SEGMENTATION AND CLUSTERING
IN NEURAL NETWORKS
FOR IMAGE RECOGNITION

A Thesis Presented to
The Faculty of the Russ College of Engineering and Technology
Ohio University

In Partial Fulfillment
of the Requirement for the Degree
Master of Science

by
Ying-Wei Jan
June, 1994
ACKNOWLEDGMENT

I would like to thank my professor Dr. Janusz A. Starzyk for his tremendous advice and guidance in this research as well as his strong recommendations for my career.

Special thanks to Dr. Robert A. Curtis, Dr. Joseph H. Nurre and Dr. Luis C. Rabelo (in alphabetic order) for their serving on the committee and giving valuable suggestions.

Thanks to Mr. Kerry Schutz for his precious friendship and laborious assistance in writing this thesis.

Finally, I would like to dedicate this thesis to my Mom and Dad. Without their love, confidence and enduring support, my graduate studies would have been impossible.
# TABLE OF CONTENTS

1. INTRODUCTION .......................................................... 1
   1.1 Global Analysis .................................................. 2
   1.2 Structural Analysis ............................................ 5
   1.3 Combined Method ............................................. 7
   1.4 Method in This Thesis ......................................... 9

2. IMAGE PREPROCESSING .............................................. 11
   2.1 Noise Deleting via Hole Removal ............................. 11
   2.2 Thinning .......................................................... 12
      2.2.1 Literature review on thinning ........................... 13
      2.2.2 The thinning algorithm for parallel hardware ........... 16
      2.2.3 Removing pixels adjacent to NBPs ..................... 17
      2.2.4 Final thinning steps .................................... 21
      2.2.5 Hardware scheme of thinning process .................. 23
   2.3 Special Pixels Marking ....................................... 25
   2.4 Noise Deleting via Small Branch Cutting .................. 28

3. SIMILARITY MEASURE OF TWO IMAGES ............................ 34
   3.1 Image Segmentation ............................................ 36
   3.2 Locking and Scaling ........................................... 42
      3.2.1 Locking .................................................... 42
      3.2.2 Scaling .................................................... 46
   3.3 Field Generation ............................................... 47
   3.4 Rotation and Evaluation of the Minimal Average Distance ... 51
   3.5 Matching ....................................................... 59
   3.6 Calculation of the Total Average Distance D and with Penalty ... 61
   3.7 Computer Simulation Results of Matching and Distance Calculation ... 64

4. THE COMPUTER SIMULATION RESULTS ............................ 70
   4.1 Clustering ...................................................... 71
   4.2 Recognition .................................................... 73
   4.3 Testing ........................................................ 74
5. CONCLUSION ................................................................. 77

BIBLIOGRAPHY ................................................................. 83

APPENDIX:

I. Comparison of Thinning Algorithms’ Results ................................. 86
II. Cluster Centers Used Selected ...................................................... 91
III. Thinned Images ........................................................................ 97
LISTS OF TABLES

Table 3.1 Scaling and Rotation Results ........................................ 65
Table 3.2 Error Function Values of All Segments Pairs ..................... 66
Table 3.3 Unmatched Segments .................................................. 68
LISTS OF FIGURES

Fig. 1.1 A simple example of characteristic loci measures. ......................... 4
Fig. 1.2 A structural classifier. ......................................................... 5
Fig. 1.3 Illustration of Freeman chain encoding. .................................. 6
Fig. 1.4 Block diagram of Dürr's multistage method. ............................. 8
Fig. 1.5 Typical scheme of multieexpert method. ................................. 9
Fig. 2.1 An example of small hole noise. ........................................... 12
Fig. 2.2 An image and its thinned image. ........................................... 13
Fig. 2.3 Designations of the nine pixels in a 3 by 3 windows. ................. 15
Fig. 2.5 An image and its NBP image. ............................................... 17
Fig. 2.6 16 templates for pixel removal in the first thinning stage. .......... 18
Fig. 2.7 Computer simulation results of thinning process. (part I) ......... 20
Fig. 2.8 Masks for final thinning. .................................................... 22
Fig. 2.9 Computer simulation results of thinning process. (part II) ......... 23
Fig. 2.10 NBP image generation scheme. ........................................... 24
Fig. 2.11 Block diagram of the thinning hardware ............................... 25
Fig. 2.12 Pixel marking. ................................................................. 26
Fig. 2.13 Templates for detecting special pixels. ............................... 27
Fig. 2.14 Determining small branch noise. ....................................... 28
Fig. 2.15 Small branch cutting operation. ......................................... 29
Fig. 2.16 Small branch deleting. ..................................................... 29
Fig. 2.17 Templates for detecting and deleting small branches. (part I) .... 30
Fig. 2.18 Templates for detecting and deleting small branches. (part II) .... 32
Fig. 2.19 Block diagram for the second part of image preprocessing. ....... 33
Fig. 3.1 Scheme in image comparison. .............................................. 35
Fig. 3.2 Segmentation example. ....................................................... 37
Fig. 3.3 Examples of image segments. .............................................. 38
Fig. 3.4 Bi-directional searching and segmenting. ............................... 39
Fig. 3.5 Segmentation results. ......................................................... 42
Fig. 3.6 Types of segments classified by special pixels. ....................... 43
Fig. 3.7 Case one example. ............................................................. 44
Fig. 3.8 Case two example. ............................................................. 45
Fig. 3.9 Examples of case three. ...................................................... 45
Fig. 3.10 Scaling factor. ................................................................. 46
Fig. 3.11 Distance calculation for two segments. .................................. 48
Fig. 3.12 An example of the distance field of a segment. ....................... 49
Fig. 3.13 A distance field of a digit '0'. .......................................... 50
Fig. 3.14 Two locked segments and their original images. .................... 52
Fig. 3.15 The rotation angle and the rotation of a whole segment. .......... 54
Fig. 3.16 Computer simulation result of locking, scaling and rotation of segments .................................................. 58
Fig. 3.17 An example of matching. .................................................. 60
Fig. 3.18 The test image which was processed by this procedure. ............ 67
Fig. 4.1 Some examples of input images, 4 5 7 0 1. ............................. 70
Fig. 4.2 The recognition results vs. threshold value T. ........................ 75
Fig. 4.3 The network to perform the recognition. ............................... 76
Fig. 5.1 An example of the thinning process and small branch elimination. ... 80
Fig. 5.2 Some expected break points which might be difficult to find. ...... 81
Fig. I.i Thinning Results (part I) .................................................. 87
Fig. I.ii Thinning Results (part II) .................................................. 88
Fig. I.iii Thinning Results (part III) .................................................. 89
Fig. I.iv Thinning Results (part IV) .................................................. 90
CHAPTER 1
INTRODUCTION

In recent years, with the distinguished improvement of VLSI technology, personal portable electronic products, such as palmtop personal computers and PADs, (Personal Digital Assistants) have become more popular. Many surprising functions can be implemented by a few tiny chips. These chips can be accommodated in small plastic cases. We can bring these cases and many marvelous functions with us and use them whenever and wherever we need them. However, it remains inconvenient to attach a full size keyboard as an input device. For this reason, a smart and fast handwritten character recognition system is an important and useful interface between humans and computers. Not only it would replace the space consuming keyboard, but it would also act as a smart "interpreter" to translate manuscript files to computer files, especially for very huge files. Furthermore, this system would be beneficial to minimize a manual labor in many services, for instance in the postal service as most of mailing addresses are handwritten and the mail is usually sorted manually.

Handwritten character recognition has been considered a tough topic, since there are many factors which expand the varieties of handwriting styles. People from different regions, educational levels, professions, and ages have different writing styles. Even the same person won't always write the same way or use the same pen. These reasons make it a real challenge to develop a good handwritten character recognition system.
In recent decades, many laborious approaches to handwritten character recognition have been developed. Though there are a large variety of approaches, they can be roughly divided into two categories: global analysis and structural analysis.

### 1.1 Global Analysis

In this category, handwritten characters are stored as digital binary images. Many digital image processing methods can help the recognition process. Some of the common digital image processing methods are template matching, measurements of moments, characteristic loci, and mathematical transformations (Fourier, Walsh, Hadamard) [11].

In general, these methods try to find some kind of degree of similarity or distance between character images. Statistical classification methods are often included to aid the character recognition process.

Template matching is popular in printed or typed character recognition. It measures the degree of similarity between input character images and the templates. Correlation is the most common way of template matching. The weakest point of template matching is that there are too many templates required for the recognition of handwritten characters. Even though all the necessary templates can be made, the system will be very slow in handling a large number of templates. The data structure here is matrix and the main operation here is to calculate the distance of two matrices.

