



National Library
of Canada

Bibliothèque nationale
du Canada

Canadian Theses Service

Services des thèses canadiennes

Ottawa, Canada
K1A 0N4

CANADIAN THESES

THÈSES CANADIENNES

NOTICE

The quality of this microfiche is heavily dependent upon the quality of the original thesis submitted for microfilming. Every effort has been made to ensure the highest quality of reproduction possible.

If pages are missing, contact the university which granted the degree.

Some pages may have indistinct print especially if the original pages were typed with a poor typewriter ribbon or if the university sent us an inferior photocopy.

Previously copyrighted materials (journal articles, published tests, etc.) are not filmed.

Reproduction in full or in part of this film is governed by the Canadian Copyright Act, R.S.C. 1970, c. C-30.

AVIS

La qualité de cette microfiche dépend grandement de la qualité de la thèse soumise au microfilmage. Nous avons tout fait pour assurer une qualité supérieure de reproduction.

S'il manque des pages, veuillez communiquer avec l'université qui a conféré le grade.

La qualité d'impression de certaines pages peut laisser à désirer, surtout si les pages originales ont été dactylographiées à l'aide d'un ruban usé ou si l'université nous a fait parvenir une photocopie de qualité inférieure.

Les documents qui font déjà l'objet d'un droit d'auteur (articles de revue, examens publiés, etc.) ne sont pas microfilmés.

La reproduction, même partielle, de ce microfilm est soumise à la Loi canadienne sur le droit d'auteur, SRC 1970, c. C-30.

**THIS DISSERTATION
HAS BEEN MICROFILMED
EXACTLY AS RECEIVED**

**LA THÈSE A ÉTÉ
MICROFILMÉE TELLE QUE
NOUS L'AVONS REÇUE**

**Structural Classification and Relaxation Matching of
Totally Unconstrained Handwritten Numerals**

Louisa Lam

A Thesis

in

The Department

of

Computer Science

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

September, 1986

Permission has been granted to the National Library of Canada to microfilm this thesis and to lend or sell copies of the film.

The author (copyright owner) has reserved other publication rights, and neither the thesis nor extensive extracts from it may be printed or otherwise reproduced without his/her written permission.

L'autorisation a été accordée à la Bibliothèque nationale du Canada de microfilmer cette thèse et de prêter ou de vendre des exemplaires du film.

L'auteur (titulaire du droit d'auteur) se réserve les autres droits de publication; ni la thèse ni de longs extraits de celle-ci ne doivent être imprimés ou autrement reproduits sans son autorisation écrite.

ISBN 0-315-32237-3

ABSTRACT

Structural Classification and Relaxation Matching of Totally Unconstrained Handwritten Numerals

Louisa Lam

In this thesis, the design and implementation of a system to classify unconstrained handwritten numerals are described. The system comprises a feature extractor and two classification algorithms functioning in sequence.

In the feature extraction process, the skeleton of a character pattern is decomposed into geometrical primitives consisting of line segments and convex polygons. A set of features is then extracted from each primitive, and these features provide the criteria for classifying the pattern.

The recognition process contains a structural classifier to identify the easily recognizable samples, and a relaxation algorithm to classify the rest of the data. The structural classifier is fast and processes the majority of the samples, while the relaxation method is robust enough to correctly recognize most of the distorted patterns.

The system was trained and tested on real-life ZIP codes obtained from the U.S. Postal Services. For the training set of 4000 samples, a recognition rate of 97.38% was obtained, while the recognition rate for the 2000 samples in

the testing data was 96%.

ACKNOWLEDGEMENTS

I am grateful to my thesis supervisor Professor C.Y. Suen for suggesting this topic, and for his suggestions and encouragement in the course of this work.

I would like to thank the U.S. Postal Services for providing the raw data, Pervez Ahmad for sharing the data he prepared for his research, and Yat-Keung Chu for the use of his skeletonization package.

I would also like to thank the staff of the Computer Center of Concordia University for their cooperation and for their generous allowances in computing facilities.

This project was supported in part by the Natural Sciences and Engineering Research Council of Canada.

TABLE OF CONTENTS

TITLE PAGE.....	i
SIGNATURE PAGE.....	ii
ABSTRACT.....	iii
ACKNOWLEDGEMENTS.....	v
TABLE OF CONTENTS.....	vi
LIST OF FIGURES.....	ix
LIST OF TABLES.....	xi
CHAPTERS	
1. INTRODUCTION.....	1
1.1 Introduction.....	1
1.2 General Pattern Recognition Systems.....	2
1.3 The Proposed Recognition System.....	4
1.4 An Outline of this Thesis.....	6
2. PREPROCESSING.....	8
2.1 Introduction.....	8
2.2 Skeletonization.....	8
2.3 Coding of the Skeleton.....	10
2.4 Tracing of the Skeleton.....	14
3. DESCRIPTION AND EXTRACTION OF FEATURES.....	16
3.1 Introduction.....	16

3.2 Description of Primitives.....	18
3.3 Decomposition into Primitives.....	20
3.3.1 Polygonal Approximation of Curves.....	21
3.3.2 Normalization of Patterns.....	24
3.3.3 Determination of Primitives.....	26
3.3.4 Convexity of Polygons.....	28
3.3.5 Types of Polygons.....	31
3.4 Feature Extraction from Primitives.....	37
4. STRUCTURAL CLASSIFICATION.....	41
4.1 Introduction.....	41
4.2 Use of Structural Classification.....	42
4.2.1 Merging of Primitives.....	42
4.2.2 Features used in Structural Classification.....	46
4.3 Classification of Characters with One Primitive.....	48
4.4 Classification of Other Characters.....	50
4.5 Observations on Structural Classification.....	55
5. RELAXATION MATCHING.....	62
5.1 Introduction.....	62
5.2 Relaxation Process.....	64
5.2.1 Feature Set of Primitives.....	64
5.2.2 Initial Probabilities.....	65
5.2.3 Iteration.....	69
5.2.4 Distance Computation.....	75
5.2.5 Classification Criteria.....	76

5.3 Mask Generation and Training.....	80
5.4 Conclusion.....	82
6. EXPERIMENTAL RESULTS.....	84
6.1 Experimental Data.....	84
6.2 Results of Structural Classification.....	87
6.3 Results of Relaxation Matching.....	91
6.4 Overall Results.....	96
7. CONCLUSIONS AND FUTURE DIRECTIONS.....	107
APPENDIX.....	110
REFERENCES.....	111

LIST OF FIGURES

1.2.1 A General Pattern Recognition System.....	3
1.3.1 Block Diagram of the Proposed Recognition System.....	5
2.3.1 3x3 Window of Pixel $P_{i,j}$	11
2.3.2 Condition Codes.....	12
2.3.3 End Points of Traced Curves.....	15
3.2.1 Classification of Polygons.....	19
3.3.1 Graph of Thresholding Function in Polygonal Approximation.....	23
3.3.2 Polygonal Approximation of Characters.....	24
3.3.3 Discard of Short Line Segment.....	27
3.3.4 Decomposition into Convex Polygons.....	28
3.3.5 'Eddy-Type' Shapes.....	29
3.3.6 Angles of a Polygon.....	30
3.3.7 Types of Polygons.....	33
3.3.8 Decomposition of a Character '2'.....	35
3.3.9 Decomposition of a Character '8'.....	36
3.4.1 A Skeleton and its Primitives.....	40
4.2.1 Samples with Primitives to be merged.....	44
4.2.2 Result of Merging Primitives.....	45
4.3.1 Some Characters with one Primitive.....	48
4.4.1 Example of Decision Tree.....	51
4.4.2 Samples of '2' and '5' with two West Polygons.....	52
4.4.3 Samples of '2', '4', and '7' with same Primitives...	53
4.4.4 Samples of '2' and '4' with same Primitives.....	54
4.5.1 Samples of '2', '4', and '8' with at least three Primitives.....	58

4.5.2 Substitutions in Structural Classification.....	60
5.2.1 Spring Connections.....	70
5.2.2 Graph of $r_{ij}^s(k,n)$	71
5.2.3 Mask and Input Character.....	74
5.2.4 Substitutions in Relaxation Matching.....	78
5.2.5 Samples Rejected in Relaxation Matching.....	79
6.1.1 Typical Data Samples and Skeletons.....	85
6.1.2 Additional Data Samples and Skeletons.....	86

LIST OF TABLES

3.4.1 Features Extracted from Primitives.....	40
4.5.1 Confusion Matrix for Structural Classification.....	61
5.2.1 Probability $P_1^t(k)$	74
5.3.1 Number of Masks per Class.....	81
6.2.1 Confusion Matrix for Structural Classification of Training Data.....	89
6.2.2 Confusion Matrix for Structural Classification of Testing Data.....	90
6.3.1 Confusion Matrix for Relaxation Matching of Training Set 1.....	93
6.3.2 Confusion Matrix for Relaxation Matching of Training Set 2.....	94
6.3.3 Confusion Matrix for Relaxation Matching of Testing Data.....	95
6.4.1 Summary of Time Requirements.....	97
6.4.2 Frequency Distribution of Masks by Number of Primitives.....	98
6.4.3 Time Distribution of Relaxation Matching.....	99
6.4.4 Overall Confusion Matrix for Training Data.....	103
6.4.5 Overall Confusion Matrix for Testing Data.....	104
6.4.6 Overall Confusion Matrix for Testing Data (Alternate Rejection Criteria).....	105
6.4.7 Overall Confusion Matrix for Training Data (Alternate Rejection Criteria).....	106

CHAPTER 1

INTRODUCTION

1.1 Introduction

Machine recognition of handwritten characters has been intensively and extensively studied during recent years. It is an intrinsically interesting problem, and satisfactory solutions to this problem would be very useful in various applications where it is necessary to process large volumes of handwritten data. A variety of approaches have been proposed and tested in research work on this subject ([23], [38]). The approaches have included the use of heuristics ([13] for example), structural and syntactic classifiers ([1], [8], [26] and [40] for example), and statistical descriptors ([11], [13], [22], [39] and [42] among others).

In particular, automatic recognition of totally unconstrained handwritten characters proves to be an even more challenging problem due to the geometric diversity of characters produced by writers with different habits and styles. Given the almost continuous spectrum of handwritten characters produced in the absence of constraints, any classification of them into a limited number of discrete

classes would inevitably contain a margin of error. It is interesting to note that even human beings, who have been trained to a greater extent than any computer, make about 4 percent mistakes when reading in the absence of context [37].

In research work on machine recognition of unconstrained handwritten characters, the result obtained would naturally be highly dependent on the quality and variability of the data used. It would depend, for example, on the number of authors producing the data, the conditions under which the samples are written, the sizes of the training and testing sets, and even on the methodologies employed in the calculation and reporting of the result itself. All these are in addition to the more fundamental issue of the particular pattern recognition system being used. So it is not surprising that recent research on automatic recognition of unconstrained handwritten numerals (for example, [1], [5], [7], [36] and [35]) has resulted in diverse recognition rates ranging from 93% to to 99.5% on testing data.

1.2 General Pattern Recognition Systems

While many different approaches have been used by numerous researchers for the recognition of patterns, a general pattern recognition system can be considered, with vast oversimplification, to contain 3 basic components: a

preprocessor, a feature extractor, and a pattern classifier. A simple block diagram of such a system is given in Figure 1.2.1.

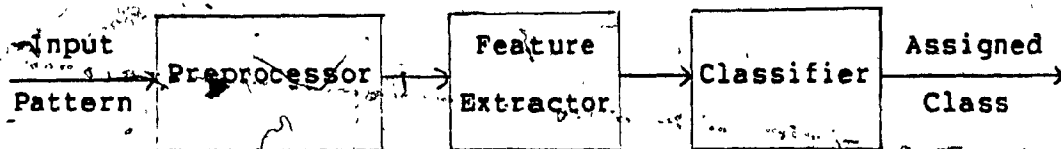


Figure 1.2.1 A General Pattern Recognition System

Under this general scheme, an input pattern, usually in the form of a matrix, is first passed through a preprocessor for smoothing and image enhancement. Significant and characterizing features are then extracted, and finally a classification is made of the pattern on the basis of the extracted features. Extraction of features is usually highly problem dependent, since what constitutes distinguishing features would depend strongly on the patterns to be classified. This process reduces an input pattern to its essential features, so it reduces the amount of memory required for storing the pattern, and at the same time it minimizes certain local distortions present in the pattern. Once the features have been extracted, a great variety of techniques can be used to classify the pattern according to its features.

1.3 The Proposed Recognition System

In this work, a character pattern is first decomposed into certain basic geometric subpatterns or primitives, after which the pattern is classified according to the features of these subpatterns and the connectivity between them. This is accomplished by the following procedure:

(1) Preprocessing. An input character pattern is smoothed and skeletonized. The skeleton is then traced, and the curves obtained are approximated by line segments.

(2) Feature extraction. The character is normalized in size and decomposed into its basis subpatterns or primitives which consist of convex polygons and line segments. Features are extracted from each primitive.

(3) Structural classification. Based on the configuration of primitives, certain patterns with simple structures are classified in this phase by decision trees according to their primitives and the features extracted from these primitives.

(4) Relaxation matching. The patterns not recognized by the structural classifier are matched against a set of templates using a relaxation process, and the distance between the pattern and each template is calculated. The

pattern is then classified according to the minimum distance criterion.

A block diagram of this recognition system is given in Figure 1.3.1.

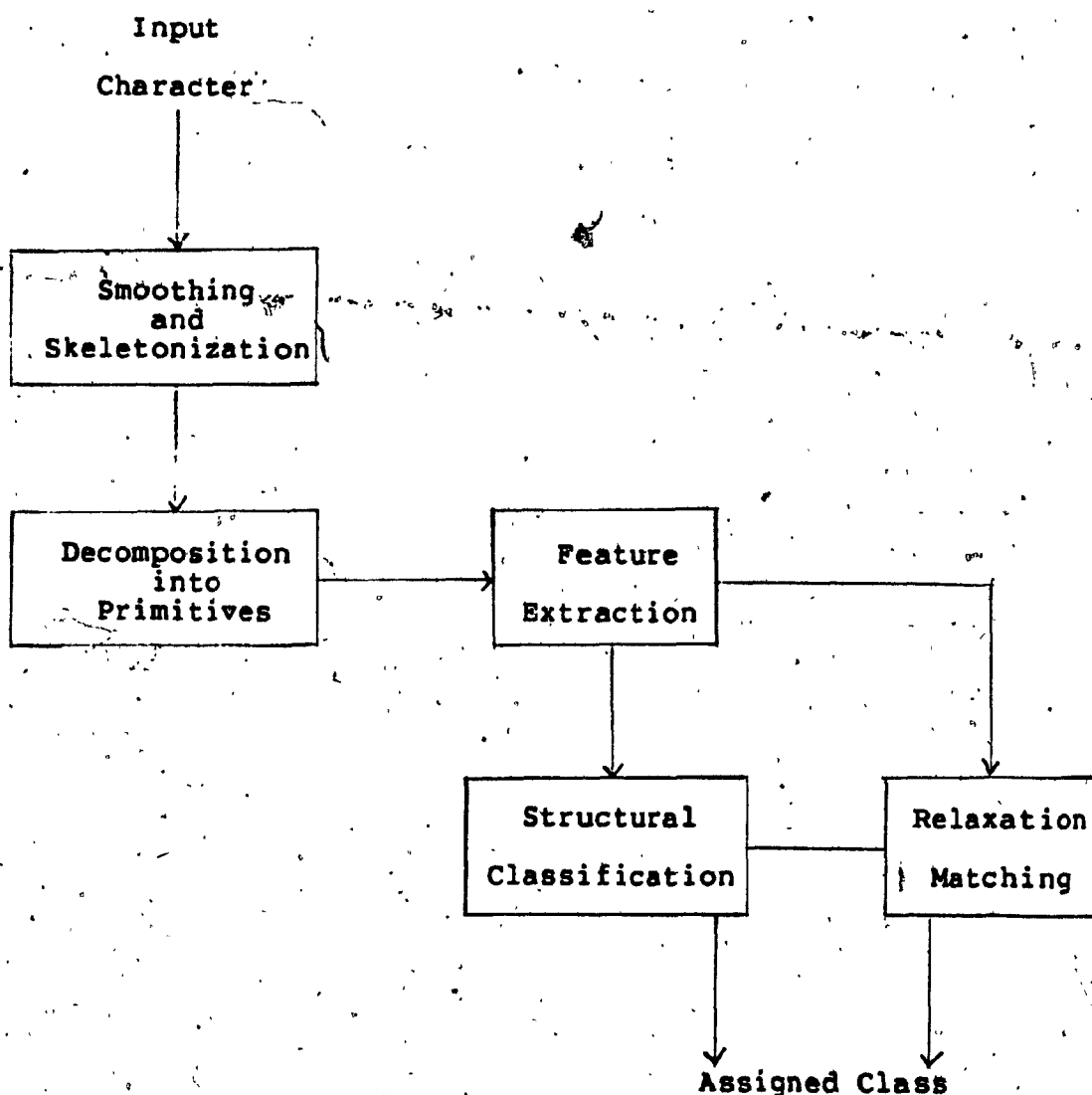


Figure 1.3.1 Block Diagram of the Proposed System

1.4 An Outline of this Thesis

In this work, a character pattern is decomposed into its basic primitives, and features are extracted from each primitive. The character is then classified according to the features extracted, either by the structural classifier or by relaxation matching against a set of stored masks. This recognition system is trained and tested on different sets of totally unconstrained handwritten numerals obtained from ZIP codes in the U.S.

