_{P_j=1}(d_i)+ \\ & \delta_{\mathrm{F}}(P_j)\mu_{P_j=2}(d_i)+\delta_{\mathrm{VF}}(P_j)\mu_{P_j=3}(d_i)\end{aligned}. \tag{4.14}$$

As for direction measure, corresponding membership functions are given as

$$\mu_{D_j=0}(\theta_{di}) = \begin{cases} 1 & \text{if } \theta_{di} \le -3\pi/4 \\ & \text{or } 3\pi/4 < \theta_{di} \text{ or } P_j = 0 \\ (2/\pi)(\theta_{di} - \pi/4) & \text{if } \pi/4 < \theta_{di} \le 3\pi/4 \\ (-2/\pi)(\theta_{di} + \pi/4) & \text{if } -3\pi/4 < \theta_{di} \le -\pi/4 \\ 0 & \text{otherwise} \end{cases}, \quad (4.15a)$$

$$\mu_{D_j=1}(\theta_{di}) = \begin{cases} 1 & \text{if } \pi/4 < \theta_{di} \le 3\pi/4 \text{ or } P_j = 0 \\ (2/\pi)(\theta_{di} + \pi/4) & \text{if } -\pi/4 < \theta_{di} \le \pi/4 \\ (-2/\pi)(\theta_{di} + 3\pi/4) & \text{if } \theta_{di} \le -3\pi/4 \\ (-2/\pi)(\theta_{di} - 5\pi/4) & \text{if } 3\pi/4 < \theta_{di} \\ 0 & \text{otherwise} \end{cases}, \quad (4.15b)$$

$$\mu_{D_j=2}(\theta_{di}) = \begin{cases} 1 & \text{if } -3\pi/4 < \theta_{di} \le -\pi/4 \text{ or } P_j = 0 \\ (2/\pi)(\theta_{di} + 5\pi/4) & \text{if } \theta_{di} \le -3\pi/4 \\ (2/\pi)(\theta_{di} - 3\pi/4) & \text{if } 3\pi/4 < \theta_{di} \\ (-2/\pi)(\theta_{di} - \pi/4) & \text{if } -\pi/4 < \theta_{di} \le \pi/4 \\ 0 & \text{otherwise} \end{cases} \quad (4.15c)$$

and

$$\mu_{D_j=3}(\theta_{di}) = \begin{cases} 1 & \text{if } -\pi/4 < \theta_{di} \le \pi/4 \text{ or } P_j = 0 \\ (2/\pi)(\theta_{di} + 3\pi/4) & \text{if } -3\pi/4 < \theta_{di} \le -\pi/4 \\ (-2/\pi)(\theta_{di} - 3\pi/4) & \text{if } \pi/4 < \theta_{di} \le 3\pi/4 \\ 0 & \text{otherwise} \end{cases}. \quad (4.15d)$$

Corresponding activation functions for direction measure are defined as

$$\delta_{\mathrm{W}}(D_j) = \begin{cases} 1 & \text{if } D_j = 0 \\ 0 & \text{otherwise} \end{cases}, \quad (4.16a)$$

$$\delta_{\mathrm{N}}(D_j) = \begin{cases} 1 & \text{if } D_j = 1 \\ 0 & \text{otherwise} \end{cases}, \quad (4.16b)$$

$$\delta_{\mathrm{S}}(D_j) = \begin{cases} 1 & \text{if } D_j = 2 \\ 0 & \text{otherwise} \end{cases} \quad (4.16c)$$

and

$$\delta_{\mathrm{E}}(D_j)=\begin{cases}1 & \text{if } D_j=3\\ 0 & \text{otherwise}\end{cases}. \tag{4.16d}$$

Therefore, the canonical form for direction measure can be represented by Eqs 4.15 and 4.16:

$$\begin{aligned}\mu_{D_j}(\theta_{di}) = {} & \delta_{\mathrm{W}}(D_j)\mu_{D_j=0}(\theta_{di})+\delta_{\mathrm{N}}(D_j)\mu_{D_j=1}(\theta_{di})+\\ & \delta_{\mathrm{S}}(D_j)\mu_{D_j=2}(\theta_{di})+\delta_{\mathrm{E}}(D_j)\mu_{D_j=3}(\theta_{di})\end{aligned}. \tag{4.17}$$

In summary, the similarity measure of the *i*th input stroke to the *j*th template stroke concerning local features is considered as the fuzzy "and" of the measures of length, type, position, and direction. Given the fuzzy "and" operation corresponding to similarity measures for length, type, position and direction in Eqs 4.8, 4.11, 4.14 and 4.17, we get

$$\begin{aligned}&\mu_{ij} = \min(\mu_{L_j}(l_i), \mu_{T_j}(\theta_{ti}), \mu_{P_j}(d_i), \mu_{D_j}(\theta_{di})) \text{ for}\\ &\forall\, l_i \in \left[0, \sqrt{2}\mathrm{N}\right], \theta_{ti} \in [0,\pi), d_i \in \left[0, \sqrt{2}\mathrm{N}/2\right], \text{and } \theta_{di} \in (-\pi,\pi]\end{aligned}. \tag{4.18}$$

Recall that when the length of a stroke is classified as very short (VS), the type of stroke is considered unstable. Similarly, when the position of a stroke is too close (VC) to the center of a character, its direction is not reliable. To ignore this unreliable information, we must maximize μ_{T_j} and μ_{D_j} in Eq 4.18 such that the overall similarity depends heavily on μ_{L_j} and μ_{P_j} during the minimum operation. Since the measure of similarity is guided by the local classification of a template stroke, we say that when L_j is equal to 0 (VS), μ_{T_j} is set to 1. Therefore, Eqs 4.9a to 4.9d must evaluate the criteria when L_j is equal to 0. Similarly, when P_j is equal to 0 (VC), μ_{D_j} is set to 1. Thus, Eqs 4.15a to 4.15d have taken into consideration when P_j is equal to 0.

Eq 4.18 considers only local properties; ambiguity exists when more than two input strokes that have same degree of similarity and compete each other for matching to the same mask stroke. To reduce ambiguity as well as improve the matching result, structural properties are used. Let c_{ij} be defined as the degree of confidence corresponding to the primitive stroke types from the ith input stroke to the jth mask stroke. Define

$$c_{ij} = \begin{cases} 1.5 & \text{if } v_i = v_j \text{ and } \mu_{ij} > 0.5 \\ 0.5 & \text{if } v_i \neq v_j \\ 1 & \text{otherwise} \end{cases} \tag{4.19}$$

where v_i and v_j represent the primitive stroke types of the input ith stroke and the mask jth stroke individually. The objective is to magnify the overall similarity when there is enough confidence from the local measure and both strokes consist of the same primitive type; on the other hand, the similarity is decreased when both strokes consist of different primitives. Finally, the similarity measure from the ith input stroke to the jth template stroke corresponding to local and structural features is defined as

$$r_{ij} = c_{ij}\mu_{ij} \ . \tag{4.20}$$

4.4 Similarity Measure between Characters

In the previous section, we have introduced a measure of similarity from an input stroke to a template stroke. Recall that every Chinese character is a collection of strokes; the similarity between an input character and a template character is based on matching among strokes. Suppose we are given an n x n rating matrix R where every

entry is the rating (degree of similarity) of matching of an input stroke to a template stroke (see Figure 4.2(a)). Our goal is to find a feasible assignment from every input stroke to a template stroke, such that the overall similarity is optimal. This problem can be formulated as a maximum assignment problem in linear programming.

Given any n x n rating matrix $R = [r_{ij}]$, find an n x n permutation matrix $X = [x_{ij}]$ that maximizes

$$Z = \sum_{i=1}^{n} \sum_{j=1}^{n} r_{ij} x_{ij} \tag{4.21}$$

where

$$x_{ij} = \begin{cases} 1 & \text{if the } i\text{th input stroke is assigned} \\ & \text{(matched) to the } j\text{th template stroke.} \\ 0 & \text{otherwise} \end{cases} \tag{4.22}$$

Since each template stroke matches exactly one input stroke, we have

$$\sum_{i=1}^{n} x_{ij} = 1 \qquad \text{for } j = 1,2,..,n \ . \tag{4.23}$$

Since each input stroke matches only one template stroke, we have

$$\sum_{j=1}^{n} x_{ij} = 1 \qquad \text{for } i = 1,2,..,n \ . \tag{4.24}$$

This typical assignment problem presents some interesting characteristics. Clearly, this is a problem of n x n variables (n input strokes and n mask strokes). However, the number of solution variables (assignments) is exactly n since every input stroke can be assigned to exactly one mask stroke. For a relatively small problem size like 5 x 5, the number of possible solutions is equal to 5! = 120. An efficient scientific

method, proposed by König [117], to solve similar minimum assignment problems is usually referred to as the Hungarian Method. It can be generalized to the following steps:

(1) Produce an initial reduced cost matrix to indicate the cost of assigning a job to a machine at the current stage. In general, a reduced cost matrix is defined as the matrix obtained by subtracting a constant from all entries in a row or column in the original matrix.

(2) Search for an optimum assignment. If the optimal assignment is not equal to the size of the maximum assignment (say n), go to step (3).

(3) Repeat step (2) until the optimum solution is reached.

r_{11}	r_{12}	...	r_{1n}
r_{21}	r_{22}	...	r_{2n}
...	...	...	...
r_{n1}	r_{n2}	...	r_{nn}

(a)

	v_1	v_2	...	v_n
u_1	w_{11}	w_{12}	...	w_{1n}
u_2	w_{21}	w_{22}	...	w_{2n}
...	...	...	...	...
u_n	w_{n1}	w_{n2}	...	w_{nn}

(b)

Figure 4.2: Rating matrices. (a) Representation of the original rating matrix; (b) representation of the reduced rating matrix.

The formulation of a typical minimum assignment problem and the algorithm concerning the Hungarian Method can be found in Appendix A. Proper adjustment is required for applying the Hungarian Method to the maximum assignment problem.

There are two approaches to solving maximum assignment problems by applying the Hungarian Method. The most direct approach simply negates the sign of all entries of an initial rating matrix. In other words, maximizing the sum of the positive rating is equal to minimizing the sum of the negative rating. Then, apply the Hungarian algorithm directly to the cost matrix $[-r_{ij}]$. The use of rating and cost matrices is to indicate whether a maximum or a minimum problem is considered. After the Hungarian algorithm terminates, select entries from the original rating matrix R corresponding to the permutation matrix X. This will give us the optimal solution to the maximum assignment problem.

An alternate approach is to modify the criteria in the algorithm and work on the rating matrix R directly. Recall that r_{ij} is the similarity (rating) of matching input ith stroke to template jth stroke; from Eqs 4.19 and 4.20, we get $0 \le r_{ij} \le 15$. As in the Hungarian Method, introduce u_i for each row i and v_j for each column j (see Figure 4.2(b)). Similar to Eq A.4, define

$$w_{ij} = r_{ij} - \left(u_i + v_j\right) \tag{4.25}$$

where w_{ij} is non-positive for every i and j. Thus, for any permutation matrix X we have

$$\begin{aligned}\sum_{i=1}^{n}\sum_{j=1}^{n} r_{ij}x_{ij} &= \sum_{i=1}^{n}\sum_{j=1}^{n} u_i x_{ij} + \sum_{i=1}^{n}\sum_{j=1}^{n} v_j x_{ij} + \sum_{i=1}^{n}\sum_{j=1}^{n} w_{ij}x_{ij} \\ &= \sum_{i=1}^{n} u_i + \sum_{j=1}^{n} v_j + \sum_{i=1}^{n}\sum_{j=1}^{n} w_{ij}x_{ij}\end{aligned} \quad . \tag{4.26}$$

Since w_{ij} is non-positive and x_{ij} is nonnegative, Eq 4.26 gives the inequality

$$\sum_{i=1}^{n} u_i + \sum_{j=1}^{n} v_j \geq \sum_{i=1}^{n}\sum_{j=1}^{n} r_{ij}x_{ij} \quad . \tag{4.27}$$

Clearly, the right side of Eq 4.27 is the maximum assignment problem. Since all feasible solutions on the left side of Eq 4.27 are greater than the feasible solutions on the right side, this introduces a minimum problem on the left side of Eq 4.27; i.e., find u_i and v_j that minimize

$$\sum_{i=1}^{n} u_i + \sum_{j=1}^{n} v_j \tag{4.28}$$

subject to the condition $u_i + v_j \geq r_{ij}$ for all i, j.

If we can find a solution that solves the minimum problem as well as the maximum assignment problem in Eq 4.27, then both will be optimal. The above statement is true if and only if either x_{ij} or w_{ij} is zero; i.e.,

$$\sum_{i=1}^{n}\sum_{j=1}^{n} w_{ij}x_{ij} = 0 \quad . \tag{4.28}$$

From this point forward, the König Algorithm, a sub-algorithm in the Kuhn's Hungarian Algorithm (see Appendix A), can be applied to find the solution to Eq 4.28. The central issue of the König's Algorithm is to find a largest independent set of zeros and a smallest cover of the zeros in a matrix. In our case, an independent set of zeros is

a set of zeros from a reduced rating matrix such that every zero lies in a different row and column. Therefore, an independent set of zeros with a largest number of zeros is called a largest independent set of zeros. For smallest cover of the zeros, we use a set of vertical and horizontal lines that covers every zero in the matrix. König has proved that the number of zeros in a largest independent set of zeros is equal to the number of lines in a smallest cover of zeros (we shall not prove this here). If the above case has been reached and the number of zeros of the largest independent set of zeros is equal to the number of the solution variables (n), the optimal assignment has been found.

Having discussed the above; we shall show a modification to the criterion of the Hungarian Method. Figure 4.3 presents the diagram of the modified Hungarian Algorithm for the maximum assignment problem. This modification is based on the Kuhn's Hungarian Algorithm discussed in Nering [116]. To produce non-positive w_{ij} for every i and j in Eq 4.25, we initialize w_{ij} by

$$\begin{aligned} u_i &= \max_j r_{ij} \\ v_j &= \max_i (r_{ij} - u_i) \\ w_{ij} &= r_{ij} - (u_i + v_j) \end{aligned} \quad . \tag{4.29}$$

Thereafter, the algorithm continues in the same way as before, except the maximum uncovered entry, e, is used to produce the new reduced rating matrix.

In the following, a step by step procedure showing the maximum similarity between an input character to a template by applying the modified Hungarian Method is presented. For consistency of representation, designate $[r_{ij}]$ as the original rating matrix

as in Figure 4.2(a). The reduced rating matrix, $[w_{ij}]$, is given in Figure 4.2 (b) where u_i, v_j are used to calculate w_{ij} as introduced in Eq 4.27.