Hu (1962) [10] and Reiss (1991) [17] introduced the theory of moment invariance of a two-dimensional image. The moments of an image are invariant to shift, changes of
scale, rotations, or general transformations of the image. The moment of an image is good for character recognition, since a character image best represents the same character after it has been shifted or scale modified. However it is possible for rotation invariance to confuse the digit '9' and the digit '6'. The degree of similarity between two images can be measured by the difference of their moments. The moment of a image \( m_{pq} \) is defined as

\[
m_{pq} = \iint x^p y^q f(x,y) \, dx \, dy, \quad p, q = 0, 1, 2, \ldots
\]

where \( f(x,y) \) is the two dimensional image density function. For a binary digital image, we can use a binary matrix \( M \) to represent \( f(x,y) \) and each element \( M[i,j] \) is either '0' for a white point or '1' for a black point. The number of moments used depends on the complexity of the recognition problem.

The measures of characteristic loci was originally designed by Glucksman (1972) [3] for the recognition of printed characters. He put a 4-digit code on each white point in the character image. The value of each digit is equal to the count of lines from that white point to the boundary of image. Four digits is for four directions of lines counting. Fig. 1.1 shows a simple example of loci measures. A three dimensional matrix is applied to represent a character image.
The 4-digit code of that white pixel is \[ [2, 1, 0, 1] \].

**Fig. 1.1** A simple example of characteristic loci measures.

In 1982 Suen [24] adopted this measure to handwritten character recognition. Because 4-digit codes cannot represent complex handwritten characters, he increased the number of codes to 8-digit and called it multi-directional loci. He also applied a K-mean algorithm, a stepwise-optimal hierarchical clustering algorithm, and a minimum distance classification technique.

Fourier transformation has been a popular mathematical tool in signal processing. In 1972 Granlund [5] applied this technology to hand print character recognition. Regardless of the applied mathematical transformation, Fourier, Walsh or Hamamard, [11] the degree of similarity between images is determined by the coefficients of the transformations.
1.2 Structural Analysis

In this category, most of the effort is concentrated on the skeletons or the contours of handwritten characters. The essential features of handwritten characters such as loops, endpoints, junction points, arcs, concavities and convexities, and strokes are extracted. Since there are many ways to represent the extracted features and not all of the features listed above are necessary to every recognition system, there is no specific mathematical principle in this category. This is the main difference between the structural analysis and the global analysis. The crucial condition of structural analysis hence become what types of features are going to be extracted, and how to code the extracted features and the relationship between them. Structural analysis usually employs a syntactical classification approach [18]. Fig. 1.2 shows a block diagram of syntactical classification.

![Fig. 1.2: A structural classifier](image)

**Fig. 1.2:** A structural classifier
In 1970 Freeman [2] introduced a way to describe an arc named Freeman chain coding. It is based on a 3 by 3 window. This window will be put in every black point in the image as shown in Fig. 1.3 (a). The scan begins at meeting a black point then tries to find the next black point in the scan direction. Following the direction of scan, a skeleton line or a contour image can be easily coded. Using window in Fig. 1.3 (b), the line in Fig. 1.3 (a) can be represented by $[2 2 1 1 0 0]$.

**Fig. 1.3** Illustration of Freeman chain encoding: (a) example; (b) encoding rule.

The mathematical descriptions of the same extracted feature can be diverse. In 1988, Suen and Lam [27] extracted features of handwritten zip-code numbers from processed characters' skeletons. The characters' skeletons are decomposed into several primitives. A primitive can be a line or a curve. Each primitive is described by its
starting, end, and center point, primitive's length, orientation, type, and connectivity with other primitives.

Skeletons of characters were also used by Walkers et al. in 1987 [28]. In this research, a more sophisticated method was employed to extract a skeleton from a handwritten character. Each character was then represented by the number of enclosed regions and the number segments in the character's skeleton.

Image preprocessing of handwritten images, such as thinning, noise deleting and smoothening, can make feature extracting much easier. Other than improving the quality of processed images, Stringa, in 1990 [21], introduced the window-based $\mu$-transformation to compress the handwritten images. The $\mu$-transformation decreases the sizes of handwritten images and keeps images' topological features. This downsizing of processed images simplifies the recognition process.

1.3 Combined Method

Recent approaches to character recognition have tended to combine both global and structural analyses. They are called the "Multistage Classification Method" and the "Multiexpert Combination Method".

In 1980 Duerr et al. [1] introduced a four-stage classification scheme. At first all samples are processed by a conventional statistical classifier and a fast structural classifier to achieve a high recognition rate. Finally a structural hypothesis reducer and if
necessary a final heuristic matching stage are included as well. Fig. 1.4 shows the block diagram of this multistage method.

Fig. 1.4 Block diagram of Duerr's multistage method.

A multiexpert combination method processes the tested data by many methods, including both the global analysis and the structural analysis. Then a proper decision maker such as majority voting are adopted to make the final conclusion. Fig. 1.5 shows the main idea of multiexpert system. In 1992, Ching Y. Suen, Christine Nadal, Raymond Legault, Tua A. Mai and Louisa Lam [22] used 4 methods to construct a multiexpert system. Three methods extract features from the characters' skeletons while the other method uses the contours.
1.4 Method in This Thesis

The main operation of global analysis is calculation. Every method has its methodical formula where characters are represented by multidimensional arrays. No matter how the images are represented for comparison, all represented data can be processed simultaneously. That is to say, global analysis methods are array operations. On the other hand, structural analysis methods extract features from characters. They do not have a certain mathematical form. In general, processed characters are cut into several parts and then represented by their respective features and their relations to each other. Dynamic data structures such as linked list or graphics are preferred. Therefore, it is beneficial to implement global analysis methods via parallel processing hardware.

The objective of this thesis is to propose a hardware solution to handwritten digit number recognition. The developed solution is a hybrid method. The processed digit numbers are represented by two dimensional arrays of digit images. First, the images are thinned and cut into several parts. Each part contains one basic component (line, circle or...
arc) of the processed image. This stage is to structuralize the image. Then parts from two different images are compared by using global analysis. After comparison, a component matcher is used to find the degree of similarity between two processed images. The result is the essential data for training a neural network or a statistical classifier, which performs the recognition process.

In this thesis **Chapter 2** presents a parallel thinning algorithm and some image processing steps. **Chapter 3** contains image cutting and comparison. **Chapter 4** presents the analysis of the results and **Chapter 5** is the conclusion.

The data base used in this research was prepared by the National Institute of Standards and Technology in Washington DC. It covers the divergence of human writing based on the quality of print, geographical regions, social cross-section, etc. In our case, we were interested only in digits. Each image is stored in a 20 by 20 array of '1's and '0's. A '1' is a back pixel and a '0' is a white pixel.
CHAPTER 2
IMAGE PREPROCESSING

It is preferable to preprocess the images independently of the analysis type. Image preprocessing can reduce errors resulting from noisy data and speed up subsequent operations which otherwise may have been too time consuming. In this thesis, the three steps of image preprocessing are noise deleting, thinning, and marking of special pixels. Two stages of noise deleting are employed. One is used before the image is thinned and the other is used after the thinning process. Image preprocessing procedures will be discussed in the following sections.

2.1 Noise Deleting via Hole Removal

Noise (unexpected pixels and holes) may exist in the images. It is preferable to erase them before further processing. There are two types of readily apparent noises, namely small holes and small branches. Small holes may show up due to scanner error. Those holes could become circles after the thinning process. Therefore they should be filled in before the image is thinned. The motivation for applying this operation is the large number of unexpected pixels and holes we found in our data base. Fig. 1.2 (a) shows an example of a small hole. Every pixel has 8 neighboring pixels. Fig. 2.1 (b) illustrates a small hole: if a white pixel's up, down, left and right neighboring pixels are all filled with black pixels and at least two of the other neighboring positions are also filled with black pixels, then this white pixel should be filled with a black pixel.
2.2 Thinning

Since handwritten images are constructed by different width of strokes, reducing the width of strokes is a way to simplify the recognition problem. A handwritten character will not become a different character after its strokes are thinned. For this reason, the thinning process is often a welcome part of the character recognition process. Depending on the problem's complexity, a fast and reliable thinning algorithm is generally required.

Thinning is the primary process described in this chapter. The principle of thinning is to transfer an object to a set of simple arc lines or circles which lie roughly along their medial axes. In this process many redundant pixels are removed. Most of the pixels remaining after thinning connect with each other. This makes an image easy to handle and preserves the important information in the image.
Fig. 2.2 shows an example of the thinning process. Fig. 2.2 (a) is a handwritten number "8" scanned and represented by $20 \times 20$ two dimensional array. It is a digital (each location in the image can be described by integer numbers) binary (black and white) image. Fig. 2.2 (b) shows the thinned image of Fig. 2.2 (a). In this image, most pixels are simply connected one by one. Two special black points are marked by the character "Y". The usage of "Y" will be discussed in the following section.