In Chapter 2, the preprocessing of the input pattern is discussed. The input pattern is skeletonized and the skeleton is traced.

Chapter 3 contains a description of the primitives and of the process whereby these primitives are determined. The features to be extracted from the primitives are then described.

In Chapter 4, the first phase of the recognition process, that of structural classification, is described. The more easily recognizable samples can generally be classified in this phase by the use of decision trees, and only the more difficult patterns are processed through the heavy computational demands of relaxation matching.

In Chapter 5, the relaxation method is considered, and details are given of the method as it is applied in this

7
work.

Chapter 6 contains the results obtained by this recognition system as well as analyses of these results, and we conclude with some observations and possible directions for further work in Chapter 7.

CHAPTER 2

PREPROCESSING

2.1 Introduction

The input patterns used in this work have already been binarized. These binary patterns are skeletonized, and condition codes are assigned to the pixels of the resulting skeleton. The use of these condition codes helps to determine the starting and end points of each portion of the skeleton to be traced, and it also facilitates the tracing process itself. The skeleton is then traced in sections.


2.2 Skeletonization

In this work, the skeletonization package used is that of [2]. This package contains two main phases: thinning and adjusting. In the thinning process, the pattern is smoothed, the contour is traced, and the trace-points are stripped. This is performed repeatedly, with the tracing done in alternately clockwise and counterclockwise directions on the outside as well as the inside of the contours in order to obtain a balanced, thinned image.

Adjustments are then made of the skeleton when necessary in order to obtain a curve that is close to the medial line of the pattern.

While the benefits of using skeletonization in pattern recognition are not universally agreed upon, there are certain distinct advantages to using a thinned image rather than the contour of the pattern. By using skeletons, the outline of each character is traced once instead of twice (once for each of the inside and outside contours). So the number of curves, and hence the number of primitives obtained, would be reduced by about a factor of two. This results in a simpler structural classification scheme, and the advantage is even more significant in relaxation matching, where both memory and time requirements would be reduced by the use of skeletons. The memory required for storing the masks would be halved when skeletons are used, and in relaxation matching, where each primitive of each template is matched against each primitive of the input pattern, a similar reduction in computational time is obtained.

However, it must be noted that skeletonization by the use of software is a time-consuming process; the package used here requires an average of 0.35 seconds to process a character. Besides, any skeletonization package, however well designed, would create certain distortions and accentuate some local irregularities such as noise spurs.



Cross points are always a problem; lines frequently become discontinuous when they pass through these points. Sometimes, as a result of skeletonization, a sample of the character '8' would become a 'typical' sample of the character '9' when the inside opening of the bottom loop is very small in the original pattern, thus resulting in an understandable misclassification.

It would appear that some of the distortions caused by skeletonization may be diminished if the skeletonization process were designed to be problem dependent, and the type of primitives to be extracted are taken into consideration during the thinning of the patterns. This point of view has been considered and implemented [24] for the strokes of Chinese characters.

2.3 Coding of the Thinned Pattern

In order to decompose a thinned character into primitives while preserving the connectivity of the primitives, it is important to trace each section of the skeleton from an end point to a junction point, or vice versa. To obtain the intersection and end points for the tracing, and to facilitate the tracing process, each point in the skeleton is assigned a condition code according to the configuration of pixels in the 3x3 window centered at the given point. The coding scheme used here is an

adaptation of that in [24], and it has been used in [20], so it will only be briefly mentioned here.

A skeletonized character is represented by a binary matrix. Suppose the pixel $P_{i,j}$ has the 3x3 window shown in Figure 2.3.1.

$P_{i-1,j-1}$	$P_{i-1,j}$	$P_{i-1,j+1}$
$P_{i,j-1}$	$P_{i,j}$	$P_{i,j+1}$
$P_{i+1,j-1}$	$P_{i+1,j}$	$P_{i+1,j+1}$

Figure 2.3.1 3x3 Window of Pixel $P_{i,j}$

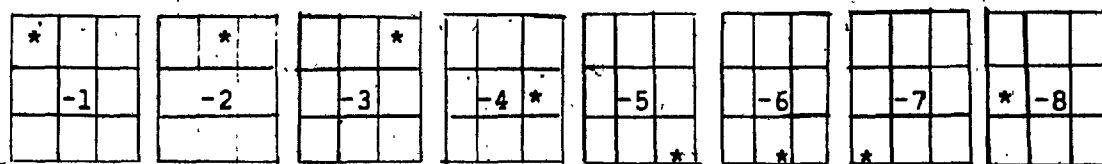
Let S be the sum of the 8 neighboring entries of $P_{i,j}$. The condition code $C_{i,j}$ of $P_{i,j}$ is assigned a value in the set $\{-9, -8, \dots, 0, 1, \dots, 9\}$ according to the following rules:

Rule 1: If $S \geq 3$, then $P_{i,j}$ is a junction point and $C_{i,j} = -9$.

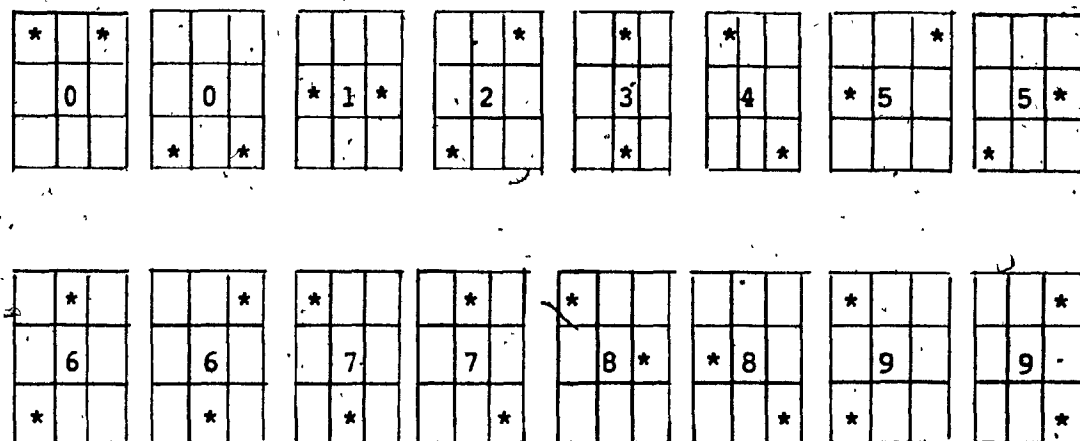
Rule 2: If $S=1$, then $P_{i,j}$ is an end point and $C_{i,j}$ is assigned a value in $\{-8, -7, \dots, -1\}$ as shown in

Figure 2.3.2(a).

Rule 3: If $S=2$, then $C_{1,j}$ is assigned a value in $\{0,1,\dots,9\}$ as shown in Figure 2.3.2(b).



(a) Negative Condition Codes from -1 to -8



(b) Positive Condition Codes from 0 to 9

Figure 2.3.2 Condition Codes

There is one significant difference, however, between [20] and this work in the way the condition codes are used. In the previous work, the codes in $\{0,5,6,7,8,9\}$ are considered to denote possible turning points, and these points are tested accordingly. In this work, only the negative condition codes have topological significance since they represent end points and junction points. The non-negative condition codes, however, are used to speed up the tracing process when considered in conjunction with the coordinates of the preceding pixel.

For example, if $C_{1,j} = 5$ and the predecessor of $P_{1,j}$ is $P_{p1,pj}$, then the next pixel to be traversed would be $P_{ni,nj}$ where

$$(ni,nj) = \begin{cases} (i-j+pj, 2*j-pj) & \text{if } i=p1, \\ (i, 2*j-pj) & \text{otherwise.} \end{cases}$$

Similar formulas are derived to determine the location of the next pixel for the other positive condition codes. In this way, a curve is traced until a pixel with a negative code is encountered, indicating that the end of a section of the skeleton is reached.

2.4 Tracing of the Skeleton

The tracing of each curve starts, in the order of preference, ~~from~~

- (1) an end point,
- (2) a junction point, and
- (3) the upper left hand corner for characters such as a closed '0' in which there are no end or junction points.

The tracing proceeds, whenever possible, in the order in which the characters are usually written, from the top of the character downwards. When a curve is traced from an end point of the skeleton to a junction point, the tracing will next resume from this junction and proceed outwards. This was designed to preserve the connectivity of the character.

Each pixel encountered in the tracing process is stored in an array while the pixel is 'switched off'. This ensures that pixels are traversed only once, and it also enables the procedure to determine when a skeleton has been completely traced in the case of characters containing disjoint strokes.

The non-negative condition codes serve to indicate the location of the next pixel, while encountering a negative condition code indicates that the tracing is complete for a section of the skeleton. If tracing ends at a junction

point, then the locations of all the adjoining junction points are averaged to give the end point of this section of the skeleton. This end point would also be the starting point of all other branches originating from this junction, so it is stored accordingly. An example of this is given in Figure 2.3.3, where the branch points to be averaged are denoted by '-', and the end points of curves are denoted by letters of the alphabet.

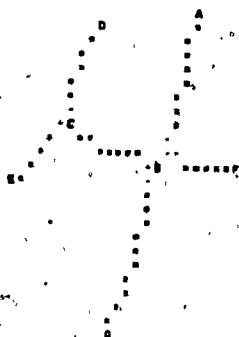


Figure 2.3.3 End Points of Traced Curves

After a curve has been traced, it is approximated by line segments. The line segments are then grouped into primitives.

CHAPTER 3

DESCRIPTION AND EXTRACTION OF FEATURES

3.1 Introduction

In the feature extraction phase of pattern recognition, the features to be extracted are naturally dependent on the recognition algorithm to be used. In this work, the ultimate aim is to study the applicability of the relaxation method to the recognition of handwritten numeric characters, so the primitives and features are designed in accordance with this objective.

The relaxation method has been applied to the recognition of handwritten Chinese characters [43]. Since Chinese characters are composed mostly of linear strokes, it is natural to use line segments as their primitives. So in [43], character patterns are approximated by line segments, and features are extracted from these line segments for relaxation matching.

However, the number of approximating line segments per character obtained by the above-mentioned process is relatively large, due partly to the complexity of Chinese

characters and partly to the fact that the contour of the character is used rather than the skeleton. In this work, the skeletons of the characters are used. In addition, it was felt that, in the case of handwritten numeric characters, curves occur with equal or greater frequency than linear strokes, so the use of line segments alone as primitives might not have been the best approach for the character patterns to be classified here. While curves can always be approximated by line segments, the number of line segments used would be highly dependent on the shape of the particular curve. The use of line segments as primitives may therefore result in curves of the same basic type being represented by different numbers of primitives, thus adding another variable to the matching process.

As a result, it was decided to use convex polygons and line segments as primitives, with the polygons approximating curves. Apart from adhering more to the nature of the characters, the use of polygons also resulted in curves of the same type being represented by a more consistent number of primitives independently of their curvatures. Some of the practical advantages of using polygons are reduced storage requirements and more rapid classification. The majority of primitives are composed of several line segments, so the number of primitives per character is significantly reduced. (On the average, each primitive contains 3 line segments). Consequently, less memory is

required for storing the masks, and relaxation matching is performed on a smaller feature set, hence it can operate more quickly. In addition, this choice of primitives has made possible the use of a fast, structural classifier to process the more easily recognizable characters.

The curves traced are thus approximated by convex polygons and line segments. Since the character patterns have very different sizes ranging from about 7x12 to 53x53, they are normalized in size for matching. This is easily accomplished since normalization is necessary only for the vertices of the polygons and the end points of the line segments. Features are then extracted from each primitive.

3.2 Description of the Primitives

As stated above, the primitives are convex polygons and line segments. It should be noted, however, that the convex polygons may be 'open', or that the polygonal convex regions may be unbounded. (A polygonal region is unbounded if and only if it contains a ray). The polygons are classified as east, north, west, and south as shown in Figure 3.2.1.

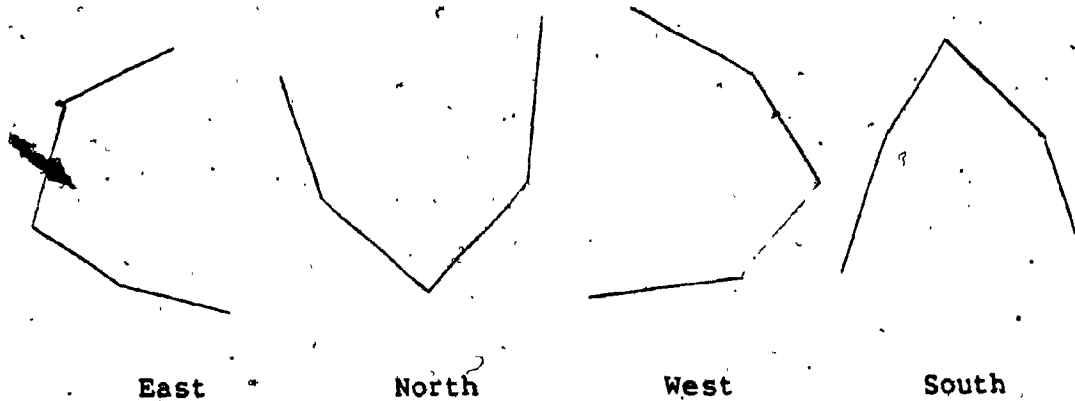


Figure 3.2.1 Classification of Polygons

The polygons shown have four sides each, but in this work no distinction is made between polygons having different numbers of sides as long as they have the same orientation, even though a polygon must have at least two sides for obvious reasons.

Due to the continuous rotation that polygons can undergo, the border lines between east and north polygons, east and south polygons etc. are necessarily fuzzy. In the structural classification phase, a polygon is considered to be north or south if the starting and end points of the polygon are at most three rows apart, and they are considered to be east or west otherwise. The direction of the opening then determines the rest. In the relaxation matching of the primitives, polygons are considered to be either east or west only, since at that stage the features extracted from the primitives are used in the calculations,

and these features would also act as distinguishing characteristics.

In both phases of classification, closed polygons or loops are considered to be a different type of primitive. So three types of primitives are used:

- (1) loops,
- (2) convex polygons with unbounded regions; these can be east, west, north, or south, and
- (3) line segments.

3.3 Decomposition into Primitives

When the skeleton of a character has been traced, the curves obtained are decomposed into a set of primitives of the types described above. This decomposition of a pattern is achieved in several steps. The traced curves are first approximated by sequences of line segments, after which the end points of the line segments are normalized so that all the resulting patterns would have the same size for relaxation matching. Some sequences of approximating line segments would already form primitives of the required types, while others may have to be regrouped into appropriate primitives.

3.3.1 Polygonal Approximation of Curves

In the approximation of a curve by a polygon, various approaches have been proposed (for example, [16], [25], and [31]). In this work, the curves to be approximated are relatively short, since tracing of a curve must terminate when an end point or a junction point of the skeleton is reached.