Suppose we are given a rating matrix as in Figure 4.4. First, we produce the initial reduced rating matrix by applying Eq 4.29. Thus, we subtract every entry in a row with the maximum in that row and write the row maximum on the left margin (u_i). This will produce an intermediate result as in Figure 4.5(a). Starting with Figure 4.5(a), subtract every entry in a column with the maximum in that column and leave the column maximum on the top margin (v_j). This will give us an initial reduced rating matrix as in

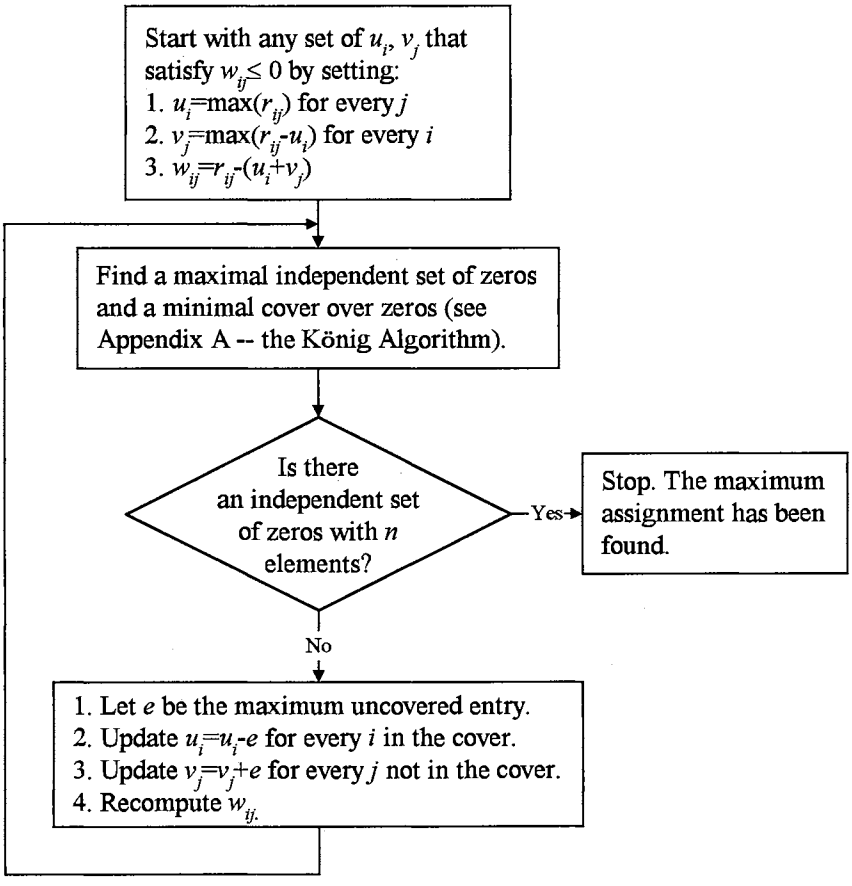


Figure 4.3: Diagram of the modified Kuhn's Hungarian Algorithm for maximum assignment problems.

Figure 4.5(b).

Now, we must find a largest independent set of zeros and a minimal cover of zeros for the reduced rating matrix in Figure 4.5(b). As a start from König's Algorithm,

0.9	0.6	0.7	0.4	0
0.4	0.8	0.4	0.6	0.5
0.3	0.7	0.3	0	0.2
0.2	0.3	0	0.6	1
0.3	0.1	0.9	0.1	0.2

Figure 4.4: Original rating matrix.

we will form a complete independent set of starred zeros. By definition, an independent set of zeros that cannot be expanded to a larger independent set is said to be complete. A consistent approach is to search every row from the first column and star the first zero that is not in a column of a previously starred zero. Shade an entire row that contains a starred zero and label the row with 0 at the right edge of the matrix (row label) for those that do not have starred zeros. This will give us a complete independent set of zeros (starred zeros) and a cover (shaded area) that covers all starred zeros (see Figure 4.6). Though we have replaced the lines of a cover by shades over an entire row or column for

0.9	0	-0.3	-0.2	-0.5	-0.9
0.8	-0.4	0	-0.4	-0.2	-0.3
0.7	-0.4	0	-0.4	-0.7	-0.5
1	-0.8	-0.7	-1	-0.4	0
0.9	-0.6	-0.8	0	-0.8	-0.7

(a)

	0	0	0	-0.2	0
0.9	0	-0.3	-0.2	-0.3	-0.9
0.8	-0.4	0	-0.4	0	-0.3
0.7	-0.4	0	-0.4	-0.5	-0.5
1	-0.8	-0.7	-1	-0.2	0
0.9	-0.6	-0.8	0	-0.6	-0.7

(b)

Figure 4.5: Reduced rating matrices. (a) Intermediate reduced rating matrix; (b) initial reduced rating matrix.

	0	0	0	-0.2	0	
0.9	0*	-0.3	-0.2	-0.3	-0.9	
0.8	-0.4	0*	-0.4	0	-0.3	
0.7	-0.4	0	-0.4	-0.5	-0.5	0
1	-0.8	-0.7	-1	-0.2	0*	
0.9	-0.6	-0.8	0*	-0.6	-0.7	

Figure 4.6: A complete independent set of zeros (starred zeros) and a cover (shaded area).

presentation, they are for the same purpose.

Next, look for the row with an uncovered zero. Clearly, there is one zero that is not covered in Figure 4.6. Shade and label the column that contains the uncovered zero with the row index at the bottom edge (column label). Meanwhile, search for a starred zero in that column. If there is no starred zero in that column, we have *breakthrough.* By definition, *breakthrough* gives an alternate independent set of zeros that has one more element than the previous one. Since there is a starred zero, remove the shade from the row and label the row by the index of the column with the starred zero. This gives Figure 4.7(a). Continue in this fashion and look for an uncovered zero, shade and label the column label with the index of the row in Figure 4.7(b). In the column just labeled, look for a starred zero. Since there is no starred zero in that column, we have *breakthrough.*

Now, start from the zero in which we have *breakthrough.* Trace the path by going to the row indexed by column label and going to the column indexed by row label (see Figure 4.8(a)). During the tracing of the path, star unstarred zeros and unstar starred zeros along the path until the row label 0 is reached. Since the path begins and

ends with unstarred zeros, this guarantees that the elements of the new independent set of zeros is increased by one. Finally, we get a new independent set of zeros and a smallest cover in Figure 4.8(b). Since the number of zeros of the independent set of zeros is equal to the number of the solution variables (5 = 5), we have found the maximum assignment. Therefore, the solution to the problem is either

$$\sum_{i=1}^{5} u_i + \sum_{j=1}^{5} v_j = (0.9+0.8+0.7+1+0.9)+(0+0+0-0.2+0) = 4.1$$

or

$$\sum_{i=1}^{5}\sum_{j=1}^{5} r_{ij}x_{ij} = r_{1,1}+r_{2,4}+r_{3,2}+r_{4,5}+r_{5,3} = 0.9+0.6+0.7+1+0.9 = 4.1\ .$$

Notice that the above problem is formulated to find an optimal one-to-one matching between n input and n template strokes. To solve the problem where the number of input strokes is not equal to the number of template strokes, we may give some dummy columns or rows and pad with zeros to generalized the matching matrix. This will give us some degree of tolerance where input and template strokes are not equal.

4.5 Summary

In summary, we have shown the similarity measure between strokes and characters. By defining fuzzy membership functions corresponding to each local feature, we are able to obtain the initial similarity between strokes. The initial similarity is increased if corresponding constraints are satisfied. First, there should be enough

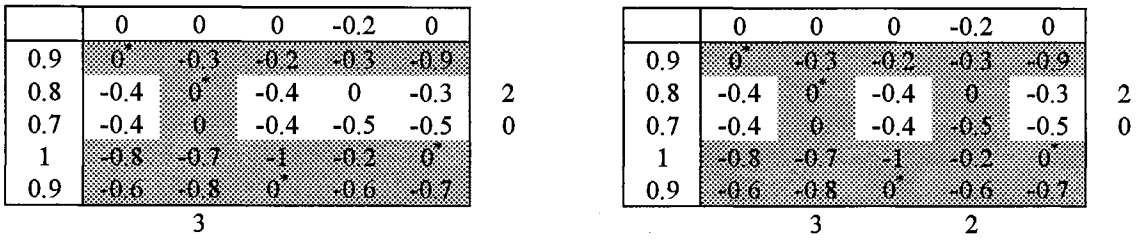


Figure 4.7: An independent set of zeros and its intermediate covers by applying the König Algorithm.

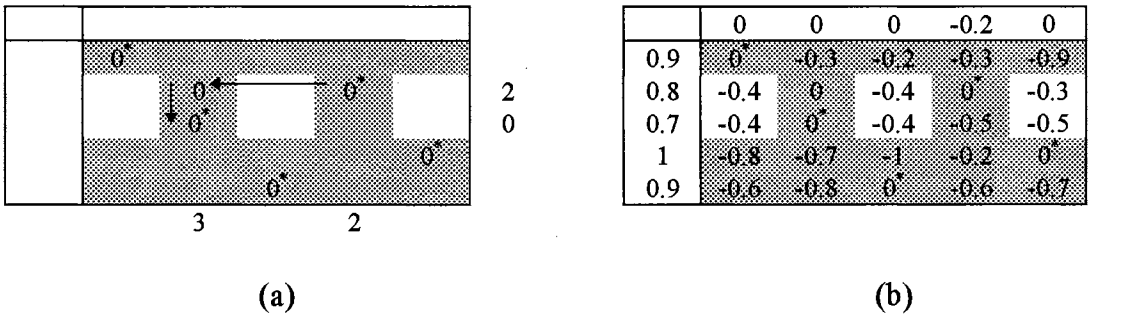


Figure 4.8: Result of a new independent set of zeros and its smallest cover. (a) Trace of the path; (b) the largest independent set of zeros and its smallest cover.

evidence to believe that two strokes are alike; i.e., initial similarity is greater than a given threshold. Second, both input and mask strokes contain the same primitive stroke type. Finally, the measure of resemblance between an input character and a mask character is formulated as a maximum assignment problem. This maximum assignment problem is then solved by the modified Hungarian Method.

CHAPTER V

IMPLEMENTATION AND EVALUATION

5.1 Introduction

The implementation and evaluation of the theoretical design are presented in this chapter. The purpose of the implementation is to evaluate the performance of the design. Two graphic user interfaces, TRAINER and RECOGNIZER, are implemented under this premise. They both provide the environment to extract stroke based dynamic information. TRAINER is for creating the initial template database; it also displays the progress of writing and the extraction of primitive strokes. RECOGNIZER is for emulating the recognition process and for sample collection. Finally, the evaluation is based on 18,000 collected and random samples from 20 volunteers.

5.2 Implementation

The implementation of the design consists of two major interfaces: TRAINER and RECOGNIZER. Both interfaces are developed by Microsoft Visual C++ under Microsoft Chinese Windows 3.1 on a personal computer. The control flows of both interfaces are based on the proposed recognition operations sketched in Figure 1.2. To receive a better writing effect, we use a graphic tablet for both database creation and sample collection.

The first problem during implementation is the creation of the initial template database. In an off-line recognition system, the creation of the template database is usually based on training among statistical samples where samples are read pixel by pixel. In an on-line system where strokes are dynamically extracted as line-based, the collection of statistical samples is not an easy task and may not always be possible. Therefore, we first build up an initial template database from direct writing on templates of the Kai (楷) font (a standard and formal handprinted writing style).

Figure 5.1 shows the first graphic user interface, TRAINER, which provides a

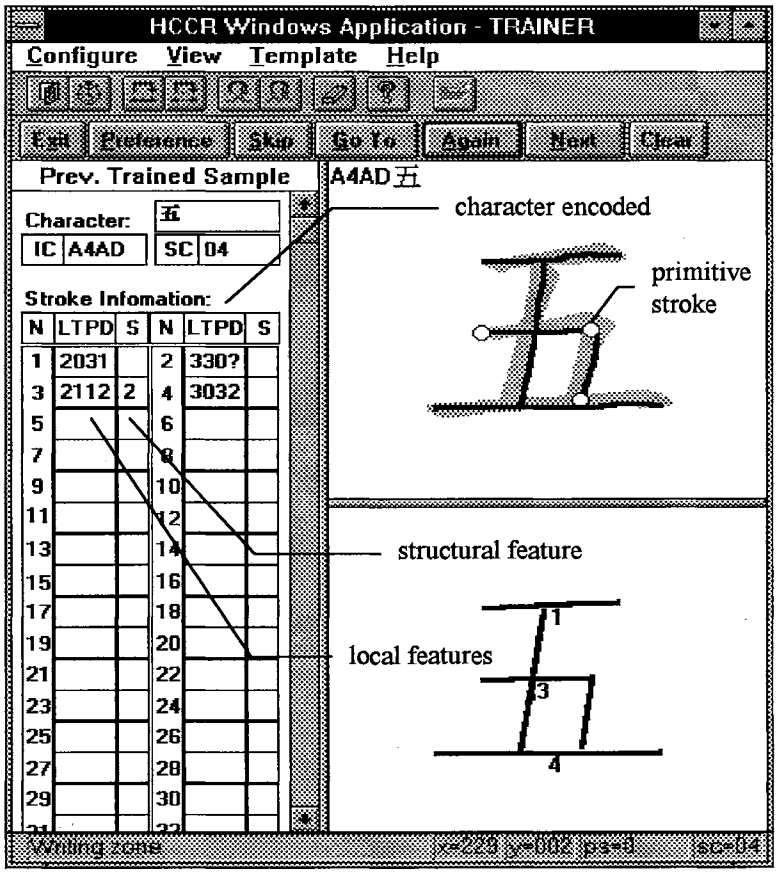


Figure 5.1: TRAINER -- A graphic user interface for creating the initial template database.

tool to supervise the writing during the creation of the initial template database. This interface has several distinct functions: identification of a primitive stroke, character encoding, and echoing the progress of writing. Algorithm of the control flow for TRAINER is presented in Figure 5.2.

```
Initialize the first template character for encoding
for every input character do
   Set stroke count to zero
   Initialize feature mask (set all bits to 1)
   for every input stroke do
      Increment stroke count by one
      if current input stroke contains a primitive stroke then
         Set corresponding bit in the feature mask to 0
      endif
      Display result of the extraction
   endfor
   Normalize each stroke of the current input character
   for every normalized input stroke do
      Extract and classify local features
   endfor
   Encode the current input character
   Advance to the next template character for encoding
endfor
```

Figure 5.2: Algorithm of the control flow for TRAINER.