![Fig. 2.2 (a) The original image. (b) Its thinned image.](image)

2.2.1 Literature Review on Thinning

Thinning operations can be achieved by iterative pixel deleting or nonpixel-based methods. Because pixel arrays are used to store handwritten images, only the pixel-based thinning algorithms are of concern. We found the pixel-based thinning methods belong to one of three categories (1) thinned by generating a new contour [14], (2) pixels removed by counting white to black transitions around the test pixels [26], and (3) template
Each approach applies many decision making rules to determine whether a pixel should be deleted or the region should be rebuilt. These thinning algorithms can be parallel, serial or sequential. In general, the serial and sequential algorithms make decisions combining the result of the previous operation; therefore they require less memory and the time consumption depends only on the size of the processed images. On the other hand, parallel algorithms needs more memory to save the intermediate result and time consumption is less dependent on the size of the processed images. It is said that the serial and sequential algorithms are best for single CPU applications, while the parallel algorithms take the best advantage of the parallel hardware such as processor array systems.

In 1988, Paul C. K. Kwok [14] introduced a thinning algorithm by way of contour generation. In his work, all decisions and new contour generations were done within a 3 by 3 square area. Each area has 9 pixels. There are $2^9 = 512$ area combinations possible. Each area combination is called a pattern. That is to say, if a 3 by 3 pattern fits some rules that have been pre-defined, this pattern will be changed to some other pattern. Images are thinned by changing patterns.

In 1984, Zhang and Suen [26] thinned the image by deleting pixels. The deleting rules were also made within a 3 by 3 window and the test pixel is located in the center. These rules include counting the black pixels around the tested pixel $B(P_i)$ and the number of black to white transitions around the test pixel in the ordered set $P_2, P_3, P_4$. 
P_5, ..., P_9. In addition, they checked for the existence of some special pixel around the test pixel (P_2, P_4, P_6, P_8). See Fig. 2.3 for the position marking of P_1, P_2, P_3, ..., P_9.

\[
\begin{array}{ccc}
P_9 & P_2 & P_3 \\
P_8 & P_1 & P_4 \\
P_7 & P_6 & P_5 \\
\end{array}
\]

**Fig. 2.3** Designations of the nine pixels in a 3 by 3 window.

P_1 can be deleted iff

(a) \(2 \leq B(P_1) \leq 6\)

(b) \(A(P_1) = 1\)

(c) \(P_2 \cdot P_4 \cdot P_6 = 0\)

(d) \(P_4 \cdot P_6 \cdot P_8 = 0\)

In addition, Y.S. Kim, W.S. Choi, and S.W. Kim (1992) [12] set up some templates to determine which pixels can be deleted. They used eight 3 x 3 templates, a 1 x 4 and a 4 x 1 template. For a more concise image, 8 templates are used for trimming after the images are thinned. Most pixel deleting rules try to delete a pixel in the black boundary area of an image. The thinning algorithm described in this thesis is also template based.

Implementation of thinning operations using neural network was discussed by Raghu Frishnapuram and Ling-Fan Chen in 1993 [13]. They implemented the
Rosenfeld-Kak (RK) algorithm by means of a four-stage neural network and Wang-Zhang (WZ) algorithm by means of a two stage neural network. Both algorithms (RK and WZ) also extract features of $3 \times 3$ pixel areas and perform some mathematical (numerical and logical) operations upon these features. Frishnapuram and Chen trained their neural network with $2^9$ ($= 512$) patterns.

2.2.2 The Thinning Algorithm for Parallel Hardware

As stated above, template matching is good for parallel hardware implementation. A new thinning algorithm based on template matching was developed and implemented for the use in this thesis. Each template can be implemented via PLA (programmable logic array), FPGA (field programmable gate array) or other digital hardware. There are two stages of removing redundant pixels. First stage defines non-boundary pixels (NBPs) in the processed image, and removes boundary pixels adjacent to the NBPs. The next stage involves removing other redundant pixels in order to make the image as neat as possible. The hardware for this algorithm will be designed using a sequential logic circuit, which contains memory to restore the intermediate result and combinational circuit to perform the logical operations.
2.2.3 Removing Pixels Adjacent to NBPs

Removal of the boundary pixels adjacent to NBPs is a very safe operation. It won't cause any discontinuity in the image. In this stage, a temporal NBP image is built. The building condition is shown in Fig. 2.5(a) and an example is shown in Fig. 2.5(b).

\[
\begin{array}{ccc}
A & B & C \\
D & P & E \\
F & G & H
\end{array}
\]

P is a NBP iff P is a black pixel and B D E G are all black pixels

(a)

Fig. 2.5 (a) The 3 by 3 mask for building a NBP image, (b) an image '2' and its NBP image

(b)

The 3 x 3 templates are applied to each pixel position and the center pixel of each template is examined by a logic circuit. A pixel can be removed from the image if any of the following templates are matched. Fig. 2.6 shows 16 templates used in the first thinning stage.
Each black pixel is checked by 16 templates. The removal happens if any template is matched. This process is iterated until no more pixels can be removed, since each raw image can be a new input image. There are only two kinds of images in this process: completely thinned and not completely thinned.
In this stage a temporal image is created. This temporal image consists of NBPs of the processed image. It is called an NBP image. After each thinning iteration, the NBPs will be redefined and the temporal image updated. This procedure is repeated until no more pixels adjacent to a NBP can be removed.
Updating

Fig. 2.7 The computer simulation of the three iterations of thinning process. '^
represents a pixel removed in a given iteration.

A computer simulation of the thinning operation explained above is shown in Fig.
2.7. The first column shows the input image. The second column shows the NBP image.
The third column shows an intermediate result of each iteration. The removed pixels in a
given iteration are marked by '^'. The intermediate result soon becomes the input image of
the next thinning iteration. All pixels in the original image or the results of a previous
process are processed simultaneously. This stage continues until no more pixels are
deleted.
2.2.4 Final Thinning Steps

Similar rules are applied in the final steps of the thinning algorithm, except that the non-boundary pixels are replaced by common black pixels. Black pixels are further removed in parallel by using four sets of templates and each set is applied only once. It is completed in four steps as illustrated in Fig. 2.8.

First step: deleting from the bottom side.

Second step: deleting from the right side.

Third Step: deleting from the top.

Fourth Step: deleting from the left side.
The computer simulation is shown in Fig. 2.9. Pixels removed in a given step are marked by "x". Not all of these four steps are necessary in every case. After thinning is completed, a skimmed image is obtained. There are no redundant pixels present in the skeleton processed image and the thinned image represents the necessary shape information.

<table>
<thead>
<tr>
<th>First step: Deleting from the bottom</th>
</tr>
</thead>
<tbody>
<tr>
<td>Second step: Deleting from the right side</td>
</tr>
</tbody>
</table>

Fig. 2.8 Masks for final thinning.
2.9 Computer simulation result of the final thinning process.

2.2.5 Hardware Scheme of Thinning Process

The hardware implementation of this thinning algorithm requires a sequential logic circuit because the first stage of the thinning process is iterative. That is, we need memory to store the intermediate results during the iterations. Each iteration removes pixels from the countor of the processed image. The iteration continues until no more black pixels can be removed. That is to say, the number of black pixels in the image remains unchanged.
The temporal NBP image can be produced by AND gate arrays. Each AND gate has 5 inputs. Since the images are restored by a $20 \times 20$ array, 361 ($19 \times 19$) AND gates are needed to build a NBP image. Each AND gate has its 5 inputs connected with the five pixels of processed image as shown in Fig. 2.10. The output connects to the corresponding location of the NBP image.

![Diagram of AND gates array](image)

**Fig. 2.10** NBP image generation scheme

The hardware block diagram about the thinning process is shown in Fig. 2.11. In order to employ a parallel processing hardware, one more storage device (image B) is necessary. The $3 \times 3$ templates used in two thinning stages can be implemented by PLA1 and PLA2. In real hardware design, a bus of 400 lines can be used for exchanging the data or a specialized signal processing architecture designed in which all data will be processed in parallel. In the final thinning stage, there are four steps. In Step 1 the image...
can be sent from image B to A by way of PLA2. Step 2: A to B. Step 3: B to A. Step 4: A to B. Hence, the output comes from image B.

**Fig. 2.11** Block diagram of the thinning hardware.

### 2.3 Special Black Pixels Marking

A black pixel in the thinned image which does not connect with two other black pixels is a special pixel. If it connects with more than two other black pixels, then it is a branch pixel. If it connects with only one black pixel, it is a terminal pixel. Special pixels
carry important information of the processed image. Marking special pixels was used in some structural analysis methods. In this thesis, special pixels is essential to detect noise resulting from the thinning procedure. Special pixels can also be used in image decomposition. Chapter 3 will discuss this matter further.