The polygonal approximation algorithm used here is adapted from that in [29]. Using this method, the beginning and end points of a curve are first determined. This is obvious if the curve is not closed; in the case of a closed curve, two points that are extreme in opposite directions can be used. A line segment is then drawn to connect these two points, and the point on the curve that is farthest away from this line is added as a vertex of the polygon, provided this maximum distance exceeds a certain threshold value. This process is performed iteratively, using a stack, until no more vertices can be added.

Due to the diverse sizes of the characters being considered, it was found that no constant threshold value would yield satisfactory approximations in all cases. Consequently a piecewise linear thresholding function was decided upon for flexibility and ease of computation.

For any character, the threshold value T used is defined by

$$T = \begin{cases} 1.2 & \text{if } X \leq 30, \\ X*0.04 & \text{if } 30 < X < 50, \\ 2 & \text{otherwise,} \end{cases}$$

where $X = m + n$ if the character has dimensions $m \times n$.

The upper limit for T was established in order that small subpatterns in a large character are not overly simplified. At the same time, a lower limit was set to avoid approximating a small character by an excessive number of line segments. The thresholding function is graphed in Figure 3.3.1.

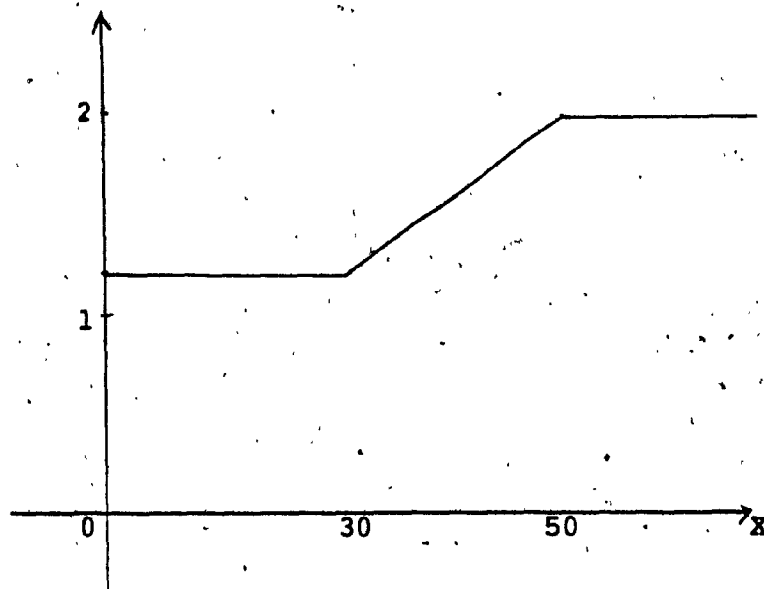


Figure 3.3.1 Graph of Thresholding Function
in Polygonal Approximation

Figure 3.3.2 illustrates the polygonal approximation of some skeletons. In this figure, letters of the alphabet in the traced skeleton denote the end points of approximating line segments.

When the end points of all approximating line segments have been determined, these points are transformed so that all resulting patterns would have the same dimensions.

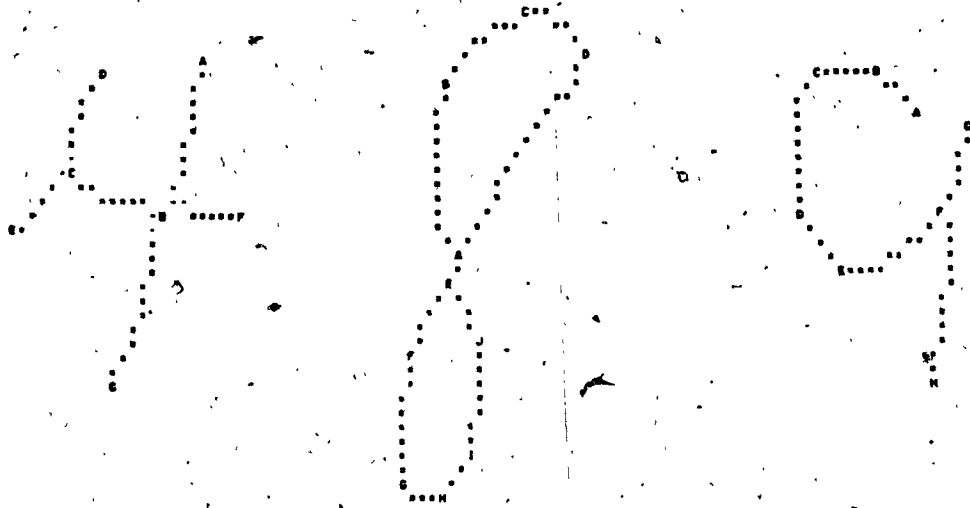


Figure 3.3.2 Polygonal Approximation of Characters

3.3.2 Normalization of Patterns

In order to accurately match character patterns against templates using features (to be described in Section 3.4) which include the coordinates of the starting and end points of primitives as well as the distance between these points, it is important that all patterns should have the same size. So size normalization is required.

For the rest of the feature extraction process, only the end points of the approximating line segments are significant. Consequently, normalization should be performed when these points have been determined, and only these points would have to be transformed. This requires a

smaller number of computations, and it also circumvents the problem of normalizing the skeleton itself, where breaks may be created. On the average, a character contains 55 pixels in its skeleton, and normalization is required for only 7 points.

Each character is normalized so that the resulting character has dimensions 25x20, the average dimensions of the data set. This is achieved by using the affine transformation

$$F(x,y) = (19*(x-1)/(n-1)+1, 24*(y-1)/(m-1)+1)$$

when the matrix has dimensions $m \times n$. It is well known that the image of a convex set (in euclidean n -space) under an affine transformation is also convex [12], so the types of the primitives are invariant under this transformation.

After normalization, characters have dimensions 25x20. It is obvious that the normalization process does change the appearance of some characters, especially those whose original dimensions are far from the ratio 5:4. This distortion is problematic in some samples of the character '1', where the presence of a weak cavity may result in an approximation that is closer to a '7'. This problem has been eliminated here by the use of a repeated approximation for certain characters, and this will be discussed in Section 4.3. In other characters, the distortion introduced

does not pose serious problems, and normalization is necessary for the application of relaxation matching.

3.3.3 Determination of Primitives

When each curve of a character skeleton has been approximated by a sequence of line segments, the line segments are rearranged into primitives when necessary. This rearrangement may be required for the following reasons:

(1) A very short line segment that is connected at only one end is assumed to be caused by noise and discarded, thus necessitating the merging of the line segment sequences to which this segment was connected. This is illustrated in Figure 3.3.3, where the deletion of the short segment results in a merging of its neighboring line segments shown in (b). A line segment is discarded this way when it contains at most k pixels, where

$$k = \min\{0.08 \cdot (m+n), 4\}$$

for a character with dimensions $m \times n$.



(a) Original Approximation

(b) New Approximation

Figure 3.3.3 Discard of Short Line Segment

(2) An approximating polygon is not convex. In this case it would be decomposed into two or more convex polygons or line segments.

It should be noted at the outset that this second consideration is necessary only for open polygons. Closed polygons occur from the approximation of loops, and loops are identified immediately from the fact that the starting and end points of the traced curves coincide.

The decomposition of open polygons into convex polygons is described in the next subsection.

3.3.4 Convexity of Polygons

To determine whether a polygon is convex, it is sufficient in the majority of cases to consider the direction of rotation as the sides of the polygon are traversed in order. If the rotations measured by the interior angles of the polygon do not have the same (clockwise or counterclockwise) direction, then the polygon is not convex. For example, the 6-sided polygon in Figure 3.3.4 has 5 interior angles $\theta_1 - \theta_5$. Since θ_1 and θ_2 represent clockwise rotation while θ_3 is counterclockwise, the polygon should be segmented at the vertex of θ_3 .

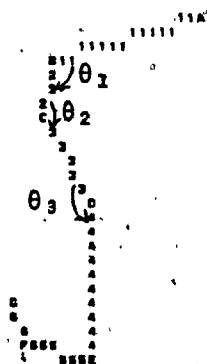


Figure 3.3.4 Decomposition into Convex Polygons

However, the above condition, while sufficient, is not necessary. This is illustrated by the polygons in Figure 3.3.5, where the direction of rotation remains the same in each case even though the polygon is not convex.

These are referred to as 'eddy-type' shapes in [22].



(a) Clockwise Rotation

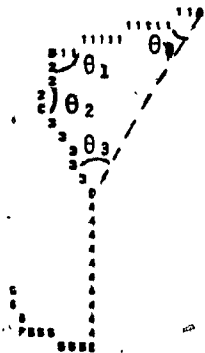
(b) Counterclockwise Rotation

Figure 3.3.5 'Eddy-type' Shapes

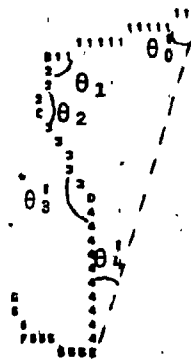
As a result, the sum of the interior angles of a polygon is used here to determine its convexity, while the direction of rotation is used to determine the direction of its opening, that is, the type of the polygon.

It is well known that the sum of the interior angles of an n -sided convex polygon is $(n-2)\pi$ radians. (These polygons are closed). To determine if an n -sided open polygon is convex, it can be assumed that $n \geq 2$, since an open polygon having only 2 sides must be convex. By joining the starting and end points of the polygon, a closed polygon having $(n+1)$ sides is obtained. In the polygon shown in Figure 3.3.6(a), for example, the polygon having vertices A,

B, C, and D is convex since $\theta_0 + \theta_1 + \theta_2 + \theta_3 = 2\pi$ radians. This polygon, however, cannot be extended to include vertex E since a calculation of the newly formed angles θ_0' , θ_3' , and θ_4' shown in Figure 3.3.6(b) results in $\theta_0' + \theta_1 + \theta_2 + \theta_3' + \theta_4' \neq 3\pi$ radians. In fact, the sum of these angles equals 3π if and only if $\theta_3' = \pi$, which would be a contradiction since it implies that the vertices C, D, and E are collinear. So the first convex polygon cannot contain vertex E, and point D must be the junction of the two convex polygons.



(a)



(b)

Figure 3.3.6 Angles of a Polygon

When the approximating line segments have been grouped into primitives, the type of each polygon is determined by the procedure described in the next subsection.

3.3.5 Types of Polygons

For a convex polygon, its type is determined by the direction of rotation of its interior angles when the sides of the polygon are traversed in order. Since the polygon is convex, the direction of rotation must remain the same for all interior angles as previously stated, so it is necessary to determine the direction of rotation for only one such angle.

The direction of rotation of an angle is determined here by the vector product of the vectors that form the arms of the angle. Since the vectors are on the xy-plane, they can also be considered as vectors in 3-dimensional space, and the vector product of two such vectors is easily calculated. It is an elementary result that the vector product of two vectors is a vector in the direction of the positive z-axis when the rotation from the first vector to the second is counterclockwise, and it is a vector in the direction of the negative z-axis when the rotation is clockwise.

At this stage of the feature extraction process, the coordinates of the vertices of the polygon are known, so the vectors and their vector products are easily calculated, thus the direction of rotation is determined.

As previously stated, a polygon is classified as north or south when its starting and end points are located at most three rows apart, and it is classified as east or west

otherwise.

A west polygon is obtained if

(i) the rotation is counterclockwise when the polygon is traced downwards (that is, the end point of the polygon is below the starting point), or

(ii) the rotation is clockwise when the polygon is traced upwards.

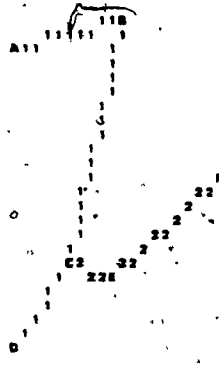
This is illustrated in Figure 3.3.7(a), where the two west polygons shown are obtained from the above conditions.

A south polygon is obtained if the starting and end points are located at most three rows apart, and

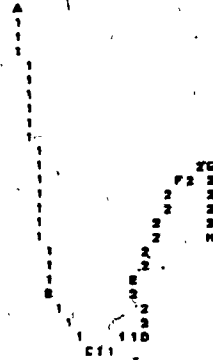
(i) the rotation is counterclockwise when the polygon is traced from left to right, or

(ii) the rotation is clockwise when the the polygon is traced from right to left.

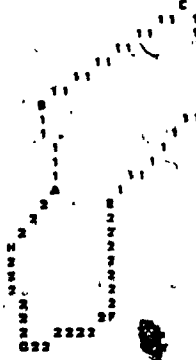
East and north polygons are defined analogously. Figure 3.3.7(b) shows two east polygons obtained from clockwise and counterclockwise rotations respectively, and Figure 3.3.7(c) shows a south and a north polygon that together constitute a sample of the character '8'.



(a) Two West Polygons



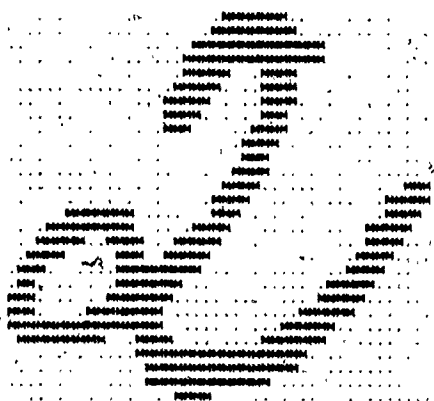
(b) Two East Polygons



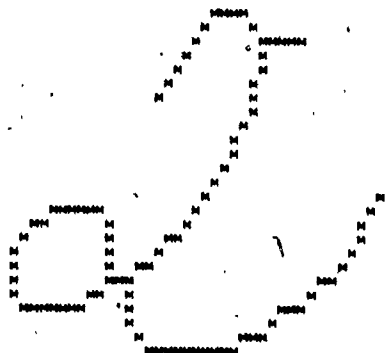
(c) North and South Polygons

Figure 3.3.7 Types of Polygons

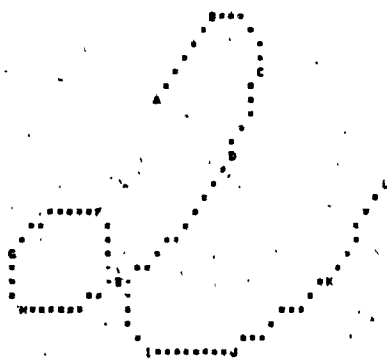
After the type of each primitive has been determined, the primitives are numbered. Figures 3.3.8 and 3.3.9 illustrate the process through which a binary character is decomposed into primitives. The binary character is shown in (a) and its skeleton is shown in (b); the polygonal approximation is shown in (c), and the resulting primitives are shown in (d). In this and subsequent figures, letters of the alphabet in the approximations denote the end points of line segments, while numerals represent the numbers of the primitives.



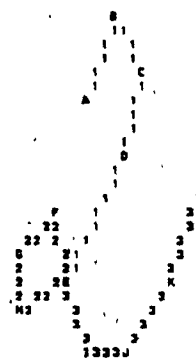
(a) Binary Character



(b) Skeleton

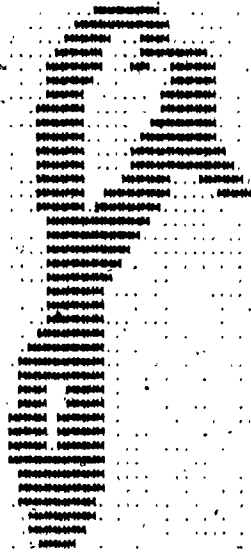


(c) Polygonal Approximation



(d) Primitives

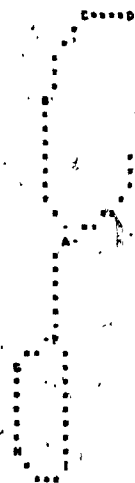
Figure 3.3.8 Decomposition of a Character '2'



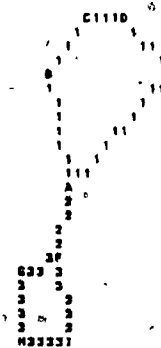
(a) Binary Character



(b) Skeleton



(c) Polygonal Approximation



(d) Primitives

Figure 3.3.9 Decomposition of a Character '8'

3.4 Features extracted from Primitives

When all the primitives of a character have been determined, the following features are extracted from each primitive:

- (1) The coordinates of its starting point.
- (2) The coordinates of its end point.
- (3) The coordinates of its center. This is obtained by averaging the maximum and minimum row and column numbers of the vertices in the primitive.
- (4) The length. This is the distance between the starting and end points. If the primitive is a line segment, this feature is its length; if the primitive is a polygon, this represents the width of its opening.
- (5) The type of the feature; this is represented by a number as follows:

Type of Feature	Type Number	
Line segment	-2	
Loop	0	
'Near' loop	1,2,3,4	for west, south, east, and north respectively.
Polygon	5,6,7,8	

A polygon is assumed to be a loop if its length is at most 4.5 units; otherwise it is designated a 'near' loop if its length is at most 8. The criterion for loops was designed to deal with the frequent occurrence of broken loops in the data set. 'Near' loops are used only in structural classification; in this phase, not all the extracted features are used, and so a finer partition among the primitives is useful.