To evaluate the system performance, RECOGNIZER is designed to utilize the initial template database created by TRAINER for simulating the recognition process and for sample collection. Figure 5.3 presents the design of the RECOGNIZER. As demonstrated in the example, candidates are selected based on resulting similarity in descending order for the recognition of character 絶. Meanwhile, raw data files of the

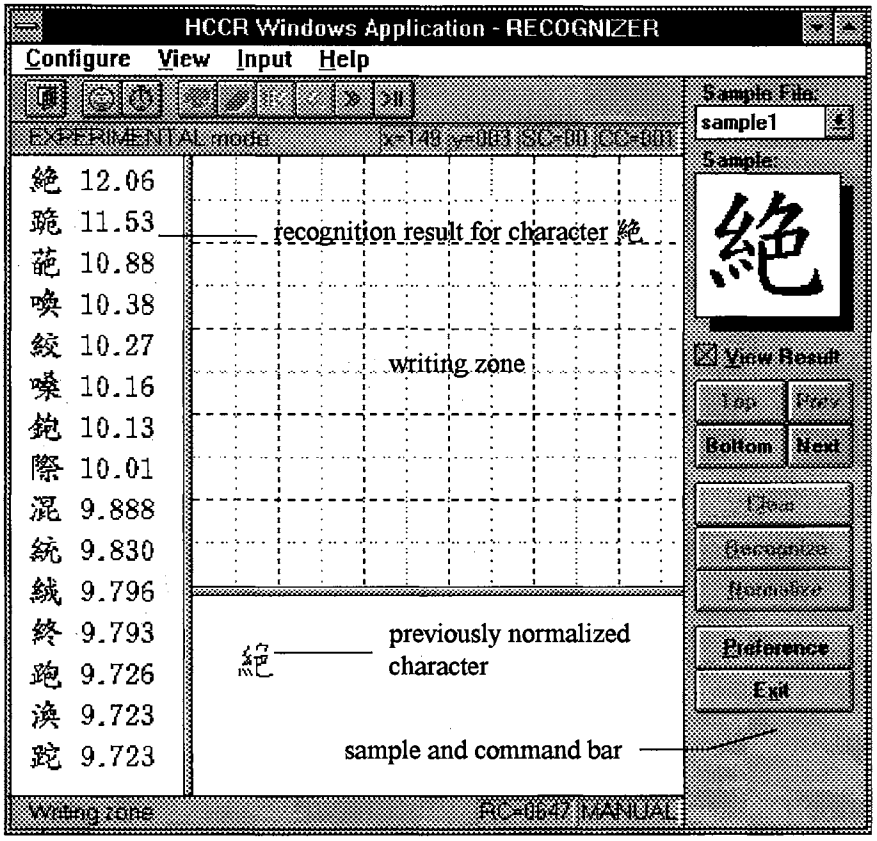


Figure 5.3: RECOGNIZER -- A graphic user interface for simulating the recognition process and for sample collection.

written samples are exported for simulation at a later stage. Algorithm of the control flow for RECOGNIZER is presented in Figure 5.4.

Although both TRAINER and RECOGNIZER are implemented for simulation, the design has taken into consideration ease of use. A thorough description for both graphic user interfaces is presented in Appendix B. In the next section, experiments and performance evaluation of the system are presented through the use of both interfaces.

```
for every input character do
   Set stroke count to zero
   Initialize feature mask (set all bits to 1)
   for every input stroke do
      Increment stroke count by one
      if current input stroke contains a primitive stroke then
         Set corresponding bit in the feature mask to 0
      endif
   endfor
   Normalize each stroke of the current input character
   for every template character selected from the database do
      for every normalized input stroke do
         Measure stroke similarity between the normalized input
              stroke to every stroke in the selected template
              character
      endfor
      Measure character similarity between the normalized input
           character to the selected template character
      Update candidate list
      Output normalized result
   endfor
   Output the list of candidates
endfor
```

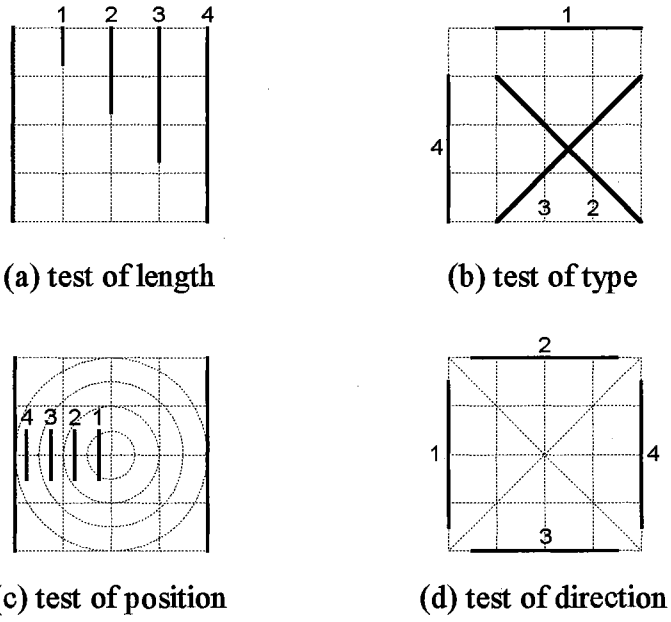
Figure 5.4: Algorithm of the control flow for RECOGNIZER

5.3 Verification of Local and Structural Encoding

For years, verification of implementation has been an important issue during the development of a software product. Not only because it is time consuming, but also because it requires a lot of human resources. Though the implementation of the system described in this dissertation is a prototype, it is crucial to verify individual components for consistency and correctness. Therefore, two experiments are designed to verify the implementation of the TRAINER. The first experiment is to verify the correctness of the implementation concerning the encoding of local features. The second experiment validates the correctness and stability of the extraction of structural features.

5.3.1 Verification of Local Encoding

To verify the correctness of the local encoding from TRAINER, a group of pre-arranged line segments is designed to test the results after encoding. Figure 5.5 presents four groups of pre-arranged line segments to test the encoding of length, type, position, and direction individually. Solid lines that are numbered from 1 to 4 represent strokes of interest while dotted lines and circles represent the imaginary grid. According to Table 3.1, solid lines numbered as 1 in Figure 5.5 should be encoded to 0, corresponding to various local features: VS (very short), H (horizontal), VC (very close) and W (west), respectively. Similarly, solid lines numbered as 2 should be encoded to 1 and so on. By using the pre-arranged line segments in Figure 5.5, the implementation of local encoding is confirmed by directly writing through TRAINER.



(a) test of length (b) test of type

(c) test of position (d) test of direction

Figure 5.5: Pre-arranged line segments to test the encoding of length, type, position, and direction.

5.3.2 Verification and Stability of Structural Encoding

To verify the extraction of structural features, we have tested twenty-three primitive strokes. They were written in a standard way as they were taught in school. As illustrated in Figure 5.6, strokes are written in the same order as in Table 3.3. Circles on a stroke represent the detected feature points during the writing. Notice that all twenty-three strokes can be identified as in Table 3.3.

In the experiment, it appears that the extraction scheme is unable to identify a primitive stroke if the stroke was written very fast and very short. Not only because

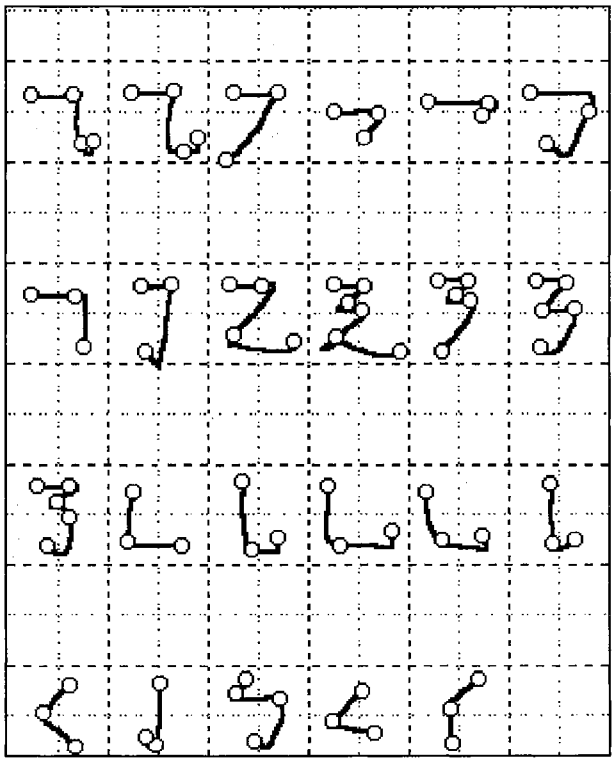


Figure 5.6: Experiment for structural feature extraction. The experiment has tested twenty-three primitive strokes from ten types (see Table 3.3) where circles on a stroke represent detected feature points during the writing.

fewer points are captured during the fast writing, but also the point elimination scheme will exclude neighbor points that are too close. As a result, it may change the direction sequence of the stroke.

Some of the primitive strokes are also sensitive to the manner of writing; such as 亅in 門 and 了 in 仔. Notice that 亅and 了 both start with a short horizontal line segment, end with a vertical line segment followed by a hook, and fall in one side of the line that connects the start point and the end point of the primitive stroke. By having these characteristics, the extraction procedure tends to fail in extracting the critical points inside the ovals illustrated in Figure 5.7. Consequently, the resulting direction sequence of a stroke may be very different. Possible solutions include rearranging the grouping of primitive strokes and modifying the extraction scheme for refinement.

5.4 Initial Template Database

Due to the lack of statistical samples, we first build up an initial template

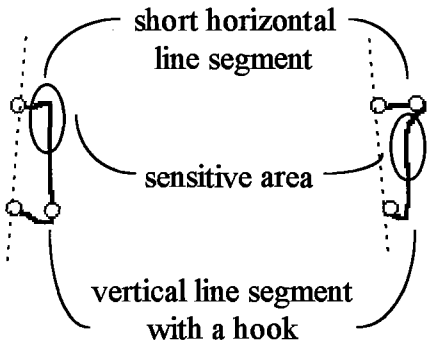


Figure 5.7: Primitive strokes that are sensitive to the writing style. They both start with a short horizontal line segment, end with a vertical line segment followed by a hook, and fall in one side of the line that connects the start point and the end point. Critical points in the ovals tend to be ignored by the extraction scheme.

database with 5,401 most frequently used characters based on the standard Kai font. This process is supervised by the author through the use of TRAINER. Table 5.1 shows some of the statistical results of the writing.

Notice that the size of the resulting template database is approximately 112 Kbytes. This result is very encouraging since the size of the database has made it possible for an application to load the entire database into primary memory for fast processing. This is particularly important for an environment where the system resource (i.e., memory space) is limited.

TABLE 5.1

STATISTICS OF THE INITIAL TEMPLATE DATABASE

number of characters	5,401
size of the database (Kbytes)	111.68
average number of strokes per character	12.25
average number of features per stroke	4.18
average number of bytes of a character	20.68

Conventional schemes of stroke based representation require detailed description of the selected features in order to portrait the characteristics of a stroke. This usually ends up needing a large database file to store the entire Chinese character set. Figure 5.8 presents the estimated file size for storing 5,401 characters corresponding to the number of features selected. For simplicity, assume each feature requires one physical byte to store the information (this may not always be true since a floating feature requires more

than one physical byte). As illustrated, our representation scheme requires less storage than the conventional approach (see Figure 5.8) for storing a similar amount of integer features. This also means that less physical bytes are required to be retrieved from the secondary storage for each character. As a result, the time of data retrieving is reduced when a large amount of information is required to load contiguously from the secondary storage. Nevertheless, since most of our features are quantified, there is more chance of ambiguity than in the conventional approach. A possible solution to the above case might be storing multiple character templates from different writing styles for a specific character.

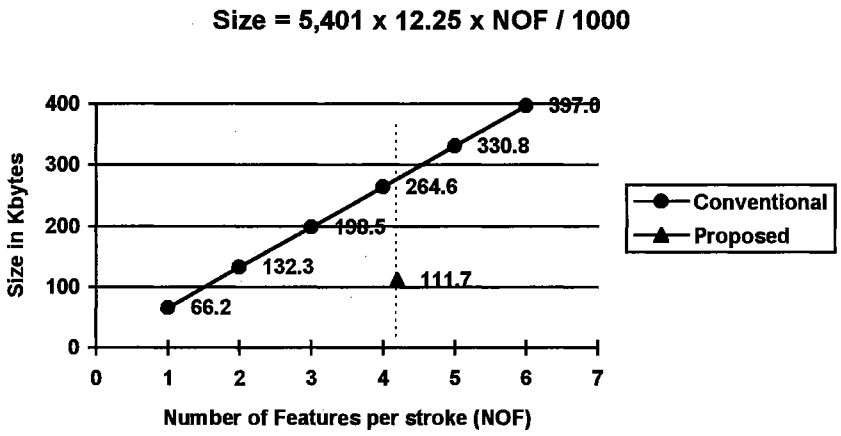


Figure 5.8: Estimated file size for 5,401 most frequently used Chinese characters with conventional approach where the average number of strokes per character is 12.25.

After writing the template characters, each character is packed based on the proposed representation scheme. The resulting database is then used as an initial template database for the recognition process. Notice that the initial database consists of

5,401 characters. It may not be appropriate to compare the entire character set to evaluate the performance since the size of the Chinese data set is too large. A Chinese recognition system usually applies a pre-candidate selection scheme to reduce the number of characters for comparison.

In our experiment, template characters are selected for comparison by three criteria: number of strokes, average projection profiles on both x and y axes. To accomplished the task, both template and input characters are required to obtain the above information during writing. The number of strokes of a template character is dynamically accumulated; average projection profiles on the x and y axes are calculated after the normalization. Since every character is a composition of short line segments, the x projected profile is the summation of lengths projected from these short line segments to the x-axis. This result is then divided by N (N = 64) to obtain the average projection profile on the x-axis. Similarly, this rule is applied to obtain the average projection profile on the y-axis. Therefore, by adjusting the displacements from an input character corresponding to the above criteria, we are able to select limited template characters for comparison.

5.5 Sample Characters

There are three randomly selected group samples (see Figure 5.9); each consists of 100 characters. During the collection of samples, writers are prompted with the next sample character to write under RECOGNIZER. RECOGNIZER also allows users to

write along by storing corresponding raw data without presenting the recognition result. The raw input data of each handprinted character contains the following information:

(a) one of the sample characters to be recognized in Figure 5.9,

(b) average x projection profile of (a) stored in the template database -- P_x,

(c) average y projection profile of (a) stored in the template database -- P_y,

(d) absolute difference between (b) and average x projection profile of the handprinted input character -- D_x,

(e) absolute difference between (c) and average y projection profile of the handprinted input character -- D_y,

(f) number of strokes of the input character -- SC, and

(g) normalized local features with or without a structural feature for each individual stroke.

The sample character to be recognized in (a) is for estimating the recognition rate at a later stage while (b), (c), (d), (e) and (f) are for the selection of reference characters. For every handprinted stroke, normalized local features (length, type, position, and direction) with or without a structural feature are used for measuring the similarity.