Fig. 2.12 shows the basic idea behind special pixels marking. There are 3 types of special pixels, T, X, and Y. A T (Terminal) type pixel connects only with one other black pixel. It gives information about the end of a line or an arc. A Y type pixel locates a y type branch point in an image. An X type pixel locates an x type branch point in an image. Some examples of special pixel markings are shown in Fig. 2.12.

The special pixels marking can be implemented by using several templates. 8 templates are needed in order to detect a T point, 16 templates for a Y and 2 templates for an X point. The templates are shown in Fig. 2.13. Thus, 26 templates are required. Just as
with the thinning template, these marking templates can be implemented by a PLA or FPGA.

Fig. 2.13 Templates for detecting special pixels.
2.4 Noise Deleting via Small Branch Cutting

The thinning process described above might produce unexpected small branches. These small branches are another source of noise that must be deleted. If a terminal pixel T is too close to a branch pixel X or Y, then this branch is a source of noise. It is too small to represent a useful feature. In this case, the special pixels marking should be performed again after the small branches are detected and deleted. A branch is between a T point and a Y point located inside a circle of radius two. There are two types of small branches being recognized in this thesis as illustrated in Fig. 2.14 (a) and (b) respectively. Each figure shows many kinds of small branches.

Fig. 2.14 Determining small branch noise.

Figs. 2.15 and 2.16 shows many possible examples of small branch noise and the computer simulation results of noise deleting.
Marking special pixels

Deleting unexpected branches

Fig. 2.16 Small branch deleting

Noise features illustrated in Fig. 2.14(a), where a T point is connected with an X or Y, are eliminated by replacing the X or Y with a common point ' # ' and taking the T away. There are eight similar conditions. All of them can be implemented by templates illustrated in Fig. 2.17.
On the other hand, the larger noise branches shown in Fig. 2.14 (b) can be found by using $3 \times 3$ templates. All common pixels ' # ' must connect with exactly 2 other black pixels, if one is a terminal pixel T and the other is a branch pixel Y or X, then ' # ' is induced in a small branch. Therefore, detection begins at putting a common pixel ' # ' at the center of 3 by 3 templates. Then its eight neighboring pixels are analyzed by using templates. If one of them is a Y or X pixel and the other one is a T pixel, there is a small branch. Fig. 2.18 shows the templates. When a small branch is detected, both the common pixel ' # ' in the center of template and the T pixel will become white pixels.
branch pixel B will be replaced by a common pixel '#'. 40 templates are needed to perform this step in parallel.

![Diagram](image-url)
Fig. 2.18 Detecting a small branch in Fig. 2.14(b).

Two types of small branches detecting and deleting can take place at the same time because two cases are exclusive. If a small branch with an X point is deleted, that X point should be marked again. An X point can become a Y point or a common pixel after small branch deleting. The special pixels are still essential to the segmentation process. The block diagram of the process which follows image thinning is shown in Fig. 2.19.

The next chapter will present the comparison of two images. The comparison is based on the output images produced by the algorithm described in this chapter.
Fig. 2.19 Block diagram for the second part of image preprocessing.
CHAPTER 3
SIMILARITY MEASURE OF TWO IMAGES

The process of thinning, special pixel marking and noise deleting provides thinned and marked images. Each thinned image contains significantly less black pixels than its original image. However, the thinned images can still represent the same handwritten character. These thinned images can be used as input data to both global analysis methods and structural analysis methods. In general, global analysis methods focus on the modes of representing images and calculate the degree of similarity between images. On the other hand, structural analysis methods usually separate processed images into several parts and investigate those part's relationships. This chapter is going to introduce a method which exploits both methods. Two images are separated into several basic parts and compared with each other shown as in Fig. 3.1 (a). In this example, there are 12 pairs of image parts that needs to be compared. Each comparison will produce an average distance d. After all possible pairs are compared, the Matcher, a match procedure, will be employed to analyze the result. As shown in Fig. 3.1 (b), this matcher will match the parts from different images and calculate the total distance D between two images. In this process, one part can only be matched with its best choice. The total distance D represents the degree of similarity between the two images. Determining the total distance between two images is the main topic of this chapter. Finding distance between two images is also the basis for image clustering and classification.
The following sections discuss the details of segmenting images, matching parts and calculating distance.
3.1 Image Segmentation

A handwritten image is composed of simple lines, arcs and circles. These lines, arcs, circles and their corresponding connections give an image its meaning. Therefore it is preferable to compare two lines, arcs or circles from different images at a time instead of two full images. For this reason, a segmenting procedure is employed. In this procedure, a handwritten image is segmented into many lines, arcs and circles.

In Chapter 2 we marked some black pixels in the thinned image with a T, X or Y. These marked black pixels offer important information about the image. A T pixel represents a beginning or an ending point of a line or arc, a Y pixel locates a branch, and an X pixel is an intersection of lines or arcs. Here we define a "segment" with these special pixels. A segment is a set of connected common black pixels ended by a special pixel(s). We also call this separation "segmentation". Fig. 3.2 shows segmentation of an image of the digit '4'. One image can be divided into many segments. In this case, each segment is ended by two special pixels, T and X. Different images may have a different number of segments.

The segmentation operation is generally simple and intuitive for humans. This section will discuss how to implement it in a computer.
In a computer, an image is stored as a two dimensional array. Different elements in this array stand for different pixels. Every pixel is independent of every other pixel. Therefore, in order to find connection relationship we check 8 neighboring positions of each black pixel. If a black pixel is found at one of the 8 neighboring positions of another black pixel, then they connect with each other. If two pixels are connected, then they belong to the same segment. We can extract every segment this way. There are many types of segments. Some segments have one terminal point and one branch point. Some segments do not have any special points. **Fig. 3.3** shows some examples we found in handwritten images. We developed an algorithm to operate on segments with different special pixels and save segments to another two dimensional array.

**Fig. 3.2** Segmentation example.
Fig. 3.3 Examples of image segments and their special pixels. (a) one segment with two Ts (b) three segments with a Y and 3 Ts (c) two segments with a Y and a T (d) one segment without any special point.

In a computer or in a parallel hardware utilizing artificial intelligence, we can search every black pixel and its connecting black pixels in an image. However, there are generally many types of segments in character images. A safe way to segment an image, is to start the search at a common black pixel ' # '. Since pixels of this kind connect with two other pixels, we must employ a bi-directional search algorithm. Fig. 3.4 shows the
idea behind bi-directional searching and segmenting. The searched pixels will be marked in order to prevent redundant searching and infinite loops in the algorithm.

\[\text{Fig. 3.4 Bi-directional searching and segmenting of the digit '6'.}\]
The segmenting algorithm has three steps:

**Step One - Beginning**: Arbitrarily choose a common pixel ' # ' then search connected black pixels in two directions (Bi-directional search). A common black pixel ' # ' always connects with two other black pixels.

**Step Two - Record and Search**: Write the searched black pixels to another 2 dimensional array. These pixels will be attributed to the same segment. Mark the searched black pixels with ' @ ' and check the eight neighboring positions of ' @ ' pixel for connected pixels. This process of searching for connected pixels of the @ pixels continues.

**Step Three, ending**: Each search is terminated in case a searched pixel is one of following:

a. ' T ' terminal pixel
b. ' X ' or ' Y ' branch pixel
c. ' @ ': a common pixel which has been previously searched, meaning this segment is a close loop without any special pixel.

A segment is found when search in both directions terminate. Repeat all three steps above until all pixels are searched and recorded. Each repetition produces a segment. **Fig. 3.5** shows the computer simulation of segmenting. **Fig. 3.5(a)**: An image representing digit '4' is decomposed into five segments. **Fig. 3.5(b)**: Another image representing digit '4' is decomposed into three segments.
(a) An image with five segments
3.2 Locking and Scaling

In comparing two images, one image is made the reference image and the other image is called the test image. After segmenting two images into several segments, as illustrated in Fig. 3.1(a), we can begin to compare these two images on the basis of their segments. The segment from the reference image is called the reference segment. The segment from the test image is called the test segment.

3.2.1 Locking

Locking is the first step of the comparison process. The objective is to exploit the transformation invariant property of handwritten characters. Segments still have the same meaning after they are moved as a rigid body. In this locking stage, two segments are put together to compute their inter-segmental pixel-to-pixel distance. The overlapping point where segments are shifted to is called the lock point.

A segment can have zero, one, or two special pixels. Segments can be roughly classified into many types by their special pixels. Fig. 3.6 shows the possible segments

(b) An image with three segments

Fig. 3.5 Segmentation Results
we found by processing our data base. In order to put different types of segments together, we should employ different principles. Here we call these principles locking rules.

Fig. 3.6 Types of segments classified by special pixels.

The lock rules depend on the type of segments compared. There are three different cases.
(1) Both segments have branch pixels (X or Y), the lock point is the branch point - see Fig. 3.7. The branch point is the original connection of this segment with other segments. If a segment has two branch points, the upper left branch point is used as the lock point. This case is the most common scenario in this thesis. This comparing also takes the most advantage of our comparison method. There are two other special cases.