(6) The direction; this is the angle from the horizontal axis to the line joining the starting and end points. If the primitive is a line segment, this obviously measures its slope; if the primitive is a polygon, this indicates the angle of its opening, which is useful in some cases.

(7) The starting point primitives. These are the primitives that end at the starting point of the primitive being considered.

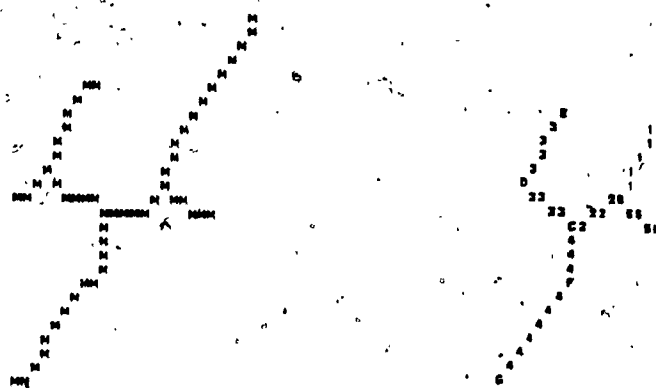
(8) The end point primitives. These are the primitives that start at the end point of the primitive being considered.

The features (7) and (8) are significant since they define the connectivity of the primitives, and this provides the contextual information used in relaxation matching. This information is not used in structural classification.

In order to reduce the structure of a character to its essential components, primitives which are very short line segments (with length less than 1.5 units) are ignored. Since this would create gaps between some primitives, two primitives are considered to be adjoining if the starting point of one is located within 1.5 units of the end point of the other. So the starting and end point primitives are determined accordingly.

The feature extraction process, from reading and tracing of the skeleton to obtaining all the features, requires on the average 0.08 seconds per character.

Figure 3.4.1 shows a character and its approximating primitives, and Table 3.4.1 lists the features extracted from these primitives.



(a) Skeleton

(b) Primitives

Figure 3.4.1 A Skeleton and its Primitives

Primitive	Starting Point	End Point	Center	Length	Type	Dir	SPP	EPP
1	A(1,20)	B(12,13)	6,16	13.04	-2	58		2,5
2	B(12,13)	C(14, 8)	13,10	5.39	-2	22	1	3,4
3	C(14, 8)	E(6, 7)	10, 5	8.06	7	97	2	
4	C(14, 8)	G(25, 1)	19, 4	13.04	5	58	2	
5	B(12,13)	H(14,17)	13,15	4.47	-2	153	1	

Dir = Direction

SPP = Starting Point Primitives

EPP = End Point Primitives

Table 3.4.1 Features extracted from Primitives

CHAPTER 4

STRUCTURAL CLASSIFICATION

4.1 Introduction

When all features have been extracted from each primitive of a character, the classification process begins. While the aim was to classify handwritten numerals by the use of relaxation matching, and the features were extracted accordingly, it was felt at the same time that using relaxation matching alone would be very time-consuming. This algorithm requires the matching of every primitive of an input pattern against every primitive of a selected subset of masks, and heavy computations are involved. Even though the number of necessary comparisons has been significantly reduced through the added use of polygons (instead of line segments alone) as primitives, it would still be expedient to have an alternative classification scheme for the easily recognizable characters.

Consequently, a structural classification scheme was designed. This process was made possible by the choice of primitives, and these primitives together with some of their

features are the parameters used in the classification.

It is recognized at the same time that a structural classification scheme that can reliably identify all unconstrained handwritten numerals would have to be considerably more complicated than what is presented here. It may have to use more types of primitives [22] and/or more extracted features. The classification process may have to be more sophisticated ([1], [34], and [35] for example). Such was not the intention here.

Instead, the intention was to design a structural classifier that uses the features already extracted to classify the easily recognizable characters rapidly. Only the characters that cannot be classified in this phase are passed on to the heavy computational demands of relaxation matching. In the results given in Chapter 6, it can be seen that the use of this structural classifier does expedite the classification process significantly.

4.2 Use of Structural Classification

Generally, only characters having at most two primitives are processed through the structural classifier. Exceptions are made for certain samples of '4' and '8' whose distinctive structures make them easily recognizable. These are samples of '4' that are composed exclusively of at least five line segment primitives, and samples of '8' that

contain two loops joined by a line segment. In these samples of '8', the line segment between the loops is often the result of skeletonization.

Apart from these samples of '4' and '8', characters having at least three primitives are not considered in this phase. Characters having two primitives are considered if the primitives are aligned vertically so that the center of one primitive is at least 5 rows above the center of the other. All other characters are passed to relaxation matching, and the same is true of characters that the structural classifier cannot identify uniquely. About 80% of the samples are classified in this phase, and the other 20% are passed to relaxation matching.

4.2.1 Merging of Primitives

In certain input characters, a merging of the primitives is performed before structural classification. That this is desirable is illustrated by the samples in Figure 4.2.1



Figure 4.2.1 Samples with Primitives to be merged

Since tracing of the skeleton and determination of primitives always stop at end or junction points of curves, each of the samples in Figure 4.2.1 contains 3 primitives as shown. This means that the samples cannot be classified structurally (without increasing the complexity of the structural classifier) even though the samples are easily recognizable.

In view of the above, a criterion for merging primitives was established: two primitives are merged to form a new primitive if the merging results in a primitive, i.e., the result is a convex polygon or a line segment. Determination of this condition is by use of the same procedure described in Subsection 3.3.3.

As a result of merging, each of the samples in Figure 4.2.1 has 2 primitives as shown in Figure 4.2.2.



Figure 4.2.2 Result of merging Primitives

It should be noted that merging primitives destroys the connectivity information: in each sample of Figure 4.2.2, the two primitives are no longer adjoining because the end point of the first primitive has been changed in merging. Since connections among the primitives are important parameters for relaxation matching, the merged primitives are used only for structural classification. In relaxation matching, the features used are those already described in Section 3.4.

In this way, each input character is represented by two slightly different sets of features. The second set of

features is easily obtained from the first. This also does not increase the memory requirements to any extent, since each character contains on the average only 2.3 primitives before merging and 2 primitives after merging. For the masks, of course, only one set of features would be required since they are used only for relaxation matching.

4.2.2 Features used in Structural Classification

In the structural classification of a character, the following features of each primitive are considered:

- (1) The type.
- (2) The center.
- (3) The length.
- (4) The direction.

The above are the parameters most often used. In addition, the following features are sometimes useful, and they are available from the feature extraction process.

(5) The number of sides in a polygon; this measures how 'curved' a polygon is.

(6) The direction of rotation: clockwise or counterclockwise.

(7) The row and column displacements of the primitive:

these are the differences between the maximum and minimum row and column numbers of the vertices in the primitive. These values are obtained in the process of determining the center of the primitive, and they indicate the size of the primitive.

(8) The direction of the first line segment in the first primitive.

Use of these features in the classification process will be illustrated in Section 4.4.

Characters having at most two primitives are processed through the structural classifier using decision trees. The rules used in the decisions are derived from considerations of the characters not only as they are expected to be written, but also as they are actually written. In the process of defining this classification procedure, an initial set of rules were established through theoretical considerations of the structures of these characters. These rules were then gradually refined in the training process by their application to about 3000 characters in the training set.

4.3 Classification of Characters with One Primitive

The only numeric characters which can consist of only one primitive are some samples of the characters '0', '1', '6', and '7'. Naturally, these characters can also be written in a variety of ways consisting of more primitives. In fact, when the character '6' is more or less correctly written, it should contain at least two primitives. Nevertheless, given a character having only one primitive, the task is to determine the class (among the four noted above) to which this character can most reasonably be assigned. Some samples of such characters are shown in Figure 4.3.1.

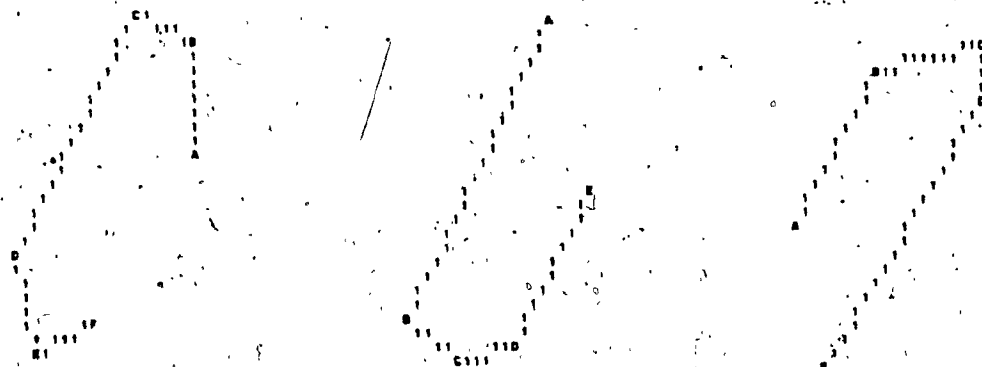


Figure 4.3.1 Some Characters with one Primitive

These characters are classified according to the following rules:

(1) If the primitive is a loop or a polygon that is close enough to a loop, i.e., the opening is small, then the character is classified as '0'.

(2) If the primitive is a north or south polygon, the character is rejected since any classification of such would involve a high degree of uncertainty. These rejected characters are not passed on to relaxation matching.

(3) If the primitive is a west polygon, the character is classified as '7'; otherwise it is a '6'.

(4) If the primitive is a single line segment, the character is '1'.

In the application of Rule (1), the frequent occurrence of handwritten zeros with broken loops has to be taken into consideration. So it is necessary to determine when a polygon can be reasonably considered to be a loop. An empirically established condition is that the length of its opening should be at most 12.5. Of course this condition was used only for characters with one primitive.

In the application of Rule (3), it was found that samples of '1' that are written with weak cavities are sometimes approximated by polygons as a result of normalization. In order to avoid misclassifying these

characters, characters having only one primitive that is a polygon with at most three sides are approximated again using a larger threshold value in the polygonal approximation. This results in a coarser approximation which preserves only the broad outline of the character, and this new approximation is used in the classification. This procedure was very effective in distinguishing between samples of '1' and '7', and it does not affect the classification of other characters since these other characters would not have such simple structures.

Structural classification of characters having two primitives is considered in the next section.

4.4 Classification of Other Characters

For characters having two primitives, their structural classification is considerably more complicated, and the features of the primitives often have to be used as parameters in the decision-making process. Decision trees are used, with the top primitive as root and the assigned classes as leaves. It is well known that the geometric diversity of shapes for handwritten numerals makes linear separation difficult, so there is more than one leaf for each class.

As an example, the decision tree in Figure 4.4.1 might be considered.

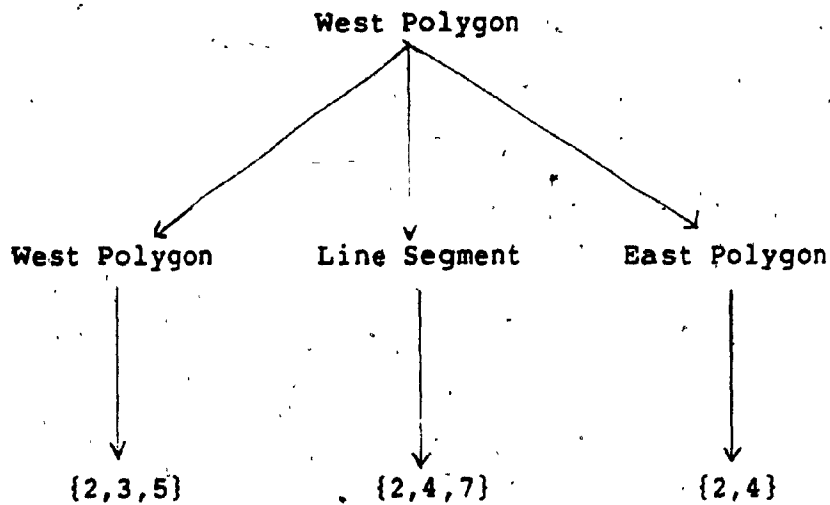


Figure 4.4.1 Example of Decision Tree

The majority of the samples of the character '3' are classified according to the left branch of the tree, in Figure 4.4.1. However, some samples of the characters '2' and '5' have the same two primitives; examples of these are shown in Figure 4.4.2.

In this and subsequent figures and discussion in this chapter, the primitives considered are those used in structural classification, so they are the result of merging primitives whenever this process is appropriate, and the numeric characters in the approximations represent the numbers of these primitives.

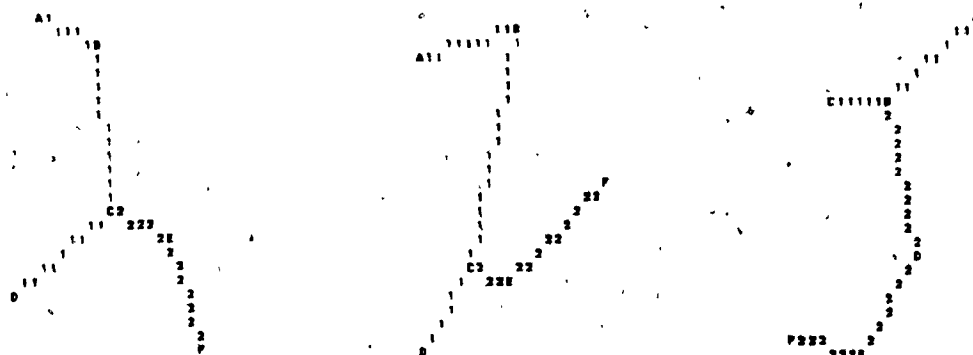


Figure 4.4.2 Samples of '2' and '5' with two West Polygons

As a result, features of the second primitive (its row displacement and direction of rotation) would have to be used to distinguish between samples of '2' and '3'. When the row displacement of the second primitive is small, or when its direction of rotation is clockwise, then the sample is classified as '2'; otherwise it is a '3'.

The sample of '5' shown occurs due to the fact that the top horizontal stroke is sometimes lost in the preparation of the data. To distinguish between such samples of '5' and samples of '3', the number of line segments in the first primitive and the direction of the first line segment was considered. If the first primitive has only two line segments of which the first is between 60 and 90 degrees from the horizontal, then the sample is classified as '5';

otherwise it is a '3'.

The center branch of the tree in Figure 4.4.1 is a result of samples of '2', '4', and '7' shown in Figure 4.4.3.

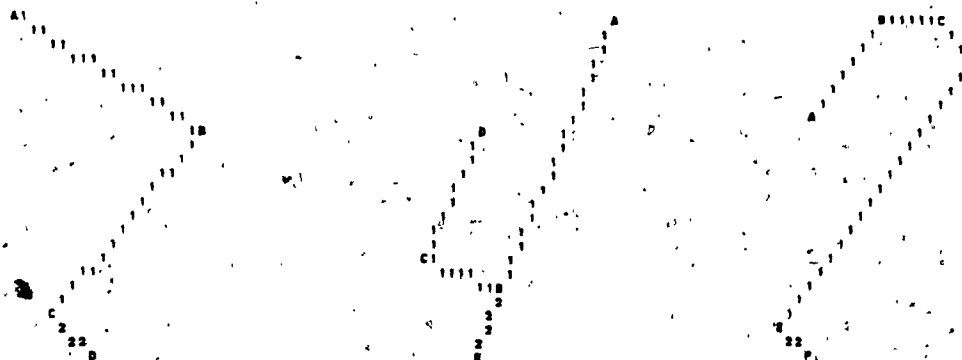


Figure 4.4.3 Samples of '2', '4', and '7'
with same Primitives

The above sample of '4' is easily identified by considering the direction of the first primitive, since this angle is usually small. Distinguishing between the '2' and '7' is much more difficult, and the criteria used here (the direction and length of the second primitive) is not entirely successful.