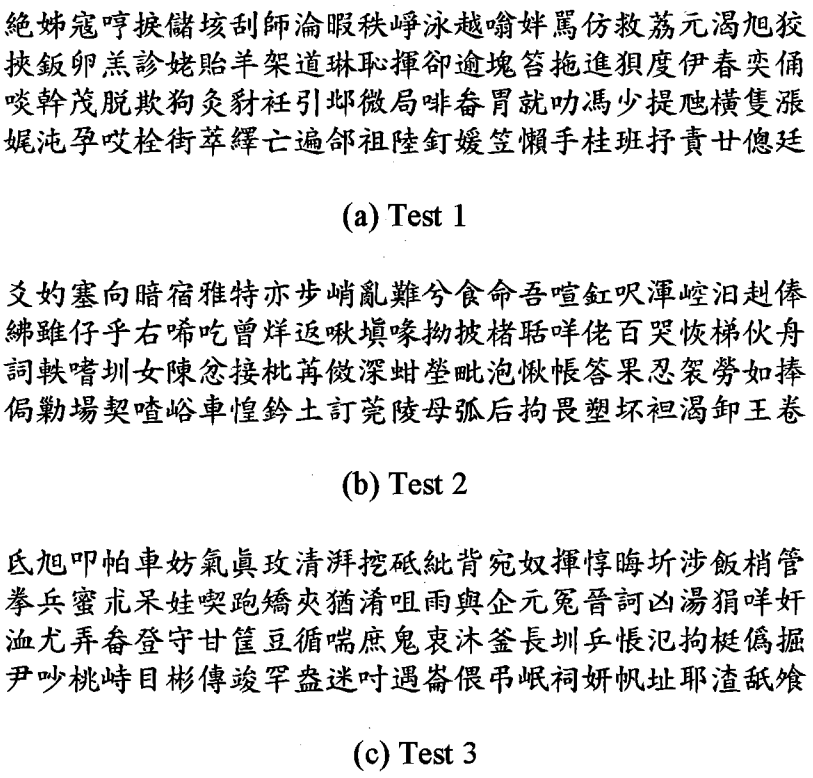
Recall that template characters are grouped by stroke count, average projection profiles on both x and y axes. For a template character to be qualified for comparison, it must satisfy the following conditions:

(1) stroke count of the template is equal to SC–1, SC, or SC+1,

(2) average x projection profile of the template is less than or equal to P_x+D_x and greater than or equal to P_x-D_x,

(3) average y projection profile of the template is less than or equal to P_y+D_y and greater than or equal to P_y-D_y.

By the above conditions, a template character is guaranteed to be in the range for comparison unless the absolute difference in stroke count is greater than 1. This will give us the opportunity to observe the performance of the recognition system.



(a) Test 1

(b) Test 2

(c) Test 3

Figure 5.9: Three selected test sets in standard Kai (楷) font.

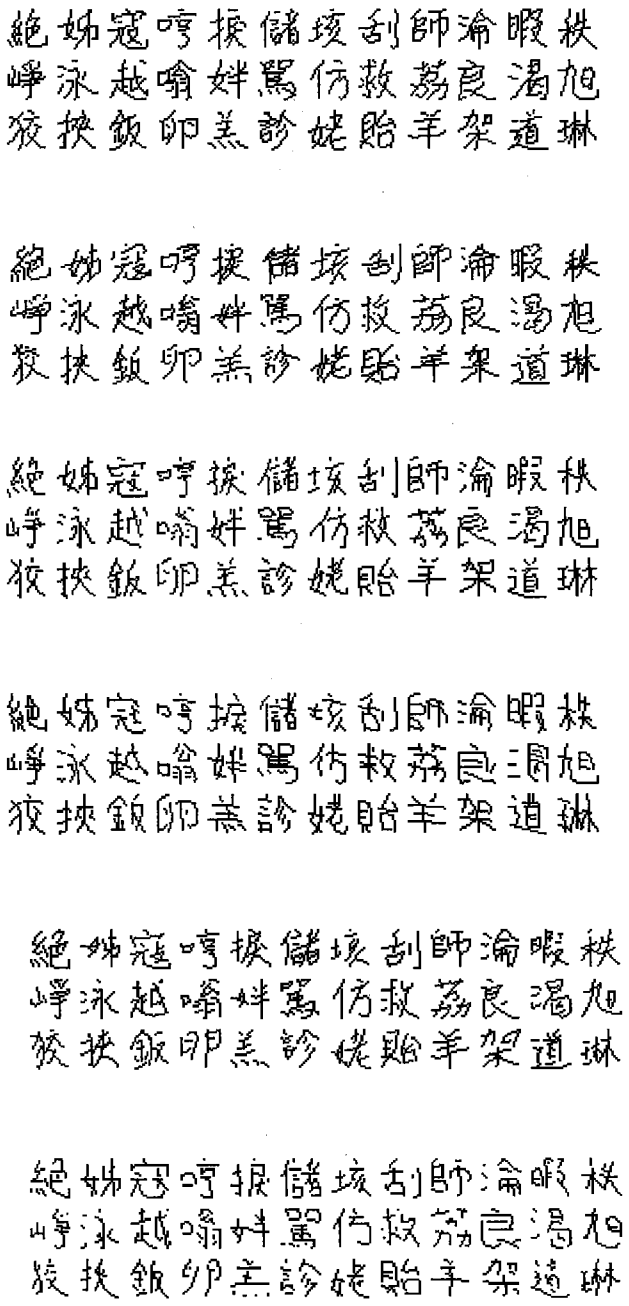


Figure 5.10: Example of partial normalized handprinted input samples during sample collection.

In the experiment, we have twenty volunteers to write all three groups of samples; this gives us 6,000 samples in total. We label this collection as ORIG for simulation. All volunteers have prior knowledge about Chinese characters. They were told to write standard printed Chinese characters. They were also instructed about types of acceptable primitive strokes presented in Table 3.3. On average, the amount of time spent in writing 300 sample characters without viewing the recognition result is about one hour. Figure 5.10 shows some of the normalized characters during the collection.

As illustrated in Figure 5.10, though all volunteers were told to write in standard printed style, every writer has certain writing preferences. That is why a general recognition system is hard to build unless the system is well trained. Although writing styles vary from person to person, writing from the same person does possess similar characteristics. Therefore, there is reason to believe that there exist only minor variations between the same character written by the same person at different times. Considering this assumption, we deliberately deviate the 6,000 collected raw data by offsetting the local properties of each stroke to generate similar input patterns for the simulation while keeping the rest of the information unchanged. Formula that changes the local properties of each stroke is defined as:

normalized local feature value + *offset.*

The *offset* is equal to 5%~10% or 10%~20% of the boundary value where boundary value is equal to N/4, $\pi/4$, N/8, or $\pi/2$ for length, type, position, and direction correspondingly. The sign of *offset* is randomly selected in the program. This gives us two random generated sample sets and each contains 6,000 character samples. These

random sample sets are labeled as RAND1 (for offsetting 5%~10%) and RAND2 (for offsetting 10%~20%) to distinguish from the original sample set (ORIG). Both sets are used to test the system performance. This will give us the opportunity to examine the system performance when similar input patters are encountered.

5.6 Evaluation of the Recognition System

5.6.1 First Stage Recognition

After samples are collected, a background recognition process is carried out by the author through the use of RECOGNIZER. The recognition result is based on comparisons between each input and some of the template characters in the initial database. Though Kai font is a standard and formal writing style, it may not closely represent the writing style used by most of the volunteers. This is due to the writing difficulty of the standard Kai font and most of the volunteers tend to write in an individual style unless they are reminded from time to time. Therefore, the first stage of recognition is to see the performance of the initial template database; in other words, how general is the initial template database. A summary of the recognition results for sample ORIG, RAND1, and RAND2 by using the initial template database is shown in Table 5.2.

Notice that ORIG has the highest average recognition rate (81.0%) among ORIG, RAND1 and RAND2. A possible explanation is that both RAND1 and RAND2 have redistributed stroke features to a certain degree such that the overall structure of

each character has been changed. Meanwhile, the *offset* for creating each local feature of a stroke is neither consistently positive nor consistently negative. As a result, the resulting stroke pattern may be very different from the standard handprinted writing style. This effect is even more obvious in recognizing RAND2 than recognizing RAND1.

In this experiment, 2% of these recognition results fall out of the first fifteen candidates list. The average computation time spent in the recognition process is 0.94 seconds/character for 133 reference characters in the 5,401 character set. This statistic result is based on the testing of 18,000 samples (ORIG, RAND1 and RAND2) from twenty writers. Compared to other systems with reported recognition rates ranging from 53% to 99% [4,7,12], the recognition results of ORIG (81%), RAND1 (80.7%), and RAND2 (80.1%) are above the average. Nevertheless, the recognition rate can be further improved by training those characters that were not recognized as the first candidate first time around. This selective training is based on the assumption that some of the initial template characters do not represent the writing styles used by most of the

TABLE 5.2

RECOGNITION RESULTS BY USING THE INITIAL TEMPLATE DATABASE

Sample Group		ORIG			RAND1			RAND2		
Test	Average Reference Characters	Recognition Rate (%)			Recognition Rate (%)			Recognition Rate (%)		
		1st	2nd	3rd	1st	2nd	3rd	1st	2nd	3rd
Test 1	135	81.6	90.3	92.7	81.6	90.1	92.7	80.7	89.9	92.9
Test 2	136	80.0	89.1	92.3	79.2	88.5	92.4	78.8	89.1	92.3
Test 3	129	81.3	90.5	93.7	81.4	90.0	93.6	80.7	90.3	93.1
Average	133	81.0	90.0	92.9	80.7	89.5	92.9	80.1	89.8	92.8

volunteers. This method was also used by Gader et al. [44] to yield higher reliability without a corresponding loss in recognition.

In Table 5.2, we have listed the recognition rates for characters that appeared in the first three candidates. Because of the writing variation and similarity among Chinese characters, it is reasonable to provide several candidates for writers to make their choice at the last stage. Therefore, if an input is considered correctly recognized in the first three candidates, the average recognition rate for ORIG, RAND1 and RAND2 is 92.9%.

5.6.2 Second Stage Recognition

In this experiment, we have selected those characters that were not recognized as the first candidates during the background recognition of ORIG. Characters that fall in

---

1. Start with any unprocessed set of characters that hold the same character code and the same stroke count. If there is no more unprocessed character set, stop.
2. Label the first character in the set as S.
3. Search for an unprocessed character in the set and label as T. If there is no more unprocessed character in the set, export S for appending and go to step 1. Otherwise, initialize the cost matrix $[c_{ij}]$ by the distance between a stroke in S and a stroke in T where the distance is defined as the summation of the normalized 1-norm corresponding to every local feature.
4. Apply the Hungarian Method (see Appendix A) to find the best match among strokes such that the sum of distances is minimum.
5. Verify the mapping by checking the compatibility of primitive types from the mapped stroke pairs. If any stroke pair does not satisfy this requirement, append character T to an unprocessed set of the same character category. Otherwise, update S to represent the mean distribution of the character. Remove T from the set and go to step 3.

Figure 5.11: Algorithm for obtaining the mean distribution of characters.

this scope are merged by character code to obtain the mean distribution of stroke features. Theoretically, if a better writing distribution can be found, the recognition rate can be improved. Figure 5.11 presents a basic and direct approach to obtain the mean distribution of stroke features. With this algorithm, the new template database can be customized for recognizing similar writing patterns collected. Without losing the general information from the initial templates, we have appended new template characters to observe the performance of the resulting database.

A summary of the recognition results for sample ORIG, RAND1, and RAND2 by using the new template database is shown in Table 5.3. About 0.4% of the recognition results fall out of the first fifteen candidates list. The average computation time spent in the recognition process is 1.1 seconds/character for 163 reference characters in the 5,401 character set.

By selective training from ORIG, the recognition rates are increased by 15% for ORIG, 14.7% for RAND1, and 12.9% for RAND2 as shown in Figure 5.12. These

TABLE 5.3

RECOGNITION RESULTS BY USING THE NEW TEMPLATE DATABASE

Sample Group		ORIG			RAND1			RAND2		
Test	Average Reference Characters	Recognition Rate (%)			Recognition Rate (%)			Recognition Rate (%)		
		1st	2nd	3rd	1st	2nd	3rd	1st	2nd	3rd
Test 1	164	95.4	98.0	98.5	94.7	97.6	98.4	92.4	96.5	97.6
Test 2	167	96.0	98.3	98.8	95.4	98.2	98.7	92.7	97.5	98.5
Test 3	158	96.7	98.7	99.2	96.0	98.5	99.0	93.8	98.0	98.6
Average	163	96.0	98.3	98.8	95.4	98.1	98.7	93.0	97.3	98.2

results have demonstrated the possibility of customization for the collected samples. In the experiment, the average recognition rate reaches 94.8%. Apart from those 0.4% that are ambiguous (fall out of the first fifteen candidates list), 4.8% are not in the first candidate. Possible causes include: (1) the trained samples may actually contain poorly written character input and (2) the encodings of several different template characters are very similar, resulting in a close match to the input. Improvement may focus on refining features selected, measure of similarity and algorithm for obtaining the mean distribution of characters.

In Table 5.3, we have also listed the recognition rates for characters that appeared in the first three candidates. Therefore, if an input is considered correctly recognized in the first three candidates, the recognition rate is 98.6%. This result has little difference in three testing sets.

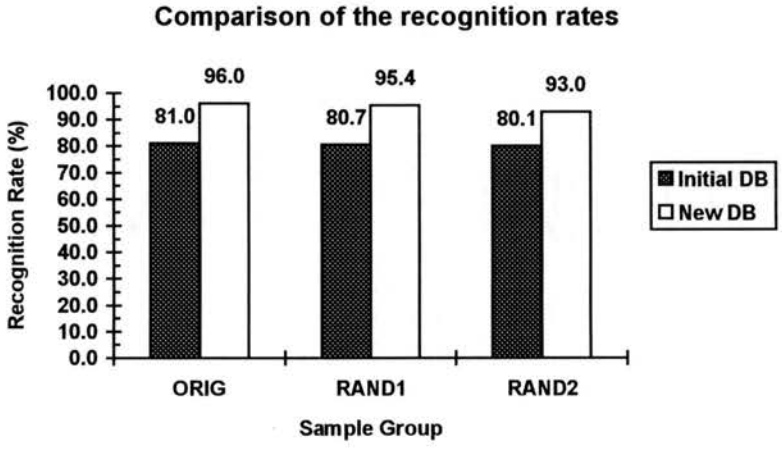


Figure 5.12: Comparison of the recognition results by using both initial template database and the new template database.

5.7 Summary

In this chapter, we have examined the implementation of both TRAINER and RECOGNIZER. Experimental procedures for creating an initial template database and collecting samples are also explained. Apparently, the size of the initial database is smaller than the estimated conventional approach, which has the advantage of loading fewer bytes per character. It also makes it possible for the application to load the database into primary memory for fast processing; particularly, for an environment where system resources are limited.

The average recognition rate for 18,000 collected and random samples from twenty writers of the initial template database is 80.6%. After tuning the system to the writers, the average recognition rate has reached 94.8%. As pointed out by Mori [4], it is difficult to compare recognition rates of different systems since the character categories and test samples are very different. To be more specific, the recognition rate of a system depends on three major factors: (1) size of the character set (number of categories in the database), (2) number of writers involved, and (3) selection of test samples. One might expect a higher recognition rate on a smaller character set than those on a large character set. This is because a large character set usually contains more similar characters. Furthermore, the more the writers, the more the writing styles may vary. The last is the selection of test samples. If the samples selected are distinct among reference characters, the recognition rate usually ends up higher rate than for those that are not.

Table 5.4 shows a summary of the system performance along with the feature(s) and matching method used. This table is organized by number of categories in the database, size of the database, number of categories tested, number of test samples and number of writers involved. For the recognition rate, the reported highest rate in the publication is given for the corresponding test set. At the bottom of the table, we also give the result of our experiments for the comparison.