![Fig. 3.7 Case one example.](image)

(2) When one segment has a branch pixel but the other segment does not, then the most top left point or upper T point on the branch pixel is placed - see Fig. 3.8.

![Fig. 3.8 Case two example.](image)
Fig. 3.8 Case two examples.

(3) If neither of segments have a branch pixel, then the top and the most left points or upper T points are placed together.

Fig. 3.9 Examples of case three.
3.2.2 Scaling

Depending on an individual's writing habits, segments with similar features are written in different sizes. By scaling the segments to an approximate size before the comparison, a desired objective can be achieved. While comparing with different segments, a segment can be differently and independently resized. This operation improves upon the shortcomings of most global analysis methods.

Fig. 3.10 illustrates the \( r_1 \) and \( r_2 \) which can roughly and easily estimate the size of the locked segments. Scaling factor \( s \) is equal to \( r_1 / r_2 \) but there is a restriction imposed on the scaling factor. It can not be larger than 1.5 nor smaller than 0.75. If segments' sizes differ too much, they could represent different features, especially in the case of two straight lines. In such a case, scaling may distort the information encoded the segments.

Fig. 3.10 Scaling factor.
In this process, the test segment is either enlarged or shrunk. The number of black pixels of the test segment is not increased when the test segment is enlarged, and is not decreased when the test segment is shrunk. That is, we don't fill additional black pixels within an enlarged segment, and we don't remove overlapping black pixels in a shrunk segment. The number of black pixels in a segment stands for the importance of that segment in its image. More important segments have more black pixels.

The scaling process is implemented in a computer by using vector computation. If the position of a lock point -see Fig. 3.10 is denoted by \([\text{lock}_x, \text{lock}_y]^T\), the position of a black pixel in the test segment is denoted by \([x, y]^T\), and the scaling factor is denoted by \(S\), then the new position of that black pixel \([x, y]^T\) can be found by

\[
\begin{bmatrix}
  x \\
  y
\end{bmatrix}
= S \cdot \begin{bmatrix}
  rx - \text{lock}_x \\
  ry - \text{lock}_y
\end{bmatrix} + \begin{bmatrix}
  \text{lock}_x \\
  \text{lock}_y
\end{bmatrix}
\]  

(3.1)

3.3 Field Generation

Thus far, the process has considered locking two segments. One of the segments is either enlarged or shrunk, and both segments have several black pixels. At this moment, we can begin to measure the similarity between these two segments. The similarity measure in this thesis was the distance from a pixel of one segment to its nearest pixel in the other segment. Fig. 3.11 illustrates the nearest distances of several
pixels. The average inter-segmental pixel-to-pixel distance $d$ between segments is defined by:

$$d = \text{average} \left( r_{tn}^2 + r_{tr}^2 \right)$$

(3.2)

where $r_{tn}$ is the minimum distance from a black pixel of the test segment to its nearest black pixel in the reference segments. $r_{tr}$ is the minimum distance from a black pixel in the reference segment to its nearest black pixel of the test segments.

![Diagram](image)

Fig. 3.11 Distance calculation for two segments.

It is not desirable to calculate all pixel-to-pixel distances and choose the smallest one every time. In addition, we are going to rotate the test segment with the locking point
as the axis. It will be very exhausting to calculate and sort the nearest distance every time.

Therefore we can create two fields associated with the black pixels of the two compared segments. Field values are the square of distances from a position to its nearest black pixel. \( F(x, y) = r^2 \). The field value of a black pixel location is equal to zero. **Fig. 3.12** shows a segment (a) and its distance field (b). The field size used in thesis is much larger than the one which is shown in **Fig. 3.12 (b)**. **Fig. 3.13** shows a distance field generated by the digit '0'.

\[
\begin{array}{cccccc}
\cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\
37 & 26 & 17 & 5 & 2 & 1 \\
36 & 25 & 16 & 4 & 1 & 0 \\
17 & 10 & 5 & 2 & 1 & 0 \\
16 & 9 & 4 & 1 & 0 & 1 \\
10 & 5 & 2 & 1 & 0 & 1 \\
9 & 4 & 1 & 0 & 1 & 2 \\
5 & 2 & 1 & 0 & 1 & 4 \\
4 & 1 & 0 & 1 & 2 & 5 \\
4 & 1 & 0 & 1 & 4 & 9 \\
4 & 1 & 0 & 1 & 4 & 9 \\
5 & 2 & 1 & 2 & 5 & 10 \\
5 & 2 & 1 & 2 & 5 & 17 \\
\end{array}
\]

**Fig. 3.12** An example of the distance field of a segment. (a) The segment, (b) It's distance field.

We generate two fields of these two segments, \( F_r \) for the reference segment and \( F_t \) for the test segment. The average inter-segmental pixel-to-pixel distance \( d \) can be evaluated as the average value of these two fields where the black pixels are located.

\[
d = \text{average} \left( r^2_r + r^2_t \right) \\
= \text{average} \left[ F_r(x_t, y_t) + F_t(x_r, y_r) \right] \quad (3.3)
\]
where \((x_t, y_t)\) is the location of a black pixel in the test segment and \((x_r, y_r)\) is the location of a black pixel in the reference segment.

This process is the most time and space consuming in our computer simulation. All fields must be large enough to cover the span area of the two compared segments and their rotation. In addition, two segments need two fields to estimate the distance to each other. These fields can be generated by a mesh resistor network for fast operation in a neural network hardware realization of this algorithm.

Fig. 3.13 A distance square field of a digit '0'.

3.4 Rotation and Evaluation of Minimal Average Distance

After locking two segments together and generating segments' distance fields, the distance between two segments, \( d \), can be calculated by

\[
d = \frac{1}{m} \sum_{i=1}^{m} F_r(x_i, y_i) + \frac{1}{n} \sum_{i=1}^{n} F_t(x_r, y_r)
\]

where \( m = \) number of black pixels in the reference segment

\( F_r = \) distance field generated by the test segment

\( x_r, y_r \): position of a black pixel in the reference segment

\( n = \) number of black pixels in the test segment

\( F_t = \) distance field generated by the reference segment

\( x_t, y_t \): position of a black pixel in the test segment

In general, this \( d \) is not always what one would expect. Some segments which have the same shape may appear a little different in their directions, especially in handwriting. To find the smallest possible distance, it is reasonable to rotate segments around the lock point. If we rotate the segments, then \( d \) changes. Hence, \( d \) is a function of relative rotation angle between these two segments, \( d(\theta) \). It is necessary to set a restriction on the rotating angle \( \theta \) (\(-\pi/3 \) to \( \pi/3 \)), because a vertical line may represent a different feature from a horizontal line, even though they are both straight lines.

**Fig. 3.14** shows the computer simulation of locking. **Fig. 3.14 (a)** shows two segments extracted from their original images and **Fig. 3.14 (b)** shows these two segments shifted and locked together. A character \( M \) represents black pixel in the
reference segment and a character P represents a black pixel in the test segment.

Character O represents the lock point.

![Diagram of two segments and their original images](image)

Fig. 3.14 Two locked segments and their original images.

The purpose of rotation is to find the smallest possible $d(\theta)$ by rotating one whole segment as if it was a rigid body. This means that every point of the rotated segment has the same rotating angle around the lock point. Since the rotating angle is
relative, it is sufficient to rotate only one segment. The rotation of a segment can be done by moving every pixel in the test segment. This movement can be implemented by applying a rotation matrix to the position of every black pixel in test segment. This rotation matrix is:

\[
\begin{bmatrix}
  x(\theta) \\
  y(\theta)
\end{bmatrix} =
\begin{bmatrix}
  \cos(\theta) & -\sin(\phi) \\
  \sin(\theta) & \cos(\phi)
\end{bmatrix}
\cdot
\begin{bmatrix}
  x - \text{lock}_x \\
  y - \text{lock}_y
\end{bmatrix}
+ \begin{bmatrix}
  \text{lock}_x \\
  \text{lock}_y
\end{bmatrix}
\tag{3.5}
\]

In equation (3.5), \(\theta\) is the rotation angle, \([\text{lock}_x, \text{lock}_y]^T\) is the position of the lock point and \([x, y]^T\) is the position of a black pixel before the test segment is rotated. \([x(\theta), y(\theta)]^T\) is the pixel position after rotation. Usually the locking point is on the center of the fields which were generated.