The right branch of the tree in Figure 4.4.1 is a result of samples illustrated in Figure 4.4.4, and the feature used

to distinguish between them is the direction of the second primitive.



Figure 4.4.4 Samples of '2' and '4' with same Primitives

It must be admitted that in this structural classification, the distinguishing features selected are not entirely foolproof, and they do fail in some cases. However, this method does provide an efficient means for classifying 80% of the characters, and the recognition rate obtained is 98.08%.

4.5 Observations on Structural Classification

The structural classifier presented here is not a complete classification scheme for all handwritten numerals, and its efficacy depends on the particular character class. In this section, the effectiveness of this method is considered.

It is worth noting that there is very little difference between the results obtained from the training and testing sets when this method is used. There are essentially two reasons for this. First of all, this method deals with the general structures of the characters and their possible variations rather than with the particular samples themselves; no effort was made to modify the rules in order to accomodate highly unusual samples since that would inevitably lead to misclassifications elsewhere. Secondly, when the classification rules were refined during the training process, a large enough set of data (about 3000 samples) was classified so that the resulting rules can probably be applied with generality and consistency.

The applicability of this method to the classification of these characters is considered below. The results given here are the combined results from the training and testing data, so there are altogether 6000 characters, with 600 characters per class. As previously stated, not all the characters are classified in this phase; out of these 6000

characters, 4783 were classified, and a confusion matrix is given at the end of this section. Separate results for the training and testing sets are given in Chapter 6.

As might be expected, this classification scheme is most effective for the character classes '0', '1', and '7', which have the simplest structures. Samples of the character '9', however, usually contain only two primitives, and most of them are also classified in this phase.

For the character '1', no problems were encountered after the introduction of a coarser polygonal approximation for samples having only one primitive with at most three line segments. All samples of this character were classified correctly in this phase.

Almost all samples of the character '0' were classified structurally, and substitutions are due to extensions of the samples above the loop, causing them to be misclassified as '6' or '8'. (At the same time, some samples of '6' and '8' are misclassified as '0' when the top primitives of the samples are very small.)

Misclassifications of '7' are usually due to similarities between these samples and some samples of '4' and '9'. Substitutions of samples of '9' are due mainly to the fact that samples of this character can be continuously transformed to samples of '5', and vice versa. Some of the samples of '9' are visually ambiguous, and it is difficult

to distinguish them from certain samples of '5'. An example of this is given in Figure 4.5.2(f).

Samples of '3', '5', and '6' are also easy to classify structurally. Samples of '3' consist mostly of two west polygons, while those of '5' would contain a east polygon above a west polygon when the top horizontal stroke is not missing. Samples of '6' are classified by a loop topped by a line segment or a east polygon with a wide opening. In this scheme, an 'eddy-type' shape with clockwise rotation and a wide opening is also assigned to the class '6'; this seems reasonable since no other character can have this type of structure.

The structural classifier is not as widely applicable to samples of the characters '2', '4', and '8' because a large number of these samples contain, at least three primitives. The character '2' has the lowest percentage (47%) of samples classified structurally. For the character '4', the problem is due partly to discontinuities at the cross point caused by skeletonization. It should also be noted that capturing the essence of this character in terms of the primitives selected has proved to be rather elusive; the top primitive, for example, can be a north, east, or west polygon depending on the way the character is written. The class '8' contains more samples with at least three primitives than any other character, partly as a result of broken loops and partly as a result of skeletonization (when a line segment is created

between the two loops). Some common samples of '2', '4', and '8' with at least three primitives are shown in Figure 4.5.1.



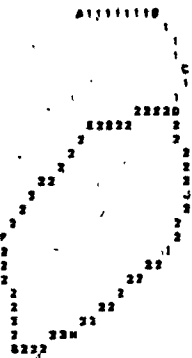
Figure 4.5.1 Samples of '2', '4', and '8' with at least three Primitives

Since structural classification is highly geometric in nature, it also conforms to the human viewpoint. With very few exceptions, samples that appear easily recognizable are correctly classified, and substitutions occur among the visually ambiguous samples.

Out of 6000 samples, 4783 were classified in this phase, and 4691 samples were correctly classified, with recognition and substitution rates of 98.08% and 1.88% respectively. Only 2 samples were rejected in this phase since most

ambiguous samples are passed on to the next phase of the classification process. Some of the substitutions are shown in Figure 4.5.2, and the confusion matrix is given in Table 4.5.1.

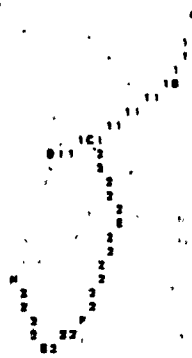
Other decision trees are given in the Appendix.



(a) '0' classified as '2'



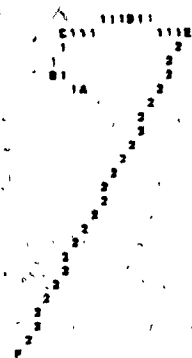
(b) '2' classified as '3'



(c) '5' classified as '3'



(d) '7' classified as '2'



(e) '7' classified as '9'



(f) '9' classified as '5'

Figure 4.5.2 Substitutions in Structural Classification

	0	1	2	3	4	5	6	7	8	9	REJECTED	SUBSTITUTED	CORRECT	TOTAL	RECOGNITION RATE
0	569	0	1	0	0	0	9	1	5	0	1	16	569	586	97.10%
1	0	600	0	0	0	0	0	0	0	0	0	0	600	600	100.00%
2	0	0	274	5	0	0	0	1	1	1	0	8	274	282	97.16%
3	0	0	0	491	0	2	0	0	0	2	0	4	491	495	99.19%
4	0	0	0	0	314	3	0	2	1	7	1	13	314	328	95.73%
5	0	0	0	0	3	1	492	0	0	1	0	7	492	499	98.60%
6	2	0	0	0	0	5	482	0	2	1	0	11	482	493	97.77%
7	1	2	2	1	3	4	0	536	0	4	0	17	536	553	96.93%
8	0	0	0	0	0	0	0	0	374	2	0	2	374	376	99.47%
9	2	0	0	1	1	5	0	0	3	559	0	12	559	571	97.90%
TOTAL	574	602	277	501	320	511	491	540	387	578	2	90	4691	4783	98.08%

RECOGNITION RATE = 98.08%

SUBSTITUTION RATE = 1.88%

REJECTION RATE = 0.04%

TABLE 4.5.1 CONFUSION MATRIX FOR STRUCTURAL CLASSIFICATION

CHAPTER 5

RELAXATION MATCHING

5.1 Introduction

A classical approach to the problem of scene and image analysis is that of template matching. It is based on the view that an image or scene can be represented by a vector or matrix of features, and that the features of an input pattern can be matched against those of the stored templates. Matching can be performed on several different levels, from comparisons of very precise corresponding locations to the higher level matching of regions. Some examples of the former are [10] and [33], and of the latter are [9] and [3].

In the template matching of features, one often overlooks the relation of a segment or subpattern to the whole. The relaxation method overcomes this neglect to some extent by considering contextual information in the matching process.

Many relaxation schemes have been proposed for image analysis tasks; different representations of the image as

well as different matching procedures have been devised. Some features used are corners of the images ([21] and [40]), edges or region boundaries of images [3], approximating ellipsoids [9], and pattern points [30]. Various relaxation matching algorithms have been proposed in [4], [15], [18], [27], and [32], for example. A comprehensive review of relaxation algorithms and their applications is given in [19].

In using the relaxation method, the features of each segment or subpattern M of a template is matched against the features of each segment I of an input or object pattern. The likelihood of the match is based on the proximity of the features, and the initial probability of this match is defined by this local information.

This initial probability is then revised according to the similarity of the context in which each segment appears. This is achieved by considering the probability of a match between the neighboring segments of M and I . The probabilities are adjusted iteratively in this manner, and associations between template and object segments are established by the highest probabilities obtained. Using this association function, a distance between the template and object patterns can be determined, and the object pattern is then classified according to the minimum distance criterion.

In this work the pattern segments used are the primitives described in Section 3.2, and the features used in the matching process are those described in Section 3.7. The relaxation matching algorithm used will be discussed in the following section. While this procedure is adapted from [43], it contains many modifications; apart from using different primitives, the matching process itself also has been adjusted according to the nature of the patterns considered here.

5.2 Relaxation Process

In this work, the basic subpatterns or primitives used in the matching process are line segments and convex polygons, and every character pattern is decomposed into a set of these primitives. Each primitive is represented by a set of features.

5.2.1 Feature Set of Primitives

Each primitive is represented by a set of features which consists of eight characteristic measurements of the primitive itself together with two lists of primitives.

The set of characteristic measurements are,

$(x_s, y_s, x_e, y_e, x_c, y_c, l, t, \theta)$, where

(x_s, y_s) are the coordinates of its starting point,

(x_e, y_e) are the coordinates of its end point,

(x_c, y_c) are the coordinates of its center,

l is the length of the line segment L joining the starting and end points,

t is the type of the primitive, and

θ is the angle from the horizontal axis to L .

The two lists of primitives are the starting and end point primitives, i.e., the primitives with end point (x_s, y_s) and the primitives with starting point (x_e, y_e) respectively.

5.2.2 Initial Probabilities

In matching primitive M_1 of a mask character against primitive I_k of an input character, suppose that M_1 has characteristic measurements

$$(x_{s1}, y_{s1}, x_{e1}, y_{e1}, x_{c1}, y_{c1}, l_1, t_1, \theta_1)$$

while I_k has corresponding measurements

$$(x_{sk}, y_{sk}, x_{ek}, y_{ek}, x_{ck}, y_{ck}, l_k, t_k, \theta_k).$$

Then I_k is considered to be associated with M_1 if the following initial conditions are satisfied:

$$(1) |\theta_1 - \theta_k| < a_2$$

$$(2) |l_1 - l_k| < a_3$$

(3) $ds_{ik}, de_{ik}, dc_{ik} < a_4$, where ds_{ik} , de_{ik} , and dc_{ik} are, respectively, the distances between the starting points, end points, and centers of M_1 and I_k . The constants a_2 , a_3 , and a_4 are parameters used to establish possible associations.

(4) $t_1 = t_k$. This condition is imposed to ensure that only primitives of the same type can be associated. The types of primitives are line segments, loops, and polygons.

If I_k is associated with M_1 , then the initial probability of the match is determined by the similarities in these measurements as follows:

$$\text{Let } d_1 = \max(|\theta_1 - \theta_k| - a_{22}, 0),$$

$$d_2 = \max(|l_1 - l_k| - a_{23}, 0),$$

$$d_3 = \max(ds_{ik} - a_{24}, 0),$$

$$d_4 = \max(de_{ik} - a_{24}, 0), \text{ and}$$

$$d_5 = \max(dc_{ik} - a_{24}, 0).$$

The parameters a_{22} , a_{23} , and a_{24} are the permissible thresholds for small variations. The quantities $d_1 - d_5$ thus measure the differences between M_1 and I_k , and the initial probability of the association of I_k to M_1 is defined by

$$p_{1(k)}^0 = \max\{1 - w_1 * (w_2 * d_1 + d_2 + d_3 + d_4 + d_5), 0\},$$

where w_1 and w_2 are assigned weights.

The values of the parameters used in this work are given below:

$$a_2 = 30, a_3 = 7, a_4 = 6,$$

$$a_{22} = 20, a_{23} = 6, a_{24} = 2,$$

$$w_1 = 0.011, w_2 = 0.2.$$

It should be noted that in [43], which deals with relaxation matching of Chinese characters, $p_{1(k)}^0$ is defined in the following way:

$$P'_{1(k)} = 1 - w_1 * (w_2 * d_1 + d_2 + d_3 + d_4), \text{ and}$$

$$P_{1(k)}^0 = \frac{P'_{1(k)}}{\sum_{k''} P'_{1(k'')}}.$$

where the summation is over all the input line segments $I_{k''}$.

associated with M_1 .

The modification made here is in accordance with the structure of the characters considered and the primitives used. In general, Chinese characters contain many strokes, and many line segments are used in their approximations. So a mask segment M_1 usually has several associated segments, and $P_{1(k)}^0$ can be obtained from the above ratio.

In the case of numeric characters and with the primitives used, a mask primitive M_1 usually has at most one associated primitive I_k in the input character. The use of the above ratio would generally result in $P_{1(k)}^0$ having a value of either 0 (when M_1 and I_k are not associated), or 1 (when they are associated). This would obliterate the information already obtained on the degree of similarity between M_1 and I_k when they are associated. The computation was therefore modified in the manner stated above in order to preserve this information.

Different values were also used here for the parameters to compensate for the more variable quality of the data set considered; some permissible thresholds were assigned larger values for this reason.

5.2.3 Iteration

Suppose primitive I_k of an input pattern is associated with mask primitive M_1 , and the initial probability $P_{1(k)}^0$ of the association has been established according to the local compatibility of the primitives described above. The relaxation process then revises this probability according to the contexts in which these primitives occur. This is an iterative process during which the starting and end point primitives of M_1 are considered. In iteration t , the revised probability $P_{1(k)}^t$, with $t \geq 1$, is obtained.

Suppose M_j is a starting point primitive of M_1 and I_n is an input primitive associated with M_j . This would require that I_n and M_j be of the same type. A function $r_{1j}^s(k,n)$ can be defined that would indicate the degree to which the matching of M_1 to I_k is compatible with the matching of M_j to I_n . It has sometimes been assumed that there are spring connections between the starting point of the pair (M_1, I_k) and the end point of the pair (M_j, I_n) [6], and that $r_{1j}^s(k,n)$ is the starting point tension. This is illustrated in Figure 5.2.1.

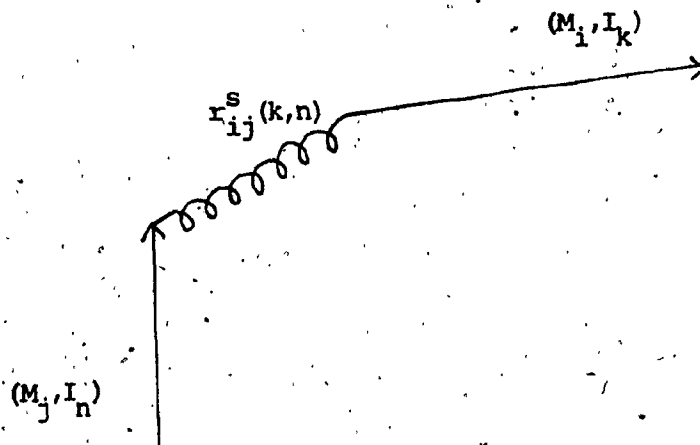


Figure 5.2.1 Spring Connections

The value of $r_{ij}^s(k, n)$ is determined by the proximity of the end point of I_n to the starting point of I_k . If this distance is denoted by d_{nk} , then

$$r_{ij}^s(k, n) = \max\{\min(1 - w_3 * d_{nk} + a_{25}, 1), 0\},$$

where a_{25} is the threshold for small variations and w_3 is the assigned weight. The fact that $r_{ij}^s(k, n)$ can be nonzero even when I_k and I_n are not neighboring primitives allows for the possible matching of characters with different numbers of primitives, or for 'missing' primitives. In this work, $w_3 = 0.3$ and $a_{25} = 0.45$; using these values, the graph of $r_{ij}^s(k, n)$ as a function of d_{nk} is shown in Figure 5.2.2.