TABLE 5.4

PERFORMANCE SUMMARY

Paper	Features	Matching methods	No. of cat. in DB	DB size (Kbytes)	No. of cat. tested	No. of test samples	No. of writers involved	Recognition rate (%)
[68]	Strokes	Dynamic programming	5,401	-	-	-	-	90
[69]	Strokes	Relaxation	14	-	14	651	-	96.3
[70]	Stroke distribution	Correlation	881	-	881	52,860	60	<95
[72]	Stroke density and cellular features	Exhausting matching	5,401	-	512	51,200	-	94.2
[74]	Line segments	Relaxation	881	278	881	140,960	-	99
[77]	Chain-code transform	Correlation	18	-	18	3,600	20	95.4
[80]	Strokes	Tree search	480	-	50	>500	10	90.5
[81]	Line segments	Dynamic programming	150	-	150	750	5	90
[83]†	Strokes	Bayesian decision rule	94	-	94	7,520	16	99.7
[84]	Strokes	Relaxation	-	-	240	8,880	37	95.3
[86]†	Stroke segments	Dynamic programming	2,500	-	180	1,800	10	94.5
[88]	Strokes	Tree search	100	-	100	1,000	10	90
[90]	Stroke segments	Iteration matching	465	-	465	930	-	96
[92]†	Strokes	Heuristic search	5,401	-	5,401	54,010	10	87.4
[93]	Strokes	Relaxation	270	-	155	255	-	98
[95]	Strokes	Tree search	50	-	50	350	7	90
[96]†	Strokes	Dynamic programming	5,400	-	5,400	54,000	10	87.4
[100]†	Radicals	Transformation invariant matching	114	-	114	114	-	99
[105]	Strokes	Fuzzy measure & linear matching	881	-	881	-	-	96
[106,107]†	Strokes and radicals	Fuzzy measure & incremental matching	-	-	240	8,980	-	91.8
[111]	Stroke junction features	Neural net	-	-	-	420	-	96.7
[112]	Pixels	Neural net	100	-	100	2,100	21	91
[113]	Pixels	Neural net	20	-	20	4,000	-	89
Proposed†	Strokes	Fuzzy measure & linear matching	5,401	112	300	18,000	20	94.8

†, On-line character recognition -, Data not available

CHAPTER VI

CONCLUSION AND FUTURE WORK

6.1 Summary

This dissertation presents a new stroke based representation scheme for handprinted Chinese characters. Owing to the complexity of each individual character and the size of the Chinese character set, our concern in designing a representation scheme is twofold; i.e., to incorporate as many features as possible and to reduce the size of the database file. These goals are accomplished by quantifying four distinct local features and packing them into one single byte to represent the local properties of a stroke. This has led to a major reduction in storage requirements for local features on the secondary storage. However, by losing a certain amount of information from the template, there is a risk that ambiguity between different templates may result. Therefore, we have selected ten groups of structural features to reduce the ambiguity among strokes.

To extract structural features, we have presented a primitive stroke extraction scheme. This scheme is simple and straight forward. The extraction procedure has successfully extracted ten groups of primitive strokes that are either formally taught in school or used by most people. Nevertheless, different writing habits may come up with different direction sequences. To overcome this problem, most of the primitive strokes

are embedded with several possible direction sequences. This gives us flexibility and ease in adding and modifying new primitive strokes.

To estimate the similarity between characters, we have defined fuzzy membership functions corresponding to various classifications for each local feature. The activation of a membership function is guided by the local properties of a template stroke. This process is also presented in canonical forms for measuring length, type, position, and direction individually. When the *intersection* operation is applied to the resulting similarity corresponding to each local feature, we can estimate the most dissimilar property between an input stroke to a template stroke. This result is further increased if the input stroke and the template stroke are of the same primitive stroke type or decreased if they are different. Finally, the degree of similarity from a handprinted character to a template character is calculated based upon the modified Hungarian Method for maximum assignment problem.

We have implemented a prototype using Microsoft Visual C++ under Microsoft Chinese Windows 3.1. Two graphic user interfaces, TRAINER and RECOGNIZER, are designed for and targeted to this Chinese environment under a personal computer. Several experiments have been conducted based upon the 18,000 collected and random samples for 300 character categories from twenty volunteers. Experiments include verification of local and structural encoding, creation of the initial template database, sample collection and evaluation of the recognition system.

At the end of the study, we have also evaluated the performance of the recognition results in two stages. In the first stage, the experiment is to see the performance of the recognition by using the initial template database created by direct

writing on the Kai font. In the experiment, 2% of the samples were reported ambiguous (i.e., out of the first fifteen candidates listed) in the first stage where the recognition rate was 81% on average. In the second stage, the ambiguity had reduced to 0.4% and the recognition rate was increased to 94.8% after selective training of the characters that were not recognized in the first stage. Overall, the average computation time spent in the recognition process is 0.94 seconds/character for 133 reference characters and 1.1 seconds/characters for 163 reference characters.

6.2 Contribution and Future Research

The major contribution of the design and implementation of the on-line handprinted Chinese character recognition system is finding a new approach to represent stroke based characters where the size of the character set is comparatively large. Compared with conventional methods, this new approach has resulted in a smaller database file (112 kbytes) for storing 5,401 Chinese characters. As in Figure 5.6, this stroke based representation scheme requires less storage than the conventional approach for storing similar amount of integer features. It also means that less physical bytes are required to retrieve from the secondary storage for each character. This may reduce the data retrieval time when a large and contiguous amount of information is required to be loaded from the secondary storage. Moreover, the size of the database file has made it possible for an application to load the entire file into primary memory for fast processing. This will result in better system performance concerning the time needed for processing. These facts have great advantages in an environment where the system resource, i.e., memory space, is limited.

Another distinguishing characteristic of this work is the design of the fuzzy membership functions where the similarity from an input stroke to a template stroke is guided by the packed template stroke. In addition, fuzzy approach preserves much of the spatial information, reduces size of representation and allows descriptive representation.

In addition to the refinements suggested during the design and implementation in the other chapters, further research may follow the following directions to improve the overall system performance.

(1) A thorough comparison of the preclassification schemes should be investigated to reduce the number of reference characters before the recognition process; particularly, schemes based on stroke order should be developed since every Chinese character is written in a standard stroke order or with a few changes from the standard order. If the stroke order can be applied, time spend in finding matching strokes can be further reduced during the measure of fuzzy similarity. Another possible approach is dividing the consecutive strokes into two parts; first part of strokes is used for coarse classification (reducing reference characters) and second part of strokes is used for matching. As a result, time for matching may be greatly reduced.

(2) Evaluation of the possibility of applying non-linear normalization is suggested as well. This is because non-linear normalization usually results in a better distribution of strokes than linear normalization does concerning local properties.

(3) The optimization of the initial database that represents most of the Chinese writing styles is another interesting direction. This may require the collection of samples and supervised training.

(4) The stability of features should be further investigated so that all features possess invariant characteristics.

BIBLIOGRAPHY

1. F.-H. Cheng and W.-H. Hsu, "Research on Chinese OCR in Taiwan," *Character & Handwriting Recognition - Expanding Frontiers*, Ed. P.S.P. Wang, World Scientific Series in Computer Science, Vol. 30, 139-164 (1991).

2. R. H. Davis and J. Lyall, "Recognition of Handwritten Characters - A Review," *Image and Vision Computing*, Vol. 4, No. 4, 208-218 (1986).

3. V. K. Govindan and A. P. Shivaprasad, "Character Recognition - A Review," *Pattern Recognition*, Vol. 23, No. 7, 671-683 (1990).

4. T. H. Hildebrandt and Wentai Liu, "Optical Recognition of Handwritten Chinese Characters: Advances since 1980," *Pattern Recognition*, Vol. 26, No. 2, 205-225 (1993).

5. S. Impedovo, L. Ottaviano, and S. Occhinegro, "Optical Character Recognition - A Survey," *Character & Handwriting Recognition - Expanding Frontiers*, Ed. P.S.P. Wang, World Scientific Series in Computer Science, Vol. 30, 1-24 (1991).

6. J. Mantas, "An Overview of Character Recognition Methodologies," *Pattern Recognition*, Vol. 19, No. 6, 425-430 (1986).

7. S. Mori, K. Yamamoto, and M. Yasuda, "Research on Machine Recognition of Handprinted Characters," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. PAMI-6, No. 4, 386-405 (1984).

8. G. Nagy, "Chinese Character Recognition: A twenty-five-year Retrospective," *9th International Conference on Pattern Recognition*, 14-17 November 1988, Rome, Italy, 163-167 (1988).

9. F. Nouboud and R. Plamondon, "On-line Recognition of Handprinted Characters: Survey and Beta Tests," *Pattern Recognition*, Vol. 23, No. 9, 1031-1044 (1990).

10. C. Y. Suen, M. Berthod, and S. Mori, "Automatic Recognition of Handprinted Characters - The State of the Art," *Proceedings of the IEEE*, Vol. 68, No. 4, 469-487 (1980).

11. J.-W. Tai, "Some Research Achievements on Chinese Character Recognition in China," *Character & Handwriting Recognition - Expanding Frontiers*, Ed. P.S.P. Wang, World Scientific Series in Computer Science, Vol. 30, 199-206 (1991).

12. C. C. Tappert, C. Y. Suen, and T. Wakahara, "The State of the Art in On-Line Handwriting Recognition," *IEEE Transactions of Pattern Analysis and Machine Intelligence*, Vol. 12, No. 8, 787-808 (1990).

13. W. C. P. Yu, H.-H. Teh, H.-B. Low, X. Yan, T.-M. Ng, and F. Gao, "Historical Advancement of the Chinese Language Computing," *Computer Processing of Chinese & Oriental Languages*, Vol. 4, No. 1, 57-81 (1988).

14. T. Kato and M. Yamada, "Quality Factors of Hand-written Characters Based on Human Visual Perception," *SPIE Vol. 1453 Human Vision, Visual Processing, and Digital Display II*, 43-50 (1991).

15. C. Y. Suen, "Distinctive Features in Automatic Recognition of Handprinted Characters," *Signal Processing*, 4, 193-207 (1982).

16. A. G esen, "Quantitative Analysis of Preprocessing Techniques for the Recognition of Handprinted Characters," *Pattern Recognition*, Vol. 8, 219-227 (1976).

17. R. Casey and G. Nagy, "Recognition of Printed Chinese characters," *IEEE Trans. Electron. Comput.*, Vol. EC-15, 91-101 (1966).

18. R. E. Bellman, R. Kalaba, and L. A. Zadeh, "Abstraction and Pattern Classification," *J. Math. Anal. Appl.*, 13, 1-7 (1966).

19. A. Kandel, "Fuzzy Techniques in Pattern Recognition," *A Wiley-Interscience Pub.*, New York (1982).

20. W. Pedrycz, "Fuzzy Sets in Pattern Recognition: Methodology and Methods," *Pattern Recognition*, Vol. 23, No. 1/2, 121-146 (1990).

21. L. A. Zadeh, "Fuzzy Sets," *Inf. Control*, 8, 338-353 (1965).

22. D. Dubois and H. Prade, "Fuzzy Sets and Systems: Theory and Applications," *Academic Press*, New York (1980).

23. H.-J. Zimmermann, "Fuzzy Set Theory and Its Applications," *Kluwer Academic Pub.*, Boston (1991).

24. G. J. Klir and T. A. Folger, "Fuzzy Sets, Uncertainty, and Information," *Prentice-Hall*, New York (1992).

25. F. Chang and S.-H. Lai, "Stroke Segmentation for Chinese Character Recognition," *First National Workshop on Character Recognition*, Taiwan, R.O.C., 1-13 (1991).

26. F. Chang, H.-S. Don, Y.-C. Chen, and W.-L. Hsu, "A Stroke Segmentation Algorithm with Applications to Chinese Character Recognition," *Second National Workshop on Optical Character Recognition*, Taipei, Taiwan, R.O.C., 44-57 (1992).

27. B.-S. Chien, B.-S. Jeng, S.-W. Sun, G.-H. Chang, K.-H. Shyu, and C.-S. Shih, "A Novel Block Segmentation and Processing for Chinese-English Document," *SPIE Vol. 1606 Visual Communications and Image Processing*, 588-598 (1991).

28. Y. P. Lan, B. S. Jeng, S. H. Lu, B. S. Chien, and M. W. Chang, "The Region and Recognition-Based Segmentation Method Used for Text in Mixed Chinese and English Characters," *SPIE Vol. 1199 Visual Communications and Image Processing IV*, 1332-1337 (1989).

29. S.-W. Sun and S. Y. Kung, "Flexible Segmentation and Matching for Optical Character Recognition," *SPIE Vol. 1199 Visual Communications and Image Processing IV*, 1314-1323 (1989).

30. Y.-S. Chen and W.-H. Hsu, "A Modified Fast Parallel Algorithm for Thinning Digital Patterns," *Pattern Recognition Letters*, 7, 99-106 (1988).

31. J. H. Sossa, "An Improved Parallel Algorithm for Thinning Digital Patterns," *Pattern Recognition Letters*, 10, 77-80 (1989).

32. J. Tsukumo and H. Tanaka, "Classification of Handprinted Chinese Characters Using Non-linear Normalization and Correlation Methods," *9th International Conference on Pattern Recognition*, Rome, Italy, 168-171 (1988).

33. H. Yamada, K. Yamamoto, and T. Saito, "A Nonlinear Normalization Method for Handprinted Kanji Character Recognition - Line Density Equalization," *9th International Conference on Pattern Recognition*, Rome, Italy, 1172-172 (1988).

34. T. Kumamoto, K. Toraichi, T. Horiuchi, K. Yamamoto, and H. Yamada, "On Speeding Candidate Selection in Handprinted Chinese Character Recognition," *Pattern Recognition*, Vol. 24, No. 8, 793-799 (1991).

35. J.-F. Wang and H.-D. Chang, "Preclassification for Handwritten Chinese Characters Recognition by a Partial Shape Coding Method," *First National Workshop on Character Recognition*, Taiwan, R.O.C., 14-32 (1991).

36. C.-H. Tung, H.-J. Lee, and J.-Y. Tsai, "Multi-Stage Pre-Candidate Selection in Handwritten Chinese Character Recognition Systems," *Pattern Recognition*, Vol. 27, No. 8, 1093-1102 (1994).

37. Y. X. Gu, Q. R. Wang, and C. Y. Suen, "Application of a Multilayer Decision Tree in Computer Recognition of Chinese Characters," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. PAMI-5, No. 1, 83-89 (1983).

38. J. S. Huang and M.-L. Chung, "Separating Similar Complex Chinese Characters by Walsh Transform," *Pattern Recognition*, Vol. 20, No. 4, 425-428 (1987).

39. R.-L. Lee and K.-R. Lu, "Multi-Feature Classification and Clustering Technique for Multi-Font Printed OCR," *First National Workshop on Character Recognition*, Taiwan, R.O.C., 33-42 (1991).

40. C. Nadal and C. Y. Suen, "Applying Human Knowledge to Improve Machine Recognition of Confusing Handwritten Numerals," *Pattern Recognition*, Vol. 26, No. 3, 381-389 (1993).

41. G. Boccignone, A. Chianese, L. P. Cordella, and A. Marcelli, "Recovering Dynamic Information From Static Handwriting," *Pattern Recognition*, Vol. 26, No. 3, 409-418 (1993).