After the tested segment is rotated, a new distance can be found by putting new positions of black pixels which belong to the test segment in the distance field generated by the reference segment \(F_r\).

\[
r_n^2(\theta) = F_r(x_t(\theta), y_t(\theta))
\tag{3.6}
\]

On the other hand, though the test segment is rotated, it is not necessary to rotate its distance field as well. In fact, the rotation of the whole field generated by the test
segment can be replaced by the rotation of the reference segment in the opposite direction.

\[ r^2_{\pi}(\theta) = F_t(x_t(-\theta), y_t(-\theta)) \]  

Hence, the

\[ d(\theta) = \text{average} \left[ r^2_{\pi}(\theta) + r^2_{\pi}(-\theta) \right] \]  

\[
\begin{align*}
\text{Fig. 3.15} & \text{ The rotation angle and the rotation of a whole segment.} \\
\text{The algorithm for finding the minimal } d \text{ and the rotating angle } \theta \text{ is:} \\
\text{set } & \theta := 0 \quad \phi := \pi/5 \\
\text{for } & i := 1 \text{ to } 5 \text{ do} \\
& \text{mAVD} := \text{min.} \left[ d(\theta + \phi), d(\theta), d(\theta - \phi) \right] \\
& \text{if } \text{mAVD} = d(\theta + \phi) \text{ then } \theta := \theta + \phi
\end{align*}
\]
if mAVD == d (θ - φ) then θ := θ - φ

if θ > π/3 then θ := π/3 then go to end of for loop

if θ < -π/3 then θ := -π/3 then go to end of for loop

φ := φ / 1.5

end for loop (i)

Fig. 3.16 shows a computer simulation of locking, scaling and rotation. Though
the similarity of two segments was evaluated by the effort to make segments look similar,
according to the shapes of segments, restrictions on the scaling factors, and rotation
angles, the compared segment pairs do not always become alike.

the original average distance d = 2.92
after scaling and rotation

scaled and rotated segments
'o' represents an overlapping pixel

reference

scaled and rotated segments

scaling factor $S = 0.8$
rotation angle $\theta = 11.55$

average distance $d(11.55) = 0.478$
# of pixels are 9 and 11

(a) Example 1

the original average distance $d = 10.125$
after scaling and rotation

scaling factor $S = 1.1$
rotation angle $\theta = -52.88$
average distance $d(-52.88) = 1.294$
# of pixels are 9 and 8

(b) Example 2
locking

the original average distance $d = 36.5$

after scaling and rotation

scaled and rotated segments

reference
test (rotated)

scaling factor $S = 0.85$

rotation angle $\theta = -60^\circ$ (restricted)

average distance $d (-60) = 11.86$

# of pixels are 9 and 13

(e) Example 3

Fig. 3.16 Computer simulation result of locking, scaling and rotation segments.
3.5 Matching

After all possible combinations of the two segments have been processed by the procedures described above, the system evaluated the rotating angle θ, scaling factor S and average minimal inter-segmental pixel-to-pixel distance d(θ) of each pair. In order to choose the best matched pair, an error function was employed. It is a function of θ, S and d(θ). The error function represents the degree of similarity of the two segments. The larger the value of this function is the less the similarity between segments.

If \( s \geq 1 \) \[ erf(s, d, \theta) = s^2 \times (1 + \left( \frac{\theta}{3 \cdot 0} \right)^2) \times (d+1) - 1 \] (3.9)

If \( s < 1 \) \[ erf(s, d, \theta) = \left( \frac{1}{5} \right)^2 \times (1 + \left( \frac{\theta}{3 \cdot 0} \right)^2) \times (d+1) - 1 \] (3.10)

Equations (3.9) and (3.10) which measure the difference between two segments have somewhat arbitrary values. They are subject to experimental adjustment. The only concept implemented in these equations is that the larger S, θ and d(θ) the larger difference between these two segments. The error function value between two identical segments is 0.

Using the described error function between two segments, each reference segment has a similarity measure with every test segment. Each reference segment chooses a test segment for which it has the smallest error function value. If two reference segments choose the same test segment, then the reference segment which has a larger error function value has to give up its choice and choose the other test segment with the second
smallest error function value. If this reference segment still chooses the other reference segment's best choice, these two segments have to compete with each other on the basis of their error function values. The loser, the one with larger error function value, must give up its choice and take the next choice. This procedure will be reiterated until all reference segments choose different best choices. Each reference segment can choose only one test segment, and each test segment can be chosen only once. For this reason, the procedure of choosing a test segment for each reference is named matching.

Fig. 3.17 Example of matching.
For an example in Fig. 3.17, the reference image has 4 segments (marked by 1, 2, 3, 4) and the test image has 5 segments (marked by a, b, c, d, e). The best matching pairs are \{1, a\}, \{2, b\}, \{3, c\} and \{4, b\}. In this case, the reference segments 1 and 4 choose the same test segment 'b'. At this moment, compare erf(2, b) with erf(4, b). If the erf(4, b) is larger, \{4, b\} should yield and change to \{4, a\} (next choice). However, \{1, a\} has been set. Now, compare erf(1, a) with erf(4, a). If \{4, a\} still has to yield, then \{4, e\} is set. Finally, the result is \{1, a\}, \{2, b\}, \{3, c\}, \{4, e\}. Still, segment d is yet unmatched.

The unmatched segments are discussed in the following section.

3.6 Calculating of the Total Average Distance D with Penalty

After all segments are matched, the total average distance of the two images can be obtained by:

\[
D = \frac{\sum \text{# of pixels in matched segment pair } \{u,v\} \times \text{erf}(u,v)}{\sum \text{# of pixels matched segment pair } \{u,v\}}
\]  

(3.11)

where the summation is based on all match segment pairs, u is a reference segment name an v is a test segment name.

If the number of segment of these two images are not the same, then there must be some unmatched segments. In such a case, it is reasonable to give a penalty to unmatched
segments. The unmatched segments display a variation in these two images. This penalty will be added to $D$, the final result.

While giving penalties to the unmatched segments, an unmatched segment can be attributed to a redundant segment or a meaningless black pixel set. The unmatched segments must belong to images with many segments. It may be reasonable to shrink them to one pixel or make them disappear. For this reason, the penalty for an unmatched segment is the effort involved to shrink it to the lock point.

There are two types of possible unmatched segments. One type of segment contains a terminal pixel and a branch pixel or two terminal points, and the other has two branch points. The penalty, $PNT$, of the first type is given by

$$\text{PNT} = \frac{\sum_i (\text{distance from } i\text{th pixel to the lock point})^2}{\# \text{ of pixels in this unmatched segment}} \times \text{srf}$$

where the summation $i$ is based on every pixel in the unmatched segment. The $\text{srf}$ is the largest rotation and scaling factor in the error function. Segments in this category have $T$ points. A $T$ point is the beginning or ending of a segment. It has more degrees of freedom in a two dimensional space. Set $\theta = 60$ and $s = 1.5$, and we can have:

$$\text{srf} = (1+(\frac{60}{30})^2) \times 1.5^2 = 11.2$$

(3.12)
On the other hand, the penalty of the second type, a segment with two branch points, is given by:

\[
PNT = \frac{\sum_i (\text{distance from } i\text{th pixel to its nearest branch pixel})^2}{\text{# of pixels in this unmatched segment}}
\]  

(3.14)

Segments in this category have two branch pixels. A branch pixel means a connection or an intersection with another segment. The segment with two branch pixels connects other segments at both ends. Therefore it can not be rotated, shrunken, or enlarged and has less degrees of freedom. In such a case, we do not need increase its PNT by multiplying scaling and rotation factor, \( sr \).

If any unmatched segment is detected, the \( D \) should become

\[
D = \frac{\sum \# \text{ of pixels in matched segments} \times \text{erf} + \sum \# \text{ of pixels in unmatched segment} \times \text{PNT}}{\sum \# \text{ of pixels}}
\]  

(3.15)

Finding the distance \( D \) between two images was the goal of this chapter. The distance \( D \) between two images shows the degree of similarity of the compared images. Having this measure of similarity, the system can cluster together the similar images in our database.
3.7 Computer Simulation Result of Matching and the Calculation of D

This computer simulation example is comparing two images shown in Fig. 3.5. In this example, the image in Fig. 3.5 (a) is the reference image and the image in Fig. 3.5 (b) is the test one. In Table 3.1, five reference segments are named seg_a, seg_b, seg_c, seg_d and seg_e, and three test segments are denoted by seg_1, seg_2 and seg_3.
### Table 3.1 Scaling and Rotation Results