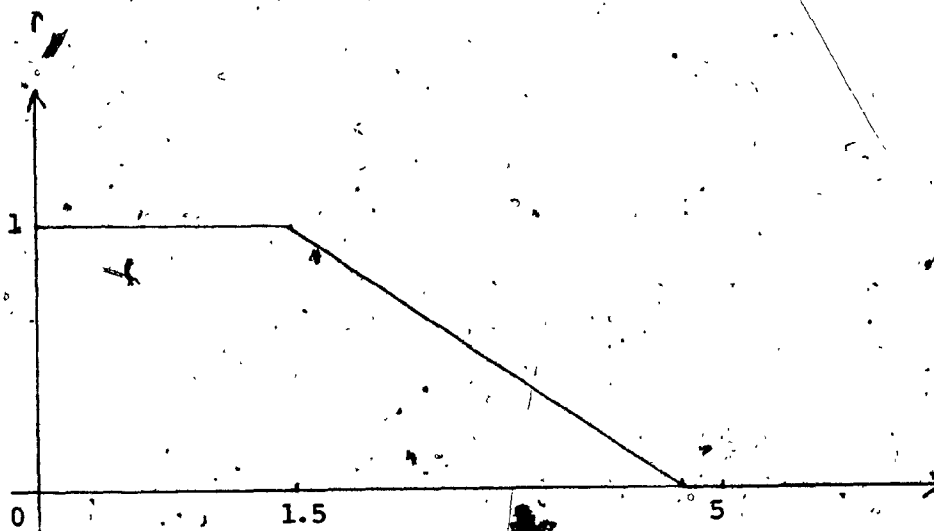


Figure 5.2.2 Graph of $r_{ij}^S(k, n)$

Analogously, if M_j is an end point primitive of M_i , and I_n is an input primitive associated with M_j , then the end point tension of the spring connection is defined by

$$r_{ij}^e(k, n) = \max\{1 - w_3 * d_{kn} + a_{25}, 1\}, 0\},$$

where d_{kn} is the distance between the end point of I_k and the starting point of I_n .

Unlike [43], when M_j is a starting point primitive of M_i , the distance between the starting point of M_i and the end point of M_j is not used here in the definition of $r_{ij}^S(k, n)$. This is due to a difference between handwritten numeric and Chinese characters. Numeric characters are

usually connected, whereas Chinese characters are not. So if M_1 and M_j are neighboring primitives in a numeric character, the distance between them can be assumed to be negligible. The same consideration was used in the definition of $r_{ij}^s(k, n)$.

For $t \geq 0$, the probability $p_{1(k)}^{t+1}$ is obtained from the following calculations:

$$p_{1(k)}^{t+1} = q_{1(k)}^s * p_{1(k)}^t \quad (5-1), \quad \text{where}$$

$$q_{1(k)} = \frac{q_{1(k)}^s + q_{1(k)}^e}{n_s + n_e} \quad (5-2),$$

$$q_{1(k)}^s = \sum_{j=1}^{n_s} \max_n \{ r_{ij}^s(k, n) * p_{j(n)}^t \}, \quad (5-3), \quad \text{and}$$

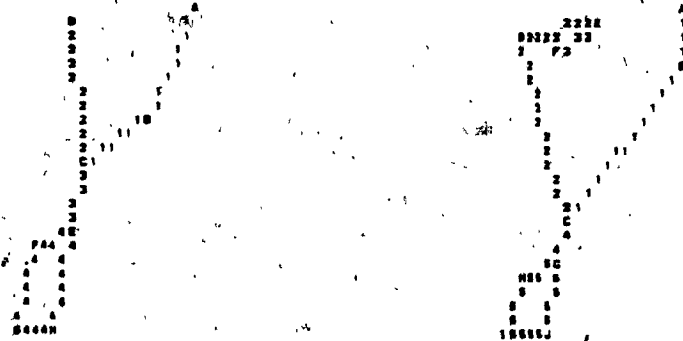
$$q_{1(k)}^e = \sum_{j=1}^{n_e} \max_n \{ r_{ij}^e(k, n) * p_{j(n)}^t \} \quad (5-4).$$

In Formula (5-2), n_s and n_e are, respectively, the numbers of the starting and end point primitives of M_1 . In Formula (5-3), each summand is the maximum value of $r_{ij}^s(k, n) * p_{j(n)}^t$, where the maximum is taken over all the primitives I_n associated with the mask primitive M_j . The summation in Formula (5-4) is analogous.

While the calculations of the probabilities can be repeated for a desired number of iterations, no noticeable

difference was observed between the correct classification rates obtained from using one or more iterations. So only one iteration was used. A similar observation was made in [18], where a relaxation method was used for point-pattern matching.

In Figure 5.2.3, a mask and an input character are shown, and the probabilities obtained in relaxation matching are given in Table 5.2.1.



(a) Mask

(b) Input Character

Figure 5.2.3 Mask and Input Character

M	I	$P_{i(k)}^0$	$P_{i(k)}^1$
1	1	0.978	0.946
3	4	0.967	1.000
4	5	1.000	1.000

All other values are equal to zero

$F(1) = 1, F(3) = 4, F(4) = 5$

Table 5.2.1 Probability $P_{i(k)}^t$

5.2.4 Distance Computation

To determine the distance D between a mask and an input character, it is necessary to calculate, for every mask primitive M_1 and every input primitive I_k associated with M_1 , the probability $P_{1(k)}^t$, where t is the desired number of iterations. Using these probabilities, an association function F can be defined so that $F(i)=i'$ denotes the input primitive that is associated with the mask primitive M_1 with the maximum probability. The distance D is obtained by using the function F . Let m be the number of mask primitives of which m_1 have associated primitives defined by F . Let m_2 be the number of unmatched input primitives.

For any mask primitive M_1 such that $F(i)=i'$ is defined, the following quantity $R_{1(i')}$ measures a degree of correlation between M_1 and $I_{1'}$, where

$$R_{1(i')} = \frac{\left[\sum_{j=1}^{ns} r^s(i', j') * P_{j(j')}^t + \sum_{j=1}^{ne} r^e(i', j') * P_{j(j')}^t \right] * P_{1(1')}^t}{ns + ne}$$

where ns and ne are the numbers of starting and end point primitives respectively of M_1 , and $F(j)=j'$.

The distance D between the mask and input characters is then given by

$$D = \frac{\sum_{i=1}^{m_1} [1 - R_{i(i')}] + (m - m_1) + m_2}{m}$$

Since $(m - m_1)$ and m_2 are the numbers of unmatched mask and input primitives, they represent additions to the distance measurement caused by 'missing' primitives.

5.2.5 Classification Criteria

For any input character, its distance from each mask can be calculated according to the procedure described in this section. However, due to the 'penalty' imposed by missing primitives in the distance measurement, it is very unlikely for the closest mask to contain a significantly different number of primitives from the input. For this reason, and also to reduce the computations involved, an input character with n primitives is only matched against masks having between $(n-1)$ and $(n+1)$ primitives.

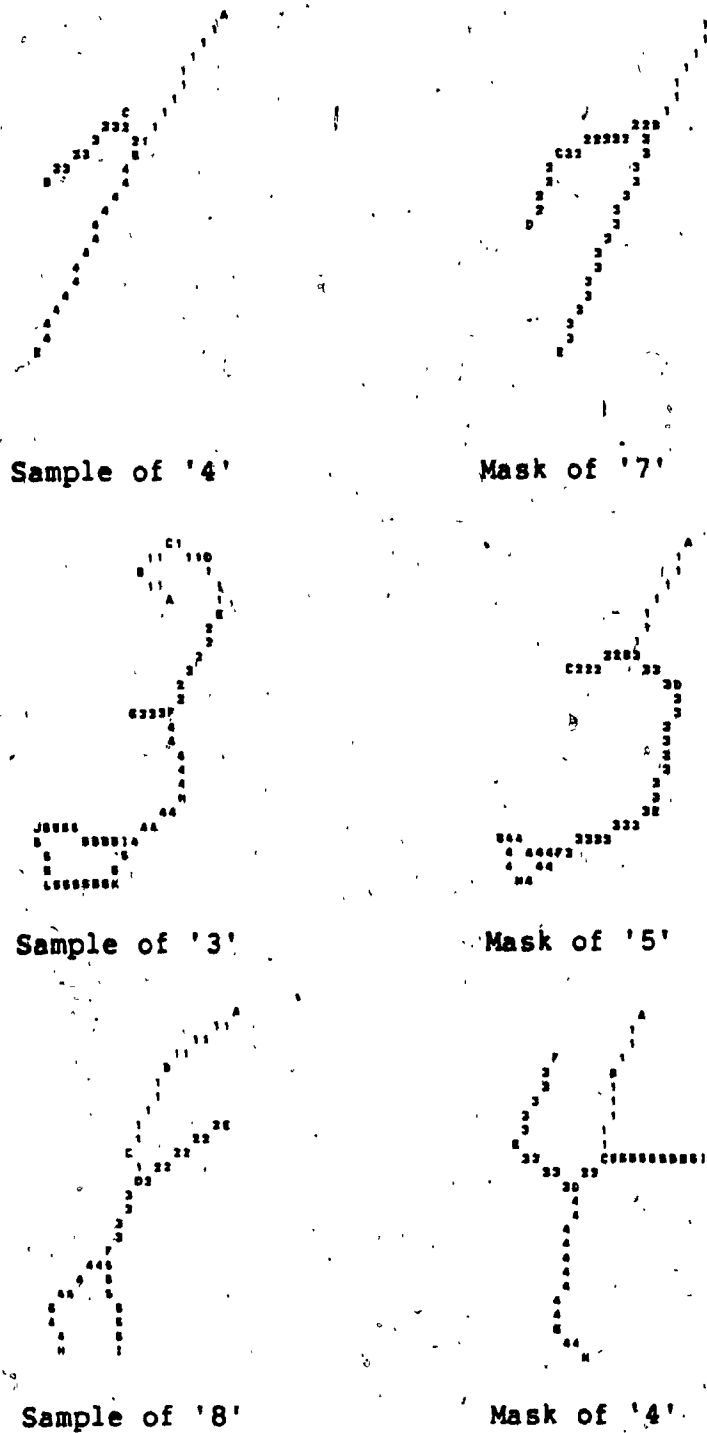
The distance of the input character from each such mask is calculated, and the two closest masks are determined. The input character is assigned to the class of the mask of minimum distance D_{\min} provided that one of the following conditions is satisfied:

(1) The two masks belong to the same class and D_{\min} does not exceed a certain upper bound U_1 .

(2) The two masks belong to different classes, the distances of the input from the two masks differ by more than a minimum value d , and D_{\min} does not exceed an upper limit U_2 .

In this work, the values established through analyses of the results are: $U_1 = 1.5$, $U_2 = 1.3$, and $d = 0.005$.

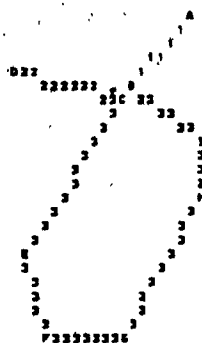
Figure 5.2.4 shows some substitutions that result from relaxation matching, and Figure 5.2.5 shows some of the samples rejected when the above criteria were applied.



(a) Input Character

(b) Closest Mask

Figure 5.2.4 Substitutions in Relaxation Matching



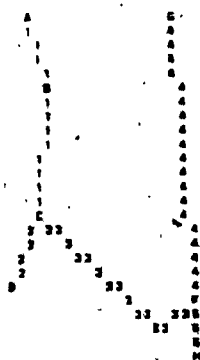
Sample of '0'



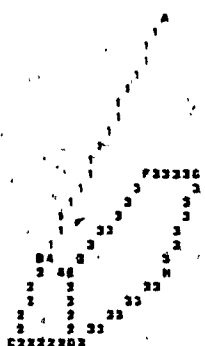
Sample of '2'



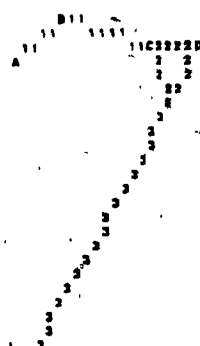
Sample of '3'



Sample of '4'



Sample of '6'



Sample of '7'

Figure 5.2.5 Samples Rejected in Relaxation Matching

5.3 Mask Generation and Training

In the training phase of relaxation matching, masks were selected and the values of the parameters were established.

In the mask selection process, a set of 165 masks were initially chosen to represent the different combinations of primitives that could constitute each character class. However, any such preliminary selection would obviously be far from complete. So, during the training process, misclassified characters in the training set were examined and some of them were added to the mask set. At the same time, masks which resulted in too many misclassifications were replaced. About 1000 characters were processed through relaxation matching in this phase, and a set of 242 masks was eventually obtained.

No fixed number of masks per character class was set in the training process; a mask was added if it was considered useful. Since relaxation matching was used here to complement structural classification, the character classes that are mostly classified by structural analysis might need fewer masks, and vice versa. This has been the case, and the number of masks per class range from 0 for the class '1' (where relaxation matching was unnecessary) to 44 for the class '2' (where, as previously explained, structural classification could only identify about 47% of the samples). The number of masks per class is given in

Table 5.3.1.

Character	0	1	2	3	4	5	6	7	8	9
# Masks	4	0	44	22	44	31	33	15	41	8

Table 5.3.1 Number of Masks per Class

Due to the distortions present in some of the samples in the training set, it was not expected that the classification would be completely correct. The use of seriously distorted samples for masks was also considered inadvisable since other misclassifications might result. The recognition rate for the training set of 819 samples was 94.02%, and the substitution and rejection rates were 3.91% and 2.07% respectively. Confusion matrices are given in Chapter 6.

5.4 Conclusion


In this work it has been observed that relaxation matching is, in most respects, a robust process. The correct recognition rate is fairly independent of small variations in the values of the weights and parameters, and it remains almost constant regardless of the number of iterations used.

Nevertheless, the essence of this method is mask selection, especially in highly variable data sets, as is the case here. For this process to function with a high degree of reliability, it is essential to have a mask set that is representative of the entire spectrum of the patterns. This is not easy to achieve when the data set can be distorted in many different ways. In such cases, it is obvious that repeated training with large sets of data would be required. The generation of masks is itself a simple process, since features are extracted by the same procedure and stored.

However, the use of a large set of masks would require a corresponding amount of storage as well as a proportional amount of computing time. The latter problem can be alleviated by taking advantage of the highly parallel nature of the relaxation process. Since all the local compatibility computations for a given iteration are independent, they can be performed simultaneously on a

suitable multiprocessor computer.

If these practical considerations are not problematic, and a comprehensive mask set is obtained, then the relaxation process should provide a consistent and reliable means of recognizing highly variable patterns. This would be especially true of the approach considered here, where primitives are selected also to preserve geometric and topological properties of the patterns.



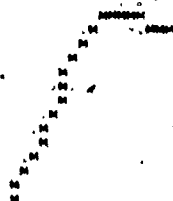
CHAPTER 6

EXPERIMENTAL RESULTS

6.1 Experimental Data

The data used in this work is a set of totally unconstrained handwritten ZIP codes collected from dead letter envelopes by the U.S. Postal Services. For this reason, the number of authors is unknown, but it can be assumed to approach the number of samples itself. The data had been digitized and binarized. The prepared character samples are accompanied by identifying labels, some of which are obviously wrong and some others are ambiguous or open to interpretation. The definitely erroneous labels were corrected, but no effort was made to eliminate any ambiguity since this was considered to be part of the nature of the data.

The samples were skeletonized, and only the skeletons were used in the recognition process. Some typical data samples and their skeletons are shown in Figures 6.1.1 and 6.1.2.



Sample of '0'

Sample of '1'

Sample of '2'

Sample of '3'



Sample of '4'

Sample of '5'

Sample of '6'

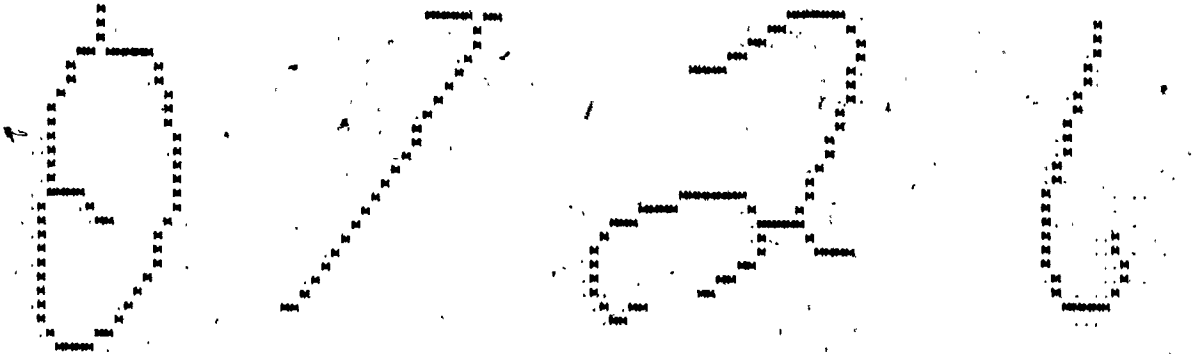


Sample of '7'

Sample of '8'

Sample of '9'

Figure 6.1.1 Typical Data Samples and Skeletons



Sample of '0'

Sample of '1'

Sample of '2'

Sample of '6'



Sample of '3'



Sample of '4'



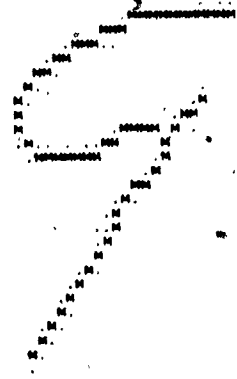
Sample of '5'



Sample of '7'



Sample of '8'



Sample of '9'

Figure 6.1.2 Additional Data Samples and Skeletons

In the training process about 5000 samples were used. Of these, about 1000 were used initially to select a set of 165 masks for relaxation matching, and the other 4000 (400 samples per class) were processed repeatedly to revise the rules for structural classification and at the same time to advance the mask selection process.