42. E. Cohen, J. J. Hull, and S. N. Srihari, "Understanding Handwritten Text in a Structured Environment: Determining ZIP Codes from Address," *Character & Handwriting Recognition - Expanding Frontiers*, Ed. P.S.P. Wang, World Scientific Series in Computer Science, Vol. 30, 221-264 (1991).

43. A. C. Downton, R. W. S. Tregidgo, and E. Kabir, "Recognition and Verification of Handwritten and Hand-Printed British Postal Addresses," *Character & Handwriting Recognition - Expanding Frontiers*, Ed. P.S.P. Wang, World Scientific Series in Computer Science, Vol. 30, 265-292 (1991).

44. P. Gader, B. Forester, M. Ganzberger, A. Gillies, B. Mitchell, M. Whalen, and T. Yocum, "Recognition of Handwritten Digits Using Template and Model Matching," *Pattern Recognition*, Vol. 24, No. 5, 421-431 (1991).

45. P. P. van der Smagt, "A Comparative Study of Neural Network Algorithms Applied to Optical Character Recognition," *Proc. of the 3rd International Conference on Industrial and Engineering Applications of Artificial Intelligence and Expert Systems* (IEA/AIE), 1037-1044 (1990)

46. J.-F. Wang and G.-E. Wang, "A New Approach for Recognition of Unconstrained Handwritten Numerals," *Second National Workshop on Optical Character Recognition*, Taipei, Taiwan, R.O.C., 159-183 (1992).

47. T. Fujisaki, T. E. Chefalas, J. Kim, C. C. Tappert, and C. G. Wolf, "On-Line Run-On Character Recognizer: Design and Performance," *Character & Handwriting Recognition - Expanding Frontiers*, Ed. P.S.P. Wang, World Scientific Series in Computer Science, Vol. 30, 123-138 (1991).

48. M. Armmar, "Progress in Verification of Skillfully Simulated Handwritten Signatures," *Character & Handwriting Recognition - Expanding Frontiers*, Ed. P.S.P. Wang, World Scientific Series in Computer Science, Vol. 30, 337-352 (1991).

49. P. S. P. Wang and A. Gupta, "An Improved Structural Approach for Automated Recognition of Handprinted Characters," *Character & Handwriting Recognition - Expanding Frontiers*, Ed. P.S.P. Wang, World Scientific Series in Computer Science, Vol. 30, 97-122 (1991).

50. F. Nouboud and R. Plamondon, "A Structural Approach to On-Line Character Recognition: System Design and Applications," *Character & Handwriting Recognition - Expanding Frontiers*, Ed. P.S.P. Wang, World Scientific Series in Computer Science, Vol. 30, 311-336 (1991).

51. K. Nakata, Y. Nakano, and Y. Uchikura, "Recognition of Chinese Characters," *Proc. Conf. Machine Perception of Patterns of Pictures*, 45-52 (1972).

52. P. P. Wang and R. C. Shiau, "Machine Recognition of Printed Chinese Characters via Transformation Algorithms," *Pattern Recognition*, Vol. 5, 303-321 (1073).

53. J. T. and K. Asai, "Machine Printed Chinese and Japanese Character Recognition Method and Experiments for Reading Japanese Pocket Books," *IEEE Computer Vision and Pattern Recognition*, Miami Beach, Florida, 162-167 (1986).

54. P.-N. Chen, Y.-S. Chen, and W.-H. Hsu, "Stroke Relation Coding - A New Approach to the Recognition of Multi-Font Printed Chinese Characters," *International Journal of Pattern Recognition and Artificial Intelligence*, Vol. 2, No. 1, 149-160 (1988).

55. B.-S. Jeng, C.-H. Shih, S.-W. Sun, T.-M. Wu, B.-S. Chien, and M.-W. Chang, "On the Use of Discrete-State Process for Chinese Character Recognition, " *SPIE Vol. 1360 Visual Communications and Image Processing*, 1663-1670 (1990).

56. S. Zhang, B. Taconet, and A. Faure, "A Complexity Measure Based Algorithm," *10th International Conference on Pattern Recognition*, Atlantic City, New Jersey, U.S.A., 573-577 (1990).

57. J. S. Huang and P.-M. Huang, "Machine Printed Chinese Character Recognition based on Linear Regression," *Character & Handwriting Recognition - Expanding Frontiers*, Ed. P.S.P. Wang, World Scientific Series in Computer Science, Vol. 30, 165-174 (1991).

58. C.-J. Wu and W.-H. Tsai, "A Decision-Tree Approach to Recognition of Printed Chinese Characters by Neural Networks Using Neocognitions," *First National Workshop on Character Recognition*, Taiwan, R.O.C., 76-106 (1991).

59. M. Zhang, C. Y. Suen, and T. D. Bui, "Multi-Font Chinese Character Recognition with Associative Memory Network," *Artificial Neural Networks*, Ed. T. Kohonen, K. Makisara, O. Simula and J. Kangas, Elsevier Science Publishers, B. V. (North-Holland), 1199-1202 (1991).

60. K.-C. Fang and Y.-K. Wang, "Sparse Distributed Memory for Use in Recognition Handwritten Character," *First National Workshop on Character Recognition*, Taiwan, R.O.C., 147-155 (1991).

61. H. Fujihara, E. Babiker, and D. B. Simmons, "Fuzzy Approach to Document Recognition," *Second IEEE International Conference on Fuzzy Systems*, 980-985 (1993).

62. P. Siy and C. S. Chen, "Fuzzy Logic for Handwritten Numeral Character, "Fuzzy Logic for Handwritten Numeral Character Recognition," *IEEE Transactions on System, Man, and Cybernetics*, 570-575 (1974).

63. W. J. M. Kickert and H. Koppelaar, "Application of Fuzzy Set Theory to Syntactic Pattern Recognition of Handwritten Capitals," *IEEE Transactions on System, Man, and Cybernetics*, 148-151 (1976).

64. J. Y. Tsai, "Speed Improvement on Coarse Classification of Handprinted Chinese Characters with Large Category Set," Master Thesis, National Chiao Tong University, CRENG, Pt. 4, Taiwan (1992).

65. K. Ikeda, T.Yamamura, Y. Mitamura, S. Fujiwara, Y. Tominaga, and T. Kiyono, "On-Line Recognition of Hand-Written Characters Utilizing Position and Stroke Vector Sequences," *Pattern Recognition*, Vol. 13, No. 3, 191-206 (1981).

66. Y. Liu and T. Kasvand, "A New Approach to Machine Recognition of Chinese Characters," *Proceedings of the 7th International Conference on Pattern Recognition*, Montreal, Canada, 381-384 (1984).

67. S. L. Xie and M. Suk, "On Machine Recognition of Hand-Printed Chinese Characters by Feature Relaxation," *Pattern Recognition*, Vol. 21, No.1, 1-7, (1988).

68. B.-S. Jeng, "Optical Chinese Character Recognition Using Accumulated Stroke Features," *Optical Engineering*, Vol. 28, No. 7, 793-799 (1989).

69. L.-H. Chen and J.-R. Lieh, "Handwritten Character Recognition Using a 2-Layer Random Graph Model by Relaxation Matching," *Pattern Recognition*, Vol. 23, No. 11, 1189-1205 (1990).

70. Y. Yamashita, K. Higuchi, Y. Yamada, and Y. Haga, "Classification of Handprinted Kanji characters by the Structured Segment Matching Method," *Pattern Recognition Letters*, 1, 475-4479 (1983).

71. M.-Y. Chen, W.-H. Hsu, and F.-H. Cheng, "An Application of Hough Transform to Recognition of Handwritten Chinese Characters," *Proceedings of International Computer Symposium*, Taiwan, R.O.C., 719-726 (1986).

72. L.-T. Tu, Y.-S. Lin, C.-P. Yeh, I.-S. Shyu, J. L. Wang, K.-H. Joe, and W.-W. Lin, "Recognition of Handprinted Chinese Characters by Feature Matching," *First National Workshop on Character Recognition*, Taiwan, R.O.C., 166-175 (1991).

73. T. F. Li, S.-S. Yu, H.-F. Sun, and S.-L. Chou, "Recognition of Handwritten Chinese Character by Using Bayes Rule," *Second National Workshop on Optical Character Recognition*, Taipei, Taiwan, R.O.C., 31-43 (1992).

74. K. Yamamoto and A. Rosenfeld, "Recognition of Hand-Printed Kanji Characters by A Relaxation Method," *Proceedings of the 6th International Joint Conference on Pattern Recognition*, 395-398 (1982).

75. K. Yamamoto, H. Yamada, T. Saito, and R.-I. Oka, "Recognition of Handprinted Chinese Characters and Japanese Cursive Syllabary," *Proceedings of the 7th International Conference on Pattern Recognition*, Montreal, Canada, 385-388 (1984).

76. H. Yamada, "Contour DP Matching Method and Its Application to Handprinted Chinese Character Recognition," *Proceedings of the 7th International Conference on Pattern Recognition*, Montreal, Canada, 389-392 (1984).

77. Y. S. Cheung and C. H. Leung, "Cain-Code Transform for Chinese Character Recognition," *IEEE Proceedings of the International Conference on Cybernetics and Society*, 42-45 (1985).

78. K. Yamamoto, H. Yamada, T. Saito, and I. Sakaga, "Recognition of Handprinted Characters in the First Level of JIS Chinese Characters," *Proceedings of the 8th International Conference on Pattern Recognition*, Paris, France, 570-572 (1986).

79. Y.-T. Tsay and W.-H. Tsai, "Model-Guided Attributed String Matching by Split-and Merge for Shape Recognition and On-Line Chinese Character Recognition," Ph.D. Dissertation, National Chiao Tung University, Taiwan, R.O.C., CRENG, Pt. 7 (1989).

80. S. W. Lu, Y. Ren, and C. Y. Suen, "Hierarchical Attributed Graph Representation and Recognition of Handwritten Chinese Characters," *Pattern Recognition*, Vol. 24, No. 7, 617-632 (1991).

81. H.-J. Lee and B. Chen, "Recognition of Handwritten Chinese Characters via Short Line Segments," *Pattern Recognition*, Vol. 25, No. 5, 543-552 (1992).

82. K. Odaka, H. Arakawa, and I. Masuda, "On-Line Recognition of Handwritten Characters by Approximation Each Stroke with Several Points," *IEEE Transactions on Systems, Man, and Cybernetics*, Vol. SMC-12, No. 6, 898-903 (1982).

83. H. Arakawa, "On-Line Recognition of Handwritten Characters - Alphanumeric, Hiragana, Katakana, Kanji," *Pattern Recognition*, Vol. 16, No. 1, 9-16 (1983).

84. C. H. Leung, Y. S. Cheung, and Y. L. Wong, "A Knowledge-Based Stroke-Matching Method for Chinese Character Recognition," *IEEE Transactions on Systems, Man, and Cybernetics*, Vol. SMC-17, No. 6, 993-1003 (1987).

85. K.-J. Chen, K.-C. Li, and Y.-L. Chang, "A System for On-Line Recognition of Chinese Characters," *Computer Processing of Chinese & Oriental Languages*, Vol. 3, No. 3 & 4, 309-318 (1988).

86. C.-K. Lin, B.-S. Jeng, and C.-J. Lee, "Stroke-Order Independent On-Line Recognition of Handwritten Chinese Characters," *SPIE Vol. 1199 Visual Communications and Image Processing IV*, 1338-1344 (1989).

87. Y. Xia and C. Sun, "Recognizing Restricted Handwritten Chinese Characters by Structure Similarity Method," *Pattern Recognition Letters*, 11, 67-73 (1990).

88. C.-C. Hsieh and H.-J. Lee, "Off-Line Recognition of Handwritten Chinese Characters by On-Line Model-Guided Matching," *Pattern Recognition*, Vol. 0025, No. 11, 1337-1352 (1992).

89. C.-C. Hsieh, "Model-Guided Recognition of Handwritten Chinese Characters," Ph.D. Dissertation, National Chiao Tung University, Taiwan, R.O.C., CRENG, Pt. 3 (1992).

90. S.-L. Chou and W.-H. Tsai, "Recognizing Handwritten Chinese Characters by Stroke-Segment Matching Using an Iteration Scheme," *Character & Handwriting Recognition - Expanding Frontiers*, Ed. P.S.P. Wang, World Scientific Series in Computer Science, Vol. 30, 175-198 (1991).

91. S.-L. Chou and W.-H. Tsai, "On-Line Chinese Character Recognition through Stroke-Segment Matching Using a New Discrete Iteration Scheme," *Computer Processing of Chinese and Oriental Languages*, Vol. 7, No. 1, 1-20 (1993).

92. C.-K. Lin, B.-S. Jeng, T.-K. Su, and K.-C. Fan, "On-Line Chinese Character Recognition by Artificial Intelligence Based Method," *Second National Workshop on Optical Character Recognition*, Taipei, Taiwan, R.O.C., 1-11 (1992).

93. A.-B. Wang, J. S. Huang, and K.-C. Fan, "Optical Recognition of Handwritten Chinese Characters by Modified Relaxation," *Second National Workshop on Optical Character Recognition*, Taipei, Taiwan, R.O.C., 115-157 (1992).

94. F.-H. Cheng, W.-H. Hsu, and M.-C. Kuo, "Recognition of Handprinted Chinese Characters via Stroke Relaxation," *Pattern Recognition*, Vol. 26, No. 4, 579-593 (1993).

95. R. I. Chou, A. Kershenbaum, and E. K. Wong, "Representation and Recognition of Handprinted Chinese Characters by String-Matching," *Information Sciences*, 67, 1-34 (1993).

96. C.-K. Lin, K.-C. Fan, and F. T.-P. Lee, "On-Line Recognition by Deviation-Expansion Model and Dynamic Programming Matching," *Pattern Recognition*, Vol. 26, No. 2, 259-268 (1993).

97. E. F. Yhap and E. C. Greanias, "An On-Line Chinese Character Recognition System," *IBM J. Res. Develop.*, Vol. 25, No. 3, 187-195 (1981).

98. F.-H. Cheng and W.-H. Hsu, "Radical Extraction by Background Thinning Method for Handwritten Chinese Characters," *International Conference on Chinese and Oriental Language Computing, Chicago*, Illinois, U.S.A., 175-182 (1987).

99. F.-H. Cheng and W.-H. Hsu, "Radical Extraction from Handwritten Chinese Characters by Background Thinning Method," *The Transaction of the IEICE*, Vol. E 71, No. 1, 88-98 (1988).

100. C.-W. Liao and J. S. Huang, "A Transformation Invariant Matching Algorithm for Handwritten Chinese Character Recognition," *Pattern Recognition*, Vol. 23, No. 11, 1167-1188 (1990).

101. F.-H. Cheng and W.-H. Hsu, "Three Stroke Extraction Methods for Recognition of Handwritten Chinese Characters," *International Conference on Chinese Computing*, 191-195 (1986).

102. J.-W. Tai, "A Syntactic-Semantic Approach," *Proceedings of the 7th International Conference on Pattern Recognition*, Montreal, Canada, 374-376 (1984).

103. M. Zhao, "2-D EAG Method for the Recognition of Hand-Printed Chinese Characters," *Journal of Computer Science and Technology*, Vol. 5, No. 4, 319-328 (1990).