<table>
<thead>
<tr>
<th></th>
<th>test</th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>seg _1</td>
<td>seg _2</td>
<td>seg _3</td>
</tr>
<tr>
<td></td>
<td>S : 1.063</td>
<td>θ : 60</td>
<td>0.777</td>
<td>0.12</td>
</tr>
<tr>
<td></td>
<td>d : 6</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>erf : 38.54</td>
<td></td>
<td>7.88</td>
<td>73.29</td>
</tr>
<tr>
<td>seg_a</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>S : 0.80</td>
<td>θ : 11.55</td>
<td>0.1050</td>
<td>0.849</td>
</tr>
<tr>
<td></td>
<td>d : 0.478</td>
<td></td>
<td>1.294</td>
<td>22.84</td>
</tr>
<tr>
<td></td>
<td>erf : 1.65</td>
<td></td>
<td>10.54</td>
<td>128.07</td>
</tr>
<tr>
<td>seg_b</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>S : 0.8544</td>
<td>θ : -60.0</td>
<td>1.20</td>
<td>0.9</td>
</tr>
<tr>
<td></td>
<td>d : 11.87</td>
<td></td>
<td>24.29</td>
<td>0.421</td>
</tr>
<tr>
<td></td>
<td>erf : 87.15</td>
<td></td>
<td>39.40</td>
<td>0.98</td>
</tr>
<tr>
<td>seg_c</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>S : 1.265</td>
<td>θ : 33.77</td>
<td>1.333</td>
<td></td>
</tr>
<tr>
<td></td>
<td>d : 57.29</td>
<td></td>
<td>14.23</td>
<td></td>
</tr>
<tr>
<td></td>
<td>erf : 210.47</td>
<td></td>
<td>228.75</td>
<td></td>
</tr>
<tr>
<td>seg_d</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>S : 0.5</td>
<td>θ : 8.0</td>
<td>0.2727</td>
<td></td>
</tr>
<tr>
<td></td>
<td>d : 4.588</td>
<td></td>
<td>2.076</td>
<td></td>
</tr>
<tr>
<td></td>
<td>erf : 22.94</td>
<td></td>
<td>19.90</td>
<td>60.52</td>
</tr>
<tr>
<td>seg_e</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
where

\[ S: \text{scaling factor} \quad \theta: \text{rotating angle} \]
\[ d: \text{average distance} \quad \text{erf: value of error function} \]

The erf value of two segments represents the degree of similarity between them.

The matcher, according to Table 3.2, chooses the underlined pairs.

**Table 3.2 Error function values of all segments pairs**
(minimal of each row are underlined)

<table>
<thead>
<tr>
<th></th>
<th>seg 1</th>
<th>seg 2</th>
<th>seg 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>seg a</td>
<td>erf(a,1) = 39.54</td>
<td>erf(a,2) = 6.88</td>
<td>erf(a,3) = 74.29</td>
</tr>
<tr>
<td>seg b</td>
<td>erf(b,1) = 2.65</td>
<td>erf(b,2) = 11.54</td>
<td>erf(b,3) = 129.07</td>
</tr>
<tr>
<td>seg c</td>
<td>erf(c,1) = 88.15</td>
<td>erf(c,2) = 40.40</td>
<td>erf(c,3) = 1.98</td>
</tr>
<tr>
<td>seg d</td>
<td>erf(d,1) = 211.4</td>
<td>erf(d,2) = 229.7</td>
<td>erf(d,3) = 98.08</td>
</tr>
<tr>
<td>seg e</td>
<td>erf(e,1) = 23.94</td>
<td>erf(e,2) = 20.90</td>
<td>erf(e,3) = 61.52</td>
</tr>
</tbody>
</table>

The first matching result is \{a,2\}, \{b,1\}, \{c,3\}, \{d,3\}, \{e,2\}. Both seg_a and seg_e choose seg_2. Since erf(a,2) < erf(e,2), \{e,2\} yields and switches to \{e,1\}. Since erf(b,1) < erf(e,1), \{e,2\} yields and switches to \{e,3\}. The result shows erf(e,3) > erf(c,3), therefore the matching \{e,3\} should be stopped. There are no more segments that seg_e can choose. Hence, seg_e becomes an unmatched segment. For the same reason, seg_d also becomes an unmatched segment. The final matching result is \{a,2\}, \{b,1\}, \{c,3\} with seg_d, seg_e being unmatched segments.

After the matching process, the best matched segment pairs are found. Each segment that belongs to the test image is scaled and rotated. The test image is modified by the reference image and is shown in Fig. 3.18.
The average distance of matched parts of the two images is given by:

\[
D = \frac{d(a,2) \times \#(a) + \#(2) + d(b,1) \times \#(b) + \#(1) + d(c,3) \times \#(c) + \#(3)}{\#(a) + \#(2) + \#(b) + \#(1) + \#(c) + \#(3)} = \frac{221.72}{60} = 3.69
\]

\[
(3.16)
\]

\#( segment name ) is the number of pixels in this segment.

Total average distance $D$ of the matched segments is 3.69. There are still two unmatched segments remaining in the reference image. They are shown in Table 3.3. The penalties are calculated and added to $D$. 

Fig. 3.18 The test image which is processed by this procedure.
Table 3.3

**seg_e**
The self-distance is 0.33.
A small self-distance also says that this segment does not play an important role in this comparison.
Number of pixels in this image is 3.

**seg_d**
The self-distance 82.31
A large self-distance shows a large difference between these two images.
Number of pixels in this segment is 13.
PNT $= 82.312 \times 11.25 = 926$

PNT of this unmatched segment is quite large and weighted by 13 pixels.

After finding out the unmatched segments and calculating their PNTs, the PNTs are averaged with the matched segments given by.
\[ D = \frac{\Sigma D \text{ of matched segments} \times \#(\text{matched segments}) + \Sigma PNT \times \#(\text{unmatched segments})}{\text{total # of pixels}} \]  

(3.16)

where \#( matched segments) denotes the number of pixels of the matched segments, and \#( unmatched segments) denotes the number of pixels of the unmatched segments.

Finally the total average distance D is 161.05. If the test image did not contain seg_e and seg_d, shown in Table 3.3, the distance between these images would have been 3.22, a small one. That is, if an image which is similar to the test image but without segment like seg_d or seg_e, it has small distance to the reference image. While considering the PNT, seg_e is too tiny to influence the result. However, seg_d is the segment which made two images differ a lot from each other. Though both images belong to the same class '4', it will be necessary to include at least two clusters to represent this class.
CHAPTER 4
THE COMPUTER SIMULATION RESULTS

The processes of image preprocessing and image comparison discussed in Chapters 2 and 3 were simulated in the C language on a SUN™ SPARC™ workstation. 700 images were used as input data. Each image is stored in a $20 \times 20$ ASCII code array. A digit '1' represents a black pixel and a digit '0' represents a white pixel in an image. Fig. 4.1 shows five examples of input images. The 700 images are divided into 10 classes which represent handwritten digits $\{0, 1, 2, 3, 4, 5, 6, 7, 8, 9\}$. Each class has 70 different handwritten images. The 700 images were thinned, marked, segmented and compared with each other. Each comparison generated a distance between two compared images. The outputs of simulation are 490,000 real numbers.

Because a handwritten digit may be written in many different ways, even images of the same class can have large distances. On the other hand, numbers belonging to

```
00000000000001110000 00000000000001100000 00000000000001110000 00000000000001100000 00000000000001110000
00000000000001110000 00000000000001100000 00000000000001110000 00000000000001100000 00000000000001110000
00000000000001110000 00000000000001100000 00000000000001110000 00000000000001100000 00000000000001110000
00000000000001110000 00000000000001100000 00000000000001110000 00000000000001100000 00000000000001110000
00000000000001110000 00000000000001100000 00000000000001110000 00000000000001100000 00000000000001110000
00000000000001110000 00000000000001100000 00000000000001110000 00000000000001100000 00000000000001110000
00000000000001110000 00000000000001100000 00000000000001110000 00000000000001100000 00000000000001110000
00000000000001110000 00000000000001100000 00000000000001110000 00000000000001100000 00000000000001110000
00000000000001110000 00000000000001100000 00000000000001110000 00000000000001100000 00000000000001110000
00000000000001110000 00000000000001100000 00000000000001110000 00000000000001100000 00000000000001110000
```

Fig. 4.1 Some examples of input images, 4 5 7 0 1.
different classes may look similar and have small distances. For this reason, we must represent each class by several clusters. One class has many clusters and one cluster has only one cluster center. A cluster center is an image from the class it represents. The process of finding the cluster centers is called clustering.

4.1 Clustering

In this research, a supervised learning method developed by J. A. Starzyk and Sinkuo Chai (1994) [18] to cluster the 700 images was employed. Since it is a new clustering algorithm and its intermediate results are essential to the recognition process, it is preferable to describe this algorithm in this section.

For a supervised learning method, that is to say, before the 700 images were compared with each other, the class which each image belongs to had been known. The 700 images are numbered from 0 to 699. The 10 classes are numbered from 0 to 9.