The testing set consists of 2000 samples (200 per class) not used in the training process. The assignment of samples to the training and testing data was completely random.

6.2 Results of Structural Classification

An analysis of the performance of this classifier has been presented in Section 4.5, where it was observed that the recognition rates obtained from the training and testing data are almost identical. This is probably due to the randomness of the characters, the large number of samples used in training, and the fact that the classification rules were designed to account for many possible variations in the structure of each class.

Out of 4000 samples in the training set, 3181 were classified in this phase, with recognition and substitution rates of 98.24% and 1.7% respectively. The confusion matrix is given in Table 6.2.1.

For the 2000 samples in the testing set, 1602 were classified in this phase, with recognition and substitution rates of 97.75% and 2.25% respectively. The confusion matrix is given in Table 6.2.2.

	0	1	2	3	4	5	6	7	8	9	REJECTED	SUBSTITUTED	CORRECT	TOTAL	RECOGNITION RATE
0	383	0	0	0	0	0	6	0	3	0	1	9	383	393	97.46%
1	0	400	0	0	0	0	0	0	0	0	0	0	400	400	100.00%
2	0	0	182	2	0	0	0	1	1	1	0	5	182	187	97.33%
3	0	0	0	328	0	2	0	0	0	1	0	3	328	331	99.09%
4	0	0	0	0	215	1	0	1	1	3	1	6	215	222	96.85%
5	0	0	0	3	1	318	0	0	1	1	0	6	318	324	98.15%
6	0	0	0	0	1	5	309	0	1	1	0	8	309	317	97.48%
7	0	1	0	0	1	4	0	361	0	3	0	9	361	370	97.57%
8	0	0	0	0	0	0	0	0	252	1	0	1	252	253	99.60%
9	2	0	0	0	1	3	0	0	1	377	0	7	377	384	98.18%
TOTAL	385	401	182	333	219	333	315	363	260	388	2	54	3125	3181	98.24%

89

RECOGNITION RATE = 98.24%

SUBSTITUTION RATE = 1.70%

REJECTION RATE = 0.06%

TABLE 6.2.1 CONFUSION MATRIX FOR STRUCTURAL CLASSIFICATION OF TRAINING DATA

	0	1	2	3	4	5	6	7	8	9	REJECTED	SUBSTITUTED	CORRECT	TOTAL	RECOGNITION RATE
0	186	0	1	0	0	0	3	1	2	0	0	7	186	193	96.37%
1	0	200	0	0	0	0	0	0	0	0	0	0	200	200	100.00%
2	0	0	92	3	0	0	0	0	0	0	0	3	92	95	96.84%
3	0	0	0	163	0	0	0	0	0	1	0	1	163	164	99.39%
4	0	0	0	0	99	2	0	1	0	4	0	7	99	106	93.40%
5	0	0	0	0	0	174	0	0	0	1	0	1	174	175	99.43%
6	2	0	0	0	0	0	173	0	1	0	0	3	173	176	98.30%
7	1	1	2	1	2	0	0	175	0	1	0	8	175	183	95.63%
8	0	0	0	0	0	0	0	0	122	1	0	1	122	123	98.19%
9	0	0	0	1	0	2	0	0	2	182	0	5	182	187	97.33%
TOTAL	189	201	95	168	101	178	176	177	127	190	0	36	1566	1602	97.75%

RECOGNITION RATE = 97.75%

SUBSTITUTION RATE = 2.25%

TABLE 6.2.2 CONFUSION MATRIX FOR STRUCTURAL CLASSIFICATION OF TESTING DATA

6.3 Results of Relaxation Matching

With the use of relaxation matching, there is a difference of 5.08% between the correct recognition rates obtained from the training and testing data. Of the 819 samples processed in the training set, the recognition rate is 94.02%, and the substitution and rejection rates are 3.91% and 2.07% respectively, while the corresponding rates are 88.94%, 6.78%, and 4.27% for the 398 samples processed in the testing set.

The difference in the recognition rates may be partly explained by the fact that some of the training samples were used as masks, but there may also be differences among the data sets.

The training data was processed through relaxation matching in two parts. The first set consists of 415 samples, of which 83 were eventually selected as masks, while among 404 samples in the second set, 45 were selected as masks. The recognition rates obtained from the two sets, however, are not noticeably different, as shown in the confusion matrices given in Table 6.3.1 and Table 6.3.2. In addition, the inclusion of more masks was attempted, but was found to be inconsequential. So the conclusion was that the relaxation process had largely stabilized when the testing data was processed, and that the difference in the results was due partly to the variable quality of the data sets. It

is also possible that continued training with a larger data set would improve the final results.

The confusion matrix for the testing data is given in Table 6.3.3.

	0	1	2	3	4	5	6	7	8	9	REJECTED	SUBSTITUTED	CORRECT	TOTAL
0	5	0	0	0	0	0	0	0	0	0	0	0	5	5
1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
2	0	0	93	2	0	1	1	0	0	0	0	4	93	97
3	0	0	0	29	0	0	0	0	0	0	1	1	29	31
4	0	0	1	0	97	0	0	1	1	0	2	3	97	102
5	0	0	0	1	0	35	0	0	1	0	0	2	35	37
6	0	0	0	0	1	0	44	0	2	0	0	3	44	47
7	0	0	1	0	0	0	0	9	0	0	2	1	9	12
8	0	0	1	0	1	0	0	0	75	0	1	2	75	78
9	0	0	0	0	1	0	0	0	2	2	1	3	2	6
TOTAL	5	0	97	32	100	36	45	10	81	2	7	19	389	415

RECOGNITION RATE = 93.73%

SUBSTITUTION RATE = 4.58%

REJECTION RATE = 1.69%

TABLE 6.3.1 CONFUSION MATRIX FOR RELAXATION MATCHING OF TRAINING SET 1

	0	1	2	3	4	5	6	7	8	9	REJECTED	SUBSTITUTED	CORRECT	TOTAL
0	2	0	0	0	0	0	0	0	0	0	0	0	2	2
1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
2	0	0	115	0	0	0	0	0	0	0	1	0	115	116
3	0	0	2	32	0	3	0	0	0	0	1	5	32	38
4	0	0	0	0	74	0	1	0	0	0	1	1	74	76
5	0	0	0	0	0	39	0	0	0	0	0	0	39	39
6	0	0	1	0	0	32	0	1	0	0	2	2	32	36
7	0	0	0	0	0	1	0	14	0	0	3	1	14	18
8	0	0	0	0	0	0	0	0	67	1	1	1	67	69
9	0	0	0	0	1	0	0	1	1	6	1	3	6	10
TOTAL	2	0	118	32	75	43	33	15	69	7	10	13	381	404

RECOGNITION RATE = 94.31%

SUBSTITUTION RATE = 3.32%

REJECTION RATE = 2.47%

TABLE 6.3.2. CONFUSION MATRIX FOR RELAXATION MATCHING OF TRAINING SET 2

	0	1	2	3	4	5	6	7	8	9	REJECTED	SUBSTITUTED	CORRECT	TOTAL
0	3	0	0	1	0	0	0	1	0	0	2	2	3	7
1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
2	0	40	100	2	0	0	1	0	0	0	2	3	100	105
3	0	0	2	33	0	0	0	0	0	0	1	2	33	36
4	0	0	0	0	88	1	0	0	2	0	3	3	88	94
5	0	0	0	1	0	22	1	0	0	1	0	3	22	25
6	0	0	0	0	0	0	21	0	0	0	3	0	21	24
7	0	0	0	1	0	0	1	12	0	0	3	2	12	17
8	0	0	1	1	2	0	1	0	70	0	2	5	70	77
9	0	0	0	0	3	1	0	2	1	5	1	7	5	13
TOTAL	3	0	103	39	93	24	25	15	73	6	17	27	354	398

RECOGNITION RATE = 88.94%

SUBSTITUTION RATE = 6.78%

REJECTION RATE = 4.27%

TABLE 6.3.3' CONFUSION MATRIX FOR RELAXATION MATCHING OF TESTING DATA

6.4 Overall Results

In using a combination of the two methods of structural classification and relaxation matching to classify handwritten numerals, each method is not applied to the same extent. In general, about 80% of the samples are classified structurally, and the rest by relaxation matching. This would seem to be a logical approach given the vast disparity in the computing time required by the two methods.

When implemented on a Cyber 835 computer, structural classification requires on the average 0.24 milliseconds per character, while relaxation matching requires about 662 milliseconds per character. In each case the time given is that of the recognition process alone; feature extraction (excluding skeletonization) requires approximately 80 milliseconds per character. Table 6.4.1 gives a summary of the time requirements.

Process	Time (milliseconds/character)
Skeletonization	350
Feature extraction	80
Structural classification	0.24
Relaxation matching	662

Table 6.4.1 Summary of Time Requirements

It should be noted that structural classification is used only for samples having simpler structures (at most two primitives), and that even among these samples, the samples that are more difficult to classify are passed on to the second method for recognition. So it should be expected that the second recognition phase would require more computing time. In addition, the time required by the relaxation process is directly related to the number of masks compared and the number of primitives in the input pattern. This is shown in Tables 6.4.2 and 6.4.3.

Table 6.4.2 gives the frequency distribution of the masks according to the number of primitives in the masks, and Table 6.4.3 gives the average time distribution of relaxation matching by the number of primitives in the input

character.

# Primitives	# Masks
1	3
2	39
3	116
4	61
5	18
6	4

Table 6.4.2 Frequency Distribution of Masks by
Number of Primitives

In Table 6.4.3, the number of masks compared is determined from the fact that a character with n primitives is compared to masks having between $(n-1)$ and $(n+1)$ primitives.

# Primitives	# Masks compared	Average Time(msecs)
2	158	289
3	216	649
4	195	870
5	83	568
6	22	222

Table 6.4.3 Time Distribution of Relaxation Matching

An examination of the raw data shows that the time required for relaxation matching of a character is approximately the product of its number of primitives and the number of masks compared. As shown in Table 6.4.2, a predominant number of masks contain two to four primitives, and this is also true of the samples processed by relaxation matching. So a large majority of the characters have to be compared to most of the masks, thus compounding the time required.

In light of the above observations, it is apparent that a more rapid scheme should be used to classify the easily recognizable samples, and this was achieved here by considering the same structural features that are used by

the relaxation process itself.

While each method was not utilized to the same extent across the character classes, and some classes are more amenable to structural classification than others, the use of the two methods in conjunction has resulted in generally consistent recognition rates across the character classes. For the training set of 4000 samples, the recognition rates range from 96% for the class '7' to 100% for the class '1', while in the testing set of 2000 samples the range is from 93.5% (for the classes '4', '7', and '9') to 100% for the class '1'.

In both the training and testing data, the recognition rates are lowest for the classes '7', '9', and '4'. The problems encountered in classifying samples of '4' have already been described. For the classes '7' and '9', it would appear that certain samples are written in ways lacking distinguishing characteristics. For example, it is difficult to separate some samples of '7' from those of '4' or '9', while some samples of '9' can resemble those of '5'.

For the training data, the overall recognition rate was 97.38%, and the substitution and rejection rates were 2.15% and 0.47% respectively. For the testing data, the corresponding rates were 96%, 3.15%, and 0.85%. The overall confusion matrices for the training and testing data are given in Tables 6.4.4 and 6.4.5 respectively.

Since the substitution rates appear to be rather high, new rejection thresholds were also tested in relaxation matching. By using smaller values for U_1 and U_2 (see Subsection 5.2.5), the substitution rate can be reduced and the rejection rate increased at the expense of the recognition rate.

When $U_1 = 1.1$ and $U_2 = 0.5$, the overall recognition rate for the testing data was 93.15%, while the substitution and rejection rates were 2.25% and 4.6% respectively. The confusion matrix is given in Table 6.4.6.

It is a logical consequence that the reduction of the substitution rate by 0.9% should be accompanied by a reduction of 2.85% in the recognition rate, since the method described here, when performing at its lowest level, correctly recognizes about 3 characters out of 4. Naturally, it can also be argued that the use of the new thresholds has resulted in a reliability rate of 97.64% (when the rejected samples are excluded from consideration) versus 96.82% for the original thresholds, and that in this sense the new results are preferable. Ultimately, this is one of the areas where adjustments can be made according to practical considerations.

When the same new thresholds were applied to the training data, the recognition, substitution, and rejection rates were 95.58%, 1.45%, and 2.97% respectively, and the

reliability increased from 97.84% to 98.51%. In this respect, the use of the new thresholds was more effective in reducing the substitution rate in the training set. This last result is only to be expected given the nature of the training process. Perhaps a logical next step would be to continue the training process until the diverse structures of every character class has been taken into consideration. If this were achieved, the results obtained from processing unknown data may be made to approach those of the training data itself.

	0	1	2	3	4	5	6	7	8	9	REJECTED	SUBSTITUTED	CORRECT	TOTAL	RECOGNITION RATE
0	390	0	0	0	0	0	6	0	3	0	1	9	390	400	97.50%
1	0	400	0	0	0	0	0	0	0	0	0	0	400	400	100.00%
2	0	0	390	4	0	1	1	1	1	1	1	9	390	400	97.50%
3	0	0	3	389	0	5	0	0	0	1	2	9	389	400	97.25%
4	0	0	1	0	386	1	1	2	2	3	4	10	386	400	96.50%
5	0	0	0	4	1	392	0	0	2	1	0	8	392	400	98.00%
6	0	0	1	0	2	5	385	0	4	1	2	13	385	400	96.25%
7	0	1	1	0	1	5	0	384	0	3	5	11	384	400	96.00%
8	0	0	1	0	1	0	0	0	394	2	2	4	394	400	98.50%
9	2	0	0	0	3	3	0	1	4	385	2	13	385	400	96.25%
TOTAL	392	401	397	397	394	412	393	388	410	397	19	86	3895	4000	97.38%

RECOGNITION RATE = 97.38%

SUBSTITUTION RATE = 2.15%

REJECTION RATE = 0.37%

TABLE 6.4.4 OVERALL CONFUSION MATRIX FOR TRAINING DATA

	0	1	2	3	4	5	6	7	8	9	REJECTED	SUBSTITUTED	CORRECT	TOTAL	RECOGNITION RATE
0	189	0	1	1	0	0	3	2	2	0	2	9	189	200	94.50%
1	0	200	0	0	0	0	0	0	0	0	0	0	200	200	100.00%
2	0	0	192	5	0	0	1	0	0	0	2	6	192	200	96.00%
3	0	0	2	196	12	0	0	0	0	1	1	3	196	200	98.00%
4	0	0	0	0	187	3	0	1	2	4	3	10	187	200	93.50%
5	0	0	0	1	0	196	1	0	0	2	0	4	196	200	98.00%
6	2	0	0	0	0	0	194	0	1	0	3	3	194	200	97.00%
7	1	1	2	2	2	0	1	187	0	1	3	10	187	200	93.50%
8	0	0	1	1	2	0	1	0	192	1	2	6	192	200	96.00%
9	0	0	0	1	3	3	0	2	3	187	1	12	187	200	93.50%
TOTAL	192	201	198	207	194	202	201	192	200	196	17	63	1920	2000	96.00%