104. F.-H. Cheng and W.-H. Hsu, "Fuzzy Recognition of Handwritten Chinese Characters," *International Conference on Computer Processing of Chinese and Oriental Languages*, Toronto, Canada, 28-34 (1988).

105. F.-H. Chen, W.-H. Hsu, and C.-A. Chen, "Fuzzy Approach to solve the Recognition Problem of Handwritten Chinese Characters," *Pattern Recognition*, Vol. 22, No. 2, 133-141 (1989).

106. K.-P. Chan and Y.-S. Cheung, "Fuzzy-Attributed Graph and its Application to Chinese Character Recognition," *Computer Processing of Chinese & Oriental Languages*, Vol. 4, No. 2 & 3, 85-98 (1989).

107. K.-P. Chan and Y.-S. Cheung, "Fuzzy-Attributed Graph with Application to Chinese Character Recognition," *IEEE Trans. Syst. Man Cybern.*, Vol. 22(1), 153-160 (1992).

108. J.-C. Cheng and D.-C. Tseng, "Handwritten Chinese Character Recognition Using Fuzzy Weighted Ring-Data," *Second National Workshop on Optical Character Recognition*, Taipei, Taiwan, R.O.C., 92-114 (1992).

109. M. T. M. Gary and C. H. P. Joe, "A Fuzzy Attributed Graph Approach to Handwritten Character Recognition," *Second IEEE International Conference on Fuzzy Systems*, 570-575 (1993).

110. Y. Yong, "Handprinted Chinese Character Recognition via Neural Networks," *Pattern Recognition Letters*, 7, 19-25 (1988).

111. W.-Y. Su and W.-H. Tsai, "Handprinted Chinese Character Recognition by Neural Networks Using Modified Neocognitron," *Second National Workshop on Optical Character Recognition*, Taiwan, R.O.C., 58-80 (1992).

112. C.-J. Lee, B.-S. Jeng, S.-W. Sun, T.-M. Wu, and Y.-H. Hu, "A Neural Nets Classifier for Chinese Character Recognition," *Second National Workshop on Optical Character Recognition*, Taiwan, R.O.C., 81-91 (1992).

113. T. H. Hildebrandt, "Optical Recognition of Handprinted Chinese Characters Using an Improved Neural Network Model," Ph.D. Dissertation, North Carolina State University, U.S.A. (1991).

114. L. Y. Tseng and T. H. Huang, "Recognition of Hand-Printed Chinese Characters Based on the Backpropagation Neural Network," *Computer Processing of Chinese and Oriental Languages*, Vol. 7, No.1, 95-110 (1993).

115. B.-S. Jeng, S.-W. Sun, C.-J. Lee, K.-H. Shyu, F.-H. Liu, and T.-M. Wu, "Clustering and Classification for Chinese Character Recognition," *SPIE Vol. 1199 Visual Communications and Image Processing IV*, 1324-1331 (1989).

116. E. D. Nering and A. W. Tucker, "Linear Programs and Related Problems," Computer Science and Scientific Computing, Academic Press, New York, 1993.

117. D. König, "Graphen und Matrizen," *Mathematikai és Fizidai Lápok* 38, 116-119 (1931).

118. C.-H. Tung and H.-J. Lee, "Increasing Character Recognition Accuracy by Detection and Correction of Erroneously Identified Characters," *Pattern Recognition*, Vol. 27, No. 9, 1259-1266 (1994).

119. T.-Z. Lin and K.-C. Fan, "Coarse Classification of On-line Chinese Characters via Structure Feature-Based Method," *Pattern Recognition*, Vol. 27, No. 10, 1365-1377 (1994).

120. A. J. Pettofrezzo and M. M. Lacatena, "Analytic Geometry with Vectors," Scott, Foresman and Company, Illinois, 1970.

APPENDIXES

APPENDIX A

THE HUNGARIAN METHOD

Formulation of the Assignment Problem

The Hungarian method introduced herein is based on Kuhn's Hungarian Algorithm presented in Nering [116]. There is no reason why an assignment problem should be formulated for minimization but not for maximization. Nevertheless, as the assignment model is usually referred to as minimizing the sum of costs, it is reasonable to consider that the general assignment problem is for minimization rather than for maximization. Thus, suppose we are given an n x n cost matrix $C = [c_{ij}]$, the general assignment can be formulated as to minimize

$$Z = \sum_{(i,j)} c_{ij} x_{ij} \tag{A.1}$$

subject to

$$\sum_{i=1}^{n} x_{ij} = 1 \quad \text{for all } j \tag{A.2}$$

and

$$\sum_{j=1}^{n} x_{ij} = 1 \quad \text{for all } i \tag{A.3}$$

where x_{ij} is equal to 0 or 1.

Formulation of the Dual Problem

The general assignment problem in the previous section can be solved efficiently by the algorithm developed by the Hungarian mathematician D. König [117] in 1931. In honor of his contribution, this method is usually referred as the Hungarian Method. The foundation of the Hungarian Method is the duality theorem, whose formulation is described based on the results presented in Nering [116].

Suppose we are given a general assignment problem. The objective is to select one c_{ij} from each row and each column such that the sum of c_{ij} is the least. Introduce a variable u_i for each row i and a variable v_j for each column j and define

$$w_{ij} = c_{ij} - \left(u_i + v_j\right) \tag{A.4}$$

such that w_{ij} is nonnegative for every i and j. Here, the value of w_{ij} is called a reduced cost and $[w_{ij}]$ is called a reduced cost matrix. Thus, from Eqs A.1 to A.4 we have

$$\begin{aligned} \sum_{(i,j)} c_{ij}x_{ij} &= \sum_{(i,j)} u_i x_{ij} + \sum_{(i,j)} v_j x_{ij} + \sum_{(i,j)} w_{ij} x_{ij} \\ &= \sum_{i=1}^{n} u_i + \sum_{j=1}^{n} v_j + \sum_{(i,j)} w_{ij} x_{ij} \end{aligned} \tag{A.5}$$

Since x_{ij} and w_{ij} are both nonnegative, we have the following inequality

$$\sum_{(i,j)} c_{ij}x_{ij} \geq \sum_{i=1}^{n} u_i + \sum_{j=1}^{n} v_j . \tag{A.6}$$

Eq A.6 introduces the dual of the assignment problem; i.e., find u_i and v_j that maximize $\sum_{i=1}^{n} u_i + \sum_{j=1}^{n} v_j$ subject to the condition $u_i + v_j \leq c_{ij}$ for all i, j. Clearly, the feasible solutions of the dual problem are bounded by the feasible solutions of the

minimal assignment problem on the left side of the inequality. If the feasible solutions to both problems are equal, then both solutions are optimal; i.e.,

$$\sum_{(i,j)} c_{ij} x_{ij} = \sum_{i=1}^{n} u_i + \sum_{j=1}^{n} v_j \ . \tag{A.7}$$

This implies that both solutions are optimal if and only if

$$\sum_{(i,j)} w_{ij} x_{ij} = 0 \ . \tag{A.8}$$

Since x_{ij} and w_{ij} are both nonnegative, Eq A.8 will be true if and only if either x_{ij} or w_{ij} is zero. In the next section, we will present the Hungarian algorithm that deals with both the minimal assignment problem and the dual problem stated above.

Kuhn's Hungarian Algorithm

1. Start with any set of u_i, v_j and $w_{ij} \geq 0$ that satisfies Eq A.4. A good start is presented in Nerling [116] by setting

$$\begin{aligned} u_i &= \min_j c_{ij} \\ v_j &= \min_i (c_{ij} - u_i) \\ w_{ij} &= c_{ij} - (u_i + v_j) \end{aligned} \ . \tag{A.9}$$

2. Find a maximal independent set of zeros and a minimal cover over the zeros in matrix $[w_{ij}]$. A set of entries from matrix $[w_{ij}]$ is said to be independent if no two entries are in the same row or column. A cover for the set of zeros in a cost matrix is a set of vertical and horizontal lines that cover every zero in that matrix. Eventually, there could be many possible independent sets and minimal covers over the zeros in a matrix. If the maximal independent set of zeros is less than the size of the cost matrix (say n), go to step 3. Otherwise, stop and the optimum solution has been found.

3. Let e be the smallest entry of matrix $[w_{ij}]$ that is not covered by a line. Change u_j to u_j-e for i in the cover. Change v_j to v_j+e for j not in the cover. Recompute the w_{ij} by Eq A.4 and go to step 1.

In the previous section, the operation in step 2 is actually König's theorem. The theorem states that the number of zeros in a largest independent set of zeros is equal to the number of lines in a smallest cover of the zeros. The algorithm for the theorem is described in the following. Meanwhile, the algorithm is called the König's algorithm to distinguish from the main procedure in the previous section.

The König's Algorithm

1. Start with a complete set of starred zeros. A complete set of starred zeros represents an independent set of zeros that cannot be further expanded to a larger independent set. A good start is to star the first zero in the first row. In each subsequent row, star the first zero that is not in a column of a previously starred zero. Draw a horizontal line that covers each starred zero. Label all uncovered rows with a 0 -- row label.
2. Search for an uncovered row (row labeled as 0) with an uncovered zero. If there is an uncovered zero, label the column with the index of the uncovered row -- column label. Go to step 3.

   If there exists no uncovered zero, stop and return the number of lines in the cover or the number of independent starred zeros.
3. In the column labeled in step 2, search for a starred zero. If there is no starred zero in that column, we have found an alternate way to star zeros. This phenomenon is

called *breakthrough*. *Breakthrough* will result in an increase of one in the number of starred zeros. Go to step 4.

If there is a starred zero in that column, label the row that contains the starred zero with the index of the column -- row label. Remove the horizontal line that covers the starred zero; draw a vertical line through the starred zero. Go to step 2.

4. Trace the path by going to the entry identified by a row label and a column label until a row label 0 is reached. Star each unstarred zero and unstarred every starred zero along the path. This will produce a new independent set of starred zeros with one more element. Erase all labels and lines (cover). Go to step 1.

APPENDIX B

GRAPHIC USER INTERFACE

Program Description

Two collections of programs and data files were used in the experiment of the on-line Chinese character recognition system. The first collection consists of the program and data files related to the graphic user interface (GUI) -- TRAINER. The second collection contains the program and data files for the interface RECOGNIZER. Figure B.1 presents both collections and individual file types and descriptions.

File Name	Type	Description
LEARN.EXE	Executable	Executable file to invoke GUI TRAINER
PST.TXT	Text	Primitive strokes and direction sequences

(a) TRAINER

File Name	Type	Description
HCCR.EXE	Executable	Executable file to invoke GUI RECOGNIZER
PST.TXT	Text	Primitive strokes and direction sequences
HEADER.IDX	Binary	First level index file -- grouped by stroke count
RECORD.IDX	Binary	Second level index file -- grouped by projection profiles
DATABASE.OCR	Binary	Encoded template database file
SAMPLE1.BIG	Binary	Sample file 1 -- 100 characters
SAMPLE2.BIG	Binary	Sample file 2 -- 100 characters
SAMPLE3.BIG	Binary	Sample file 3 -- 100 characters

(b) RECOGNIZER

Figure B.1: Description of programs and data files.

TRAINER

TRAINER consists of several interfaces that allow the user to invoke commands or convey the information back to the user. The menu bar, toolbar and command bar allow users to invoke pre-defined command operations; status bar, writing window, approximation window and encoding window echo the status and results of the writing. Figure B.2 shows the TRAINER and its related components.

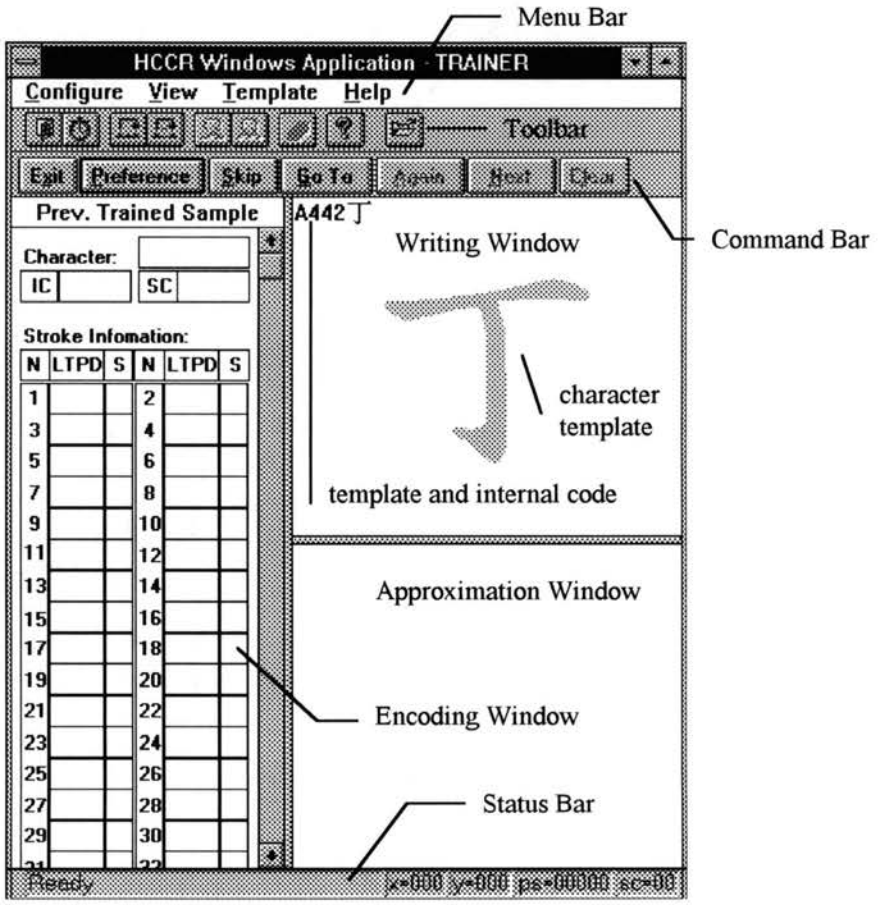


Figure B.2: TRAINER and its components.

Menu Bar Functions

The menu bar in the TRAINER contains pull-down menus for three groups of menu options: Configure, View and Sample. The options available in these menus are described in the following sections.

Configure Menu Options

(1) Preference: This option allows the user to initialize and save preferences corresponding to the display during writing experiments. Figure B.3 presents the dialog box for selecting the user preference for the simulation. In this dialog box, the user can select the processing mode (manual or automatic), viewing options, pen width, template size and point of elimination for the experiment.

(2) Exit: This option will terminate the execution of TRAINER.

View Menu Options

(1) Toolbar: This option allows the user to change the display of the toolbar by checking or unchecking the toolbar option. When this option is checked, toolbar will be displayed.

(2) Status Bar: This option allows the user to change the display of the status bar at the bottom of TRAINER.

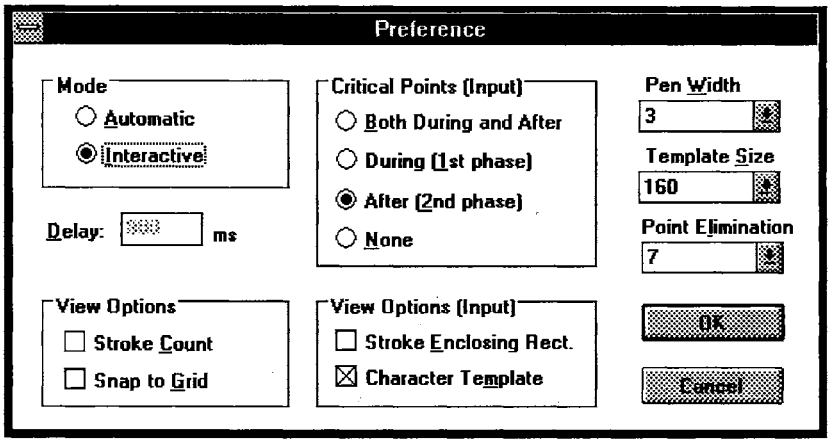


Figure B.3: Preference dialog box under TRAINER.

Sample Menu Options

(1) Skip: This option allows the user to ignore the character currently under training and go to the next character in order.

(2) GoTo: This option presents the dialog box as shown in Figure B.4 for the user to go to a specific character by selecting the internal character code.

(3) Encode Again: This option allows the user to encode the present character template again.

(4) Encode Next: This option encodes the present character template and goes to

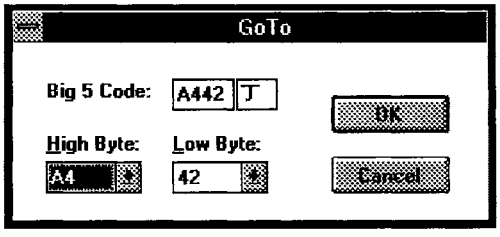


Figure B.4: GoTo dialog box.

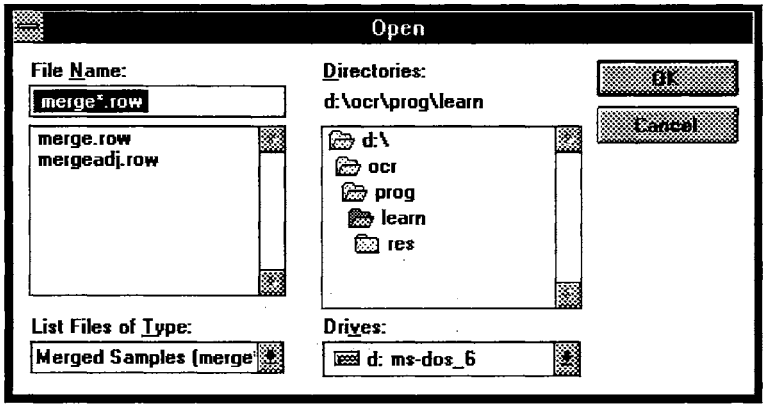


Figure B.5: Dialog box for file selection.

the next character template.

(5) Clear: This option clears the writing in the writing window.

(6) Encode from File: This option allows the user to encode row character data from a specific file selected from the dialog box shown in Figure B.5.

Toolbar Functions

Between menu bar and command bar, there is a collection of toolbar button functions. These button functions allow the user to perform commonly used operations in TRAINER. Since they perform the same operations as those menu options in menu bar, toolbar buttons are labeled to indicate related menu options (see Figure B.6).

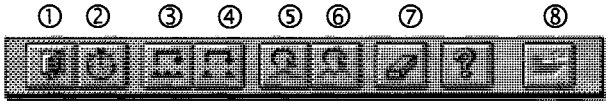


Figure B.6: Toolbar button functions and their relative menu options. ①: Exit; ②: Preference; ③: Skip; ④: GoTo; ⑤: Encode Again; ⑥: Encode Next; ⑦: Clear; ⑧: Encode from File.

Command Bar Functions

Below the toolbar, there is a collection of command button functions. These button functions allow the user to perform commonly used operations in TRAINER. They are self-explanatory; they also perform the same functions as those menu options in menu bar.

Status Bar

Status bar displays corresponding information to echo user commands or input. This information includes messages from the main window, x-y coordinates of the pointing device in the writing zone, current direction sequence, and stroke count of the writing. Figure B.7 shows the components of the status bar.

Writing Window

This is a child window inside TRAINER. It is the only place where the user can write. When the writing window is active, a pen style cursor is displayed. Otherwise, either an arrow cursor or a stop cursor is displayed. On the upper-left corner of the

Figure B.7: Status bar and its components. ①: message pane; ②: x coordinate of the pointing device; ③: y coordinate of the pointing device; ④: direction sequence; ⑤: stroke count.

wiring zone, an actual training template and its internal code are displayed (see Figure B.2). If the character template option is checked in the preference dialog box, a character template is presented to the user close to the center of the writing window with a specified size.

Approximation Window

This is a client window of the writing window. For each stroke written in the writing window, the line approximation is displayed in the approximation window. If the stroke count option is selected, strokes are labeled in order.

Encoding Window

Encoding window presents the local and structural encoding for the previous training sample. The information encoded includes: character internal code, stroke count and length, type, position, direction, and primitive stroke type of each stroke. Figure B.8 shows the encoded information for the character 丁.

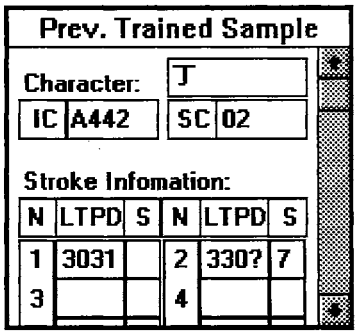


Figure B.8: Encoding window and encoded character 丁.

RECOGNIZER

RECOGNIZER consists of several interfaces that allow the user to invoke commands or convey information back to the user. Menu bar, toolbar and command bar allow users to invoke pre-defined command operations; status bars, writing window, normalization window and result window echo the status and results of the writing. Figure B.9 shows the RECOGNIZER and its related components.

Menu Bar Functions

The menu bar in the RECOGNIZER contains pull-down menus for three groups of menu options: Configure, View and Input. The options available on these menus are described in the following sections.

Configure Menu Options

(1) Preference: This option allows the user to initialize and save preferences corresponding to the display during the writing. Figure B.10 presents the dialog box to select user preferences for the simulation. In this dialog bar, the user can select the processing mode (manual or automatic), pen width, grid lines and filters (stroke count, number of candidates and projection profiles).

(2) Exit: This option will terminate the execution of RECOGNIZER.

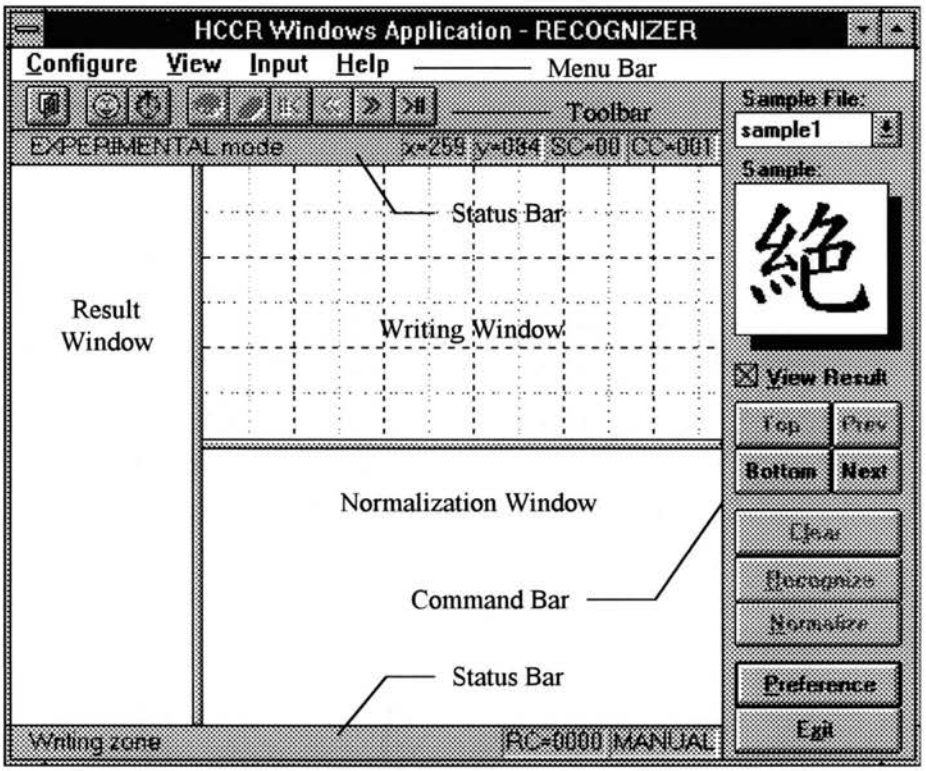


Figure B.9: RECOGNIZER and its components.

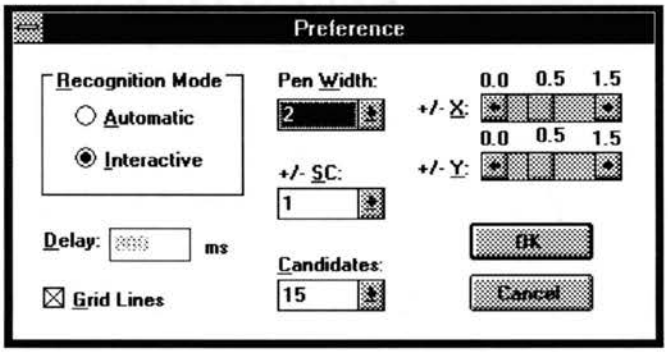


Figure B.10: Preference dialog box under RECOGNIZER.

Input Menu Options

(1) Recognize: This option performs recognition of the input.

(2) Normalize: This option performs normalization of the input without recognition.

(3) Clear: This option clears the writing in the writing window.

(4) Record: This option has four sub-options that allow the user to do the following operations to the sample records in the file; i.e., Top, Next, Previous and Bottom.

(5) File: This option presents a dialog box similar to Figure B.4 for the user to select an input row data file for the background recognition process.

Toolbar Functions

Toolbar contains a collection of button functions that allows the user to perform commonly used operations in RECOGNIZER. Since most of the button functions perform the same operations those menu options in the menu bar, toolbar buttons are labeled to indicate related menu options (see Figure B.11).

Command Bar Functions

On the right side of the main window, there is a collection of command bar button functions (see Figure B.9). These button functions allow the user to perform commonly used operations during the collection of samples in the experiment. Most of

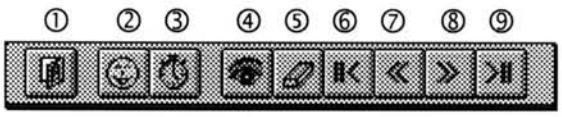


Figure B.11: Toolbar button functions and their related menu options.
①: Exit; ②: Mode Selector (REAL or EXPERIMENTAL);
③: Preference; ④: Recognize; ⑤: Clear; ⑥: Top; ⑦: Previous;
⑧: Next; ⑨: Bottom.

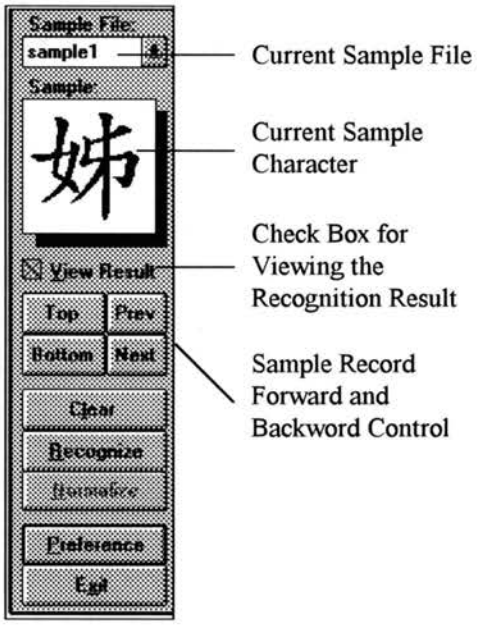


Figure B.12: Command bar and its components.

the buttons are self-explanatory and are the same as the menu options. Figure B.12 gives a brief labeling of those that are not mentioned in the menu options.

Status Bar

There are two status bars under RECOGNIZER. They both present information which echoes user commands or input. This information includes messages from the

main window, x-y coordinates of the pointing device in the writing zone, stroke count of the current writing, number of characters collected, number of reference characters and processing mode. Figure B.13 shows the components of the two status bars.

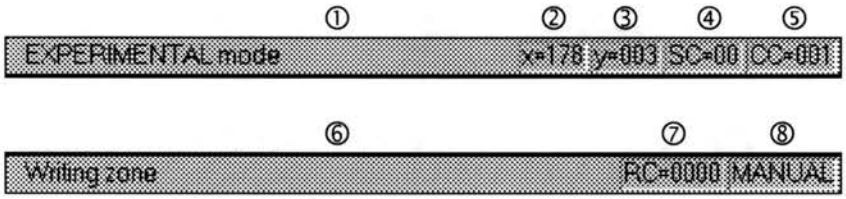


Figure B.13: Status bars and their components. ①: mode specifier (REAL or EXPERIMENTAL); ②: x coordinate of the pointing device; ③: y coordinate of the pointing device; ④: stroke count of the current writing; ⑤: characters collected; ⑥: message pane; ⑦: reference character count; ⑧: processing mode (MANUAL or AUTOMATIC).

Writing Window

This is a child window inside the main window under RECOGNIZER. It is the only place where the user can write. When the writing window is active, a pen style cursor is displayed; otherwise, either an arrow cursor or a stop cursor is displayed.

Normalization Window

For characters processed, each input is normalized and displayed in order, up to 36 characters maximum. This window provides a brief idea about the standarization of each input.

Result Window

The result window presents the result of recognition to the user when the 'View Result' check box in the command bar is checked. This window displays the selected candidates in descending order based on the measure of similarity. If the system is in experimental mode, the recognition result will also be output to an external file for statistics reasons.

VITA

Song-Shen Yeh

Candidate for the Degree of

Doctor of Philosophy

Thesis: ON-LINE CHARACTER RECOGNITION OF HANDPRINTED CHINESE CHARACTERS USING FUZZY MEASURING AND STRUCTURAL ANALYSIS

Major Field: Computer Science

Biographical:

Personal Data: Born in Taiwan, Republic of China, December 11, 1961, the son of Hsiu-Pin Yeh and Fu-Mei Pen.

Education: Graduated from Cheng Kung High School, Taipei, Taiwan, Republic of China, in June 1980; received Bachelor of Science degree in Navigation from National Taiwan College of Marine Science and Technology in January, 1985; received Master of Science degree in Computing and Information Science from Oklahoma State University in July 1990; completed requirements for the Doctor of Philosophy degree at Oklahoma State University in May, 1995.

Professional Experience: Teaching/Research Assistant, Department of Computer Science, Oklahoma State University, August 1989 to December 1994. Navigator, Evergreen Marine Corporation, Taiwan, Republic of China, November 1985 to November 1986. Salesman, Evergreen Marine Corporation, Taiwan, Republic of China, January 1985 to November 1985.