RECOGNITION RATE = 96.00%

SUBSTITUTION RATE = 3.15%

REJECTION RATE = 0.85%

TABLE 6.4.5 OVERALL CONFUSION MATRIX FOR TESTING DATA

	0	1	2	3	4	5	6	7	8	9	REJECTED	SUBSTITUTED	CORRECT	TOTAL	RECOGNITION RATE
0	187	0	1	0	0	0	3	1	2	0	5	7	187	200	93.50%
1	0	200	0	0	0	0	0	0	0	0	0	0	200	200	100.00%
2	0	0	179	3	0	0	0	0	0	0	18	3	179	200	89.50%
3	0	0	2	186	0	0	0	0	0	1	11	3	186	200	93.00%
4	0	0	0	0	174	2	0	1	0	4	19	7	174	200	87.00%
5	0	0	0	1	0	191	0	0	0	1	7	2	191	200	95.50%
6	2	0	0	0	0	0	192	0	1	0	5	3	192	200	96.00%
7	1	1	2	1	2	0	0	184	0	1	8	8	184	200	92.00%
8	0	0	0	1	0	0	0	0	185	1	13	2	185	200	92.50%
9	0	0	0	1	2	3	0	1	3	185	5	10	185	200	92.50%
TOTAL	190	201	184	193	178	196	195	187	191	193	92	45	1863	2000	93.15%

RECOGNITION RATE = 93.15%

SUBSTITUTION RATE = 2.25%

REJECTION RATE = 4.60%

TABLE 6.4.6 OVERALL CONFUSION MATRIX FOR TESTING DATA (ALTERNATE REJECTION CRITERIA)

	0	1	2	3	4	5	6	7	8	9	REJECTED	SUBSTITUTED	CORRECT	TOTAL	RECOGNITION RATE
0	390	0	0	0	0	0	6	0	3	0	1	9	390	400	97.50%
1	0	400	0	0	0	0	0	0	0	0	0	0	400	400	100.00%
2	0	0	378	2	0	0	0	1	0	1	18	4	378	400	94.50%
3	0	0	1	381	0	3	0	0	0	1	14	5	381	400	95.25%
4	0	0	0	0	366	1	0	2	0	1	30	4	366	400	91.50%
5	0	0	0	4	1	379	0	0	2	1	13	8	379	400	94.75%
6	0	0	0	0	0	5	381	0	3	1	9	10	381	400	95.25%
7	0	1	0	0	1	4	0	380	0	3	11	9	380	400	95.00%
8	0	0	0	0	0	0	0	0	383	2	15	2	383	400	95.75%
9	2	0	0	0	1	3	0	0	1	385	8	7	385	400	96.25%
TOTAL	392	401	379	387	370	395	387	383	392	395	119	58	3823	4000	95.58%

RECOGNITION RATE = 95.58%

SUBSTITUTION RATE = 1.45%

REJECTION RATE = 2.97%

TABLE 6.4.7 OVERALL CONFUSION MATRIX FOR TRAINING DATA (ALTERNATE REJECTION CRITERIA)

CHAPTER 7

CONCLUSIONS AND FUTURE DIRECTIONS

It was the purpose of this work to design and implement a system to classify totally unconstrained handwritten numerals, and this aim has been largely met.

Using skeletons obtained from binarized characters, features are extracted, and the characters are classified in two phases. There is a fast structural classifier to process the easily recognizable characters, and a robust relaxation process to complete the task when no unique identification is obtained from the first phase. Essentially the same features are used in the two classification schemes, so feature extraction is performed only once and without redundancy.

The system has been tested on real-life samples of handwritten ZIP codes obtained from the U.S., and the performance is consistent. The results obtained, while not perfect, are at least on the level achieved by the human eye in deciphering handwritten characters in the absence of context, and is comparable to other results obtained in

current research on the subject (for example, [1], [5], and [7]).

Naturally, while most of the problem has been solved by these means, some of the challenge remains. Several suggestions for improvement and future research are discussed below:

(1) More types of primitives can be used to ensure a finer separation of the character classes. It is also possible to extract more features from each primitive, even though the feature set used here does contain a considerable amount of information which has not been fully utilized.

(2) Refinements and improvements can be made on the decision trees used in structural classification. They can also be extended so that more characters can be classified, or,

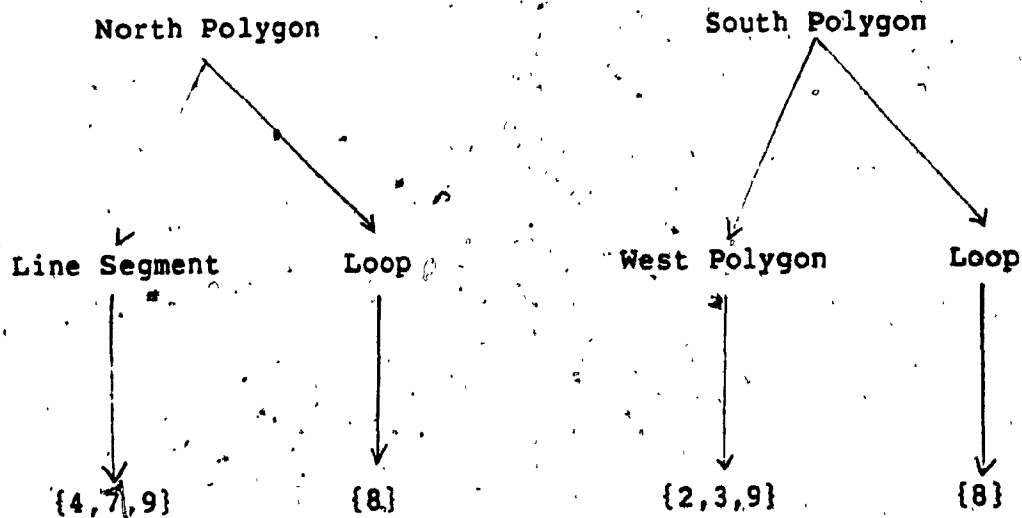
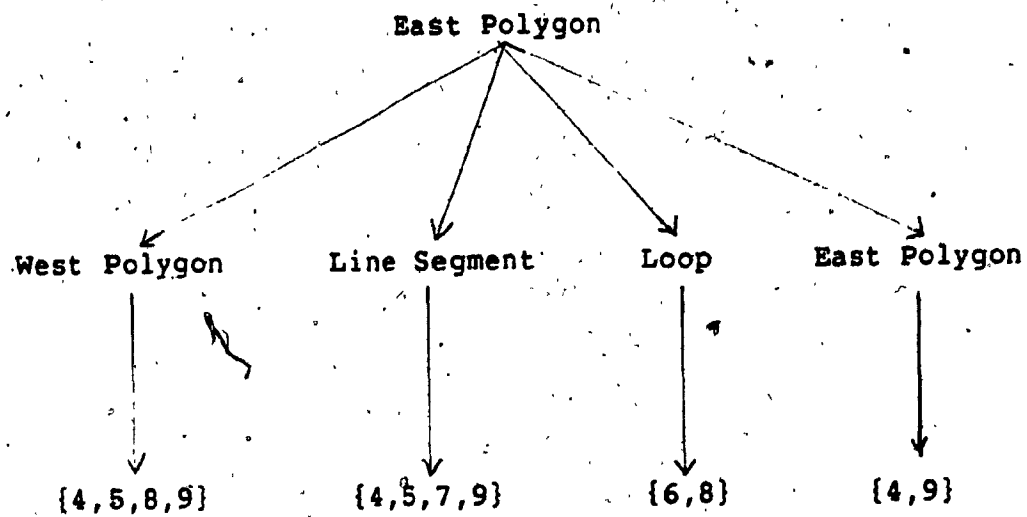
(3) Structural classification can be used to give a preliminary classification of an input character. If no unique identification can be obtained in this phase, perhaps the character can be assigned to a subset of possible classes. This is more complicated than it may appear because of the large number of possible variations in each character class. For example, it has been found that any character except '1' can contain a loop, so any reliable

preliminary classification would have to use finer partitions. If an effective procedure can be designed for this task, it would narrow the scope for the next phase of the classification process.

(4) The relaxation process can be expedited by taking advantage of its essentially parallel nature, perhaps through the use of hardware.

APPENDIX

Other Decision Trees



REFERENCES

- [1] F. Ali and T. Pavlidis, "Syntactic recognition of handwritten numerals," IEEE Trans. Syst., Man, Cybern., vol. SMC-7, pp.537-541, July 1977.
- [2] Y.K. Chu, "An alternate smoothing and stripping algorithm for thinning digital binary patterns," Master's Major Report, Dept. Of Computer Science, Concordia University, March 1985.
- [3] C.S. Clark, D.K. Conti, W.O. Eckhardt, T.A. McCulloh, R. Nevatia, and D.Y. Tseng, "Matching of natural terrain scenes," in Proc. 5th Int. Conf. Pattern Recognition, Miami, Florida, pp.217-222, Dec. 1980.
- [4] L.S. Davis, "Shape matching using relaxation techniques," IEEE Trans. Pattern Anal. Machine Intell., vol. PAMI-1, pp.60-72, Jan. 1979.
- [5] B. Duerr, W. Haettich, H. Tropsf, and G. Winkler, "A combination of statistical and syntactical pattern recognition applied to classification of unconstrained handwritten numerals," Pattern Recognition, vol. 12, pp.189-199, 1980.
- [6] M. Fischler and R. Elschlager, "The representation and matching of pictorial structures," IEEE Trans. Comput., vol. C-22, pp.67-92, 1973.
- [7] L.R. Focht and A. Burger, "A numeric script recognition processor for postal ZIP code application," in Proc. Int. Conf. Cybernetics and Society, pp.489-492, Nov. 1976.
- [8] K.S. Fu, Syntactic Pattern Recognition and Applications, Prentice Hall, Englewood Cliffs, NJ, 1982.

- [9] D.B. Gennery, "A feature-based scene matcher," in Proc. 7th Int. Joint Conf. Artif. Intell., Vancouver, B.C., Canada, pp.667-673, Aug. 1981.
- [10] N. Golshan and C.C. Hsu, "A recognition algorithm for hand-printed Arabic numerals," IEEE Trans. Syst. Sci. Cybern., vol. 6, pp.246-250, July 1970.
- [11] G.H. Granlund, "Fourier preprocessing for handprint character recognition," IEEE Trans. Comput., vol. C-21, pp.195-201, Feb. 1972.
- [12] B. Grünbaum, Convex Polytopes, John Wiley and Sons, New York, 1967.
- [13] A. Gudesen, "Quantitative analysis of preprocessing techniques for the recognition of handprinted characters," Pattern Recognition, vol. 8, pp.219-227, 1976.
- [14] J.S. Huang and K. Chuang, "Heuristic approach to handwritten numeral recognition," Pattern Recognition, vol. 19, No.1, pp.15-19, 1986.
- [15] R. Hummel and S. Zucker, "On the foundations of relaxation labeling processes," IEEE Trans. Pattern Anal. Machine Intell., vol. PAMI-5, pp.267-287, May 1983.
- [16] E. Johnson and A. Rosenfeld, "Angle detection on digital curves," IEEE Trans. Comput., vol. C-22, pp.875-878, 1973.
- [17] L. Kitchen, "Relaxation applied to matching quantitative relational structures," IEEE Trans. Syst., Man, Cybern., vol. SMC-10, pp.96-101, Feb. 1980.
- [18] L. Kitchen, "Relaxation for point-pattern matching: What it really computes," in Proc. IEEE Conf. On Computer Vision and Pattern Recognition, pp.405-407, June 1985.

- [19] J. Kittler and J. Illingworth, "Relaxation labelling algorithms - a review," Image and Vision Computing, vol. 3, No.4, pp.206-216, Nov. 1985.
- [20] Y. Mong, "An experimental study of the syntactic analysis and recognition of hand-written numerals," Master Thesis, Concordia University, Aug. 1982.
- [21] H. McRave, "Rover visual obstacle avoidance," in Proc. 7th Int. Conf. Artif. Intell., Vancouver, B.C., Canada, pp.785-790, Aug. 1981.
- [22] S. Mori and M. Doh, "A sequential tracking extraction of shape features and its constructive description," Comput. Graphics Image Processing, vol. 19, No.4, pp.349-366, 1982.
- [23] S. Mori, K. Yamamoto, and M. Yasuda, "Research on machine recognition of handprinted characters," IEEE Trans. Pattern Anal. Machine Intell., vol. PAMI-6, pp.386-405, July 1984.
- [24] H. Ogawa and K. Taniguchi, "Thinning and stroke segmentation for handwritten Chinese character recognition," Pattern Recognition, vol. 15, No.4, pp.299-308, 1982.
- [25] T. Pavlidis and S. Horowitz, "Segmentation of plane curves," IEEE Trans. Comput., vol. C-23, pp.860-870, 1974.
- [26] T. Pavlidis and F. Ali, "Computer recognition of handwritten numerals by polygonal approximations," IEEE Trans. Syst., Man, Cybern., vol. SMC-5, pp.610-614, Nov. 1975.
- [27] S. Peleg, "A new probabilistic relaxation scheme," IEEE Trans. Pattern Anal. Machine Intell., vol. PAMI-2, pp.362-369, July 1980.
- [28] E. Persoon and K.S. Fu, "Shape discrimination using Fourier descriptors," IEEE Trans. Syst., Man, Cybern., vol. SMC-7, pp.170-179, 1977.

- [29] U. Ramer, "An iterative procedure for the polygonal approximation of plane curves," Computer Graphics and Image Processing, vol. 1, pp.244-256, 1972.
- [30] S. Ranade and A. Rosenfeld, "Point pattern matching by relaxation," Pattern Recognition, vol. 12, pp.269-275, 1980.
- [31] B. Rosenberg, "The analysis of convex blobs," Comput. Graphics Image Processing, vol. 1, pp.183-192, 1972.
- [32] A. Rosenfeld, R. Hummel, and S. Zucker, "Scene labeling by relaxation operations," IEEE Trans. Syst., Man, Cybern., vol. SMC-6, pp.420-453, June 1976.
- [33] M. Shimura, "Multicategory learning classifiers for character reading," IEEE Trans. Syst., Man, Cybern., vol. SMC-3, pp.74-85, Jan. 1973.
- [34] M. Shridhar and A. Badreldin, "A high-accuracy syntactic recognition algorithm for handwritten numerals," IEEE Trans. Syst., Man, Cybern., vol. SMC-15, No.1, pp.152-158, Jan. 1985.
- [35] M. Shridhar and A. Badreldin, "Recognition of isolated and simply connected handwritten numerals," Pattern Recognition, vol. 19, No.1, pp.1-12, 1986.
- [36] A.A. Spanjersberg, "Experiments with automatic input of handwritten numerical data into a large administrative system," in Proc. Int. Conf. Cybernetics and Society, pp.476-478, Nov. 1976.
- [37] C.Y. Suen, R. Shinghal, and C.C. Kwan, "Dispersion factor: A quantitative measurement of the quality of handprinted characters," in Proc. Int. Conf. Cybernetics and Society, pp.681-685, Sept. 1977.
- [38] C.Y. Suen, M. Berthod, and S. Mori, "Automatic recognition of handprinted characters - The state of the art," Proc. IEEE, vol. 68, pp.469-487, April 1980.

- [39] C.Y. Suen, "The role of multi-dimensional loci and clustering in reliable recognition of characters," in Proc. 6th Int. Conf. Pattern Recognition, pp.1020-1022, Oct. 1982.
- [40] J.T. Tou and R.C. Gonzalez, "Recognition of handwritten characters by topological feature extraction and multilevel categorization," IEEE Trans. Comput., vol. C-21, pp.776-785, 1972.
- [41] C. Wang, H. Sun, S. Yada, and A. Rosenfeld, "Some experiments in relaxation image matching using corner features," Pattern Recognition, vol. 16, pp.167-182, 1983.
- [42] Q.R. Wang and C.Y. Suen, "Analysis and design of a decision tree based on entropy reduction and its application to large character set recognition," IEEE Trans. Pattern Anal. Machine Intell., vol. PAMI-6, No.4, pp.406-417, July 1984.
- [43] K. Yamamoto and A. Rosenfeld, "Recognition of hand-printed Kanji characters by a relaxation method," in Proc. 6th Int. Conf. Pattern Recognition, pp.395-398, Oct. 1982.