At first, a cluster center in a given class is an arbitrarily chosen image. Each cluster center \( i \) that belongs to class \( C \) contributes a part of potential energy \( U_{Cj} \) to every other image \( j \), \( j \in \{ 0, 1, 2, ..., 699, i \neq j \} \). The cluster center does not contribute a part of potential energy to itself. Each image \( j \) has 10 potential energies \( U_{Cj} \), \( C \in \{ 0, 1, 2, 3, 4, 5, 6, 7, 8, 9 \} \) for 10 classes. This potential energy \( U_{Cj} \) is given by

\[
U_{Cj} = \sum_i (D_{ij})^{\gamma_i} \tag{4.1}
\]

where \( D_{ij} \) is distance from image \( j \) to cluster center \( i \), image \( i \) is a cluster center which belongs to a class \( C \),
C ∈ \{0, 1, 2, \ldots, 9\},

j ∈ \{0, 1, 2, \ldots, 699\} and i ≠ j,

Hence, each image j ∈ \{0, 1, 2, \ldots, 699\} has 10 potential energies to all classes (C ∈ \{0, 1, 2, \ldots, 9\}) generated by all cluster centers from these classes. If the j's image largest potential energy is given by cluster centers of class C, and that image is known to belong to C, then the following observation is made:

if C = C, image j is correctly classified
else, image j is not classified.

Each unclassified image can be chosen to be a cluster center. If a non-cluster center image j is shifted to be a cluster center of its class C, we must update the potential energies of all images to class C due to this change. If a cluster center image i can be correctly classified due to a newly added cluster center, that cluster center image i can be removed from its own class and become a non-cluster center image. Regardless of whether or not a cluster center is added or removed, the potential energy of each image to the changed cluster center's class will be updated. If any updating occurs, all images must be reexamined to verify that they are correctly classified. This process of adding or removing a cluster center will be repeated until all images are either correctly classified or become cluster centers. The number of cluster centers should be kept as low as possible, yet large enough to classify all images.

Using 490,000 distances between 700 images for clustering, 131 cluster centers were found to represent 10 classes. The results are nine cluster centers for class \{0\}, six
cluster centers for class \{ 1 \}, fifteen cluster centers for class \{ 2 \}, sixteen cluster centers for class \{ 3 \}, eleven cluster centers for class \{ 4 \}, eighteen cluster centers for class \{ 5 \}, fourteen cluster centers for class \{ 6 \}, eleven cluster centers for class \{ 7 \}, sixteen cluster centers for class \{ 8 \} and fifteen cluster centers for class \{ 0 \}. They are all shown in the appendix. Images in Appendix II are the cluster center images chosen by the distances between the 700 images. Appendix III shows the thinned and marked images from Appendix II.

For fast parallel processing, each cluster center needs an independent comparing unit for simultaneous distance calculation. Each comparing unit has its own memory unit for storing the distance fields of cluster center image's segments, and a computing unit to calculate inter-image distance. Thus, the number of the cluster centers is critical. The fewer the cluster centers, the less money is required to build the hardware.

4.2 Recognition

The block diagram of the hardware that performs the recognition task is shown in Fig. 4.3. The recognition of an input image is achieved by comparing the input image with all cluster centers from all classes. There are 4 layers in this parallel processing network: input layer, cluster layer, class layer and output layer.

The input layer performs the images preprocessing -- noise deleting, thinning and marking. First, the input image is thinned and marked by the fast parallel hardware shown in Fig 2.19. Then the thinned and marked image is sent to the cluster layer to compare
with 131 thinned cluster center images shown in Appendix III. The input image is the test image and the 131 cluster center images are the reference images. There are 131 independent distance measuring units in the cluster layer to perform the distance measuring simultaneously. Each unit performs the distance measuring -- segmentation, scaling, fields generation, rotation and matching. Then these 131 distances, \( \{ D_k \mid k = 1, 2, 3, \ldots, 131 \} \), between thinned image and all cluster center images are sent to the class layer. Each unit in the class layer makes the summation of inverted distances, \( \{ U_m = \sum D^{-1} \mid m = 0, 1, 2, \ldots, 9 \} \), of that class's cluster center units to calculate the input image's potential energies for the 10 classes. The output unit classifies the input image to the class that has the largest potential energy.

4.3 Testing

In order to test this network, we use another set of 800 handwritten images. These images are stored in random order. That is to say, we do not know the class that each image belongs to in advance. After comparing the test 800 images with 131 cluster center images and summing up the inverse of distances, each test image \( k \in \{ 0, 1, 2, \ldots, 799 \} \) has 10 potential energies \( U_{ck} \), \( C \in \{ 0, 1, 2, \ldots, 9 \} \) for 10 classes. Sorting \( U_{ck} \) for each test image \( k \), two winners are selected -- \( U_z \) is the \( k \)'s image largest potential energy and \( U_n \) is the second largest one. We employ a similarity measure \( M \) for the network given by.

\[
M = 1 - \frac{U_n}{U_c}
\]  
(4.2)
For each test image, if M is larger than a predefined threshold value T, then the image is accepted by the network. Comparing the simulation output with the image id records offered by National Institute of Standards and Technology in Washington DC, if the recognition result for an image is the same as the image id, this image is correctly recognized. The results of the simulation vs. different threshold values T are shown Fig. 4.2. P(A) is the acceptance rate of the network. P(R|A) is the image recognition rate (i.e. the percentage of images that are correctly classified) while the input images are accepted. If T = 0, the network acceptance rate P(A) = 100%. That is, all images are accepted by the network. However, in this case P(R|A) is only 88.5%. On the other hand, P(R|A) can be 100% by setting T=0.42, but the network acceptance rate P(A) will drop to 21.1%. The preferable region of T is from 0.03 to 0.08, in which both P(A) and P(R|A) are larger than 90%. (P(R|A)= 90.27% and P(A)=96.37% when T=0.03; P(R|A)= 92.85% and P(A)=91.0% when T=0.08.)

Fig. 4.2 The recognition results vs. threshold value T.
Fig. 4.3 The network to perform the recognition.
CHAPTER 5
CONCLUSION

This thesis introduced a fast parallel hardware implementable thinning method and defined a "distance" between two thinned images by comparing their segments. Seven hundred classified handwritten images from 10 different classes were used as the training data, and were grouped into several clusters based on the distances between the images. Each class has several cluster centers. After finding cluster centers of all classes, a new handwritten image is characterized by 10 potential energies for 10 classes. Then this new image can be recognized by finding its largest potential. This method is applicable for recognition of other black and white images that can be described by lines, arcs and circles. A number of classes is defined by the recognition problem. That is to say, 10 classes are required for digits recognition \{ 0, 1, 2, \ldots, 9 \}, 26 classes are required for recognition of English characters \{ a, b, \ldots, z \}. A cluster center in a class represents a standard, common, or paradigmatic pattern in the class. In the training process, especially a supervised learning used in this thesis, if a training sample can not find any similar cluster center in this sample's class, then this sample can not be correctly classified. In such a case, the training and clustering process will make this non-classified sample become a new cluster center in the sample's class. That is, the number of cluster centers used to represent classes depends on the complexity, variety the training data and the number of training samples.
In this thesis, 700 handwritten images are used as the training data. It is difficult to say, if using the 131 cluster centers found from 700 training samples one can recognize all handwritten digits on our planet. In order to make a more powerful recognition system, it is desirable to use more training data to train the system, and make the system more experienced and sophisticated. However, more training data may complicate the clustering process and arouse the appearance of more cluster centers. The clustering result examines the definition of the distance between two images. The number of cluster centers is lowered by using better way two measure the distance between images. One the other hand, for a fast parallel hardware in Fig. 4.2, the distance measures are performed at the same moment in order to achieve a fast response of the recognition process. Hence, each cluster center needs an individual hardware unit to perform the distance measurements. It means, that more cluster centers need more devices in VLSI circuits. More devices increases the difficulty of VLSI design and the manufacturing cost. Hence, before finding more training data and more sophisticated clustering algorithms, it is important to consider the image preprocessing and comparison again.

In this research, the handwritten images were analyzed by their strokes or segments. Doubtlessly, these strokes are the principal part of a handwritten image. In order to emphasize the strokes and separate images into few segments, the images were thinned and their special pixels were marked by using some 3 x 3 templates before the images are compared. In the image thinning process, some small branches that either had been in the row image or were generated during the thinning process were removed to
achieve a smoother and clearer image. After the thinning process, the image can offer a clear stroke information. Though this process is sufficient for most images, in some cases, as illustrated in Fig. 5.1, a few branch points and some segments could be lost in the noise deleting stage. The deleted branch shown in Fig. 5.1 b and c is too small to influence the result of comparison. However, one important branch pixel disappeared in the image preprocessing. Although the image only lost two black pixels, an essential branch point that can break this image into two segments was lost. A branch point is an essential clue to segment the image. This image thus has only one segment. Losing this segmentation causes this image to have a "big" distance with respect to other images in class {9}. This phenomenon makes the clustering process more complicated, and it increases the number of clustering centers. Nevertheless, there is still no simple way to distinguish which small branches are caused by our thinning process and which small branches are important for the segmentation process. Otherwise, an additional checking procedure should be added to every iteration in thinning process to decide whether a small branch should be deleted or not. However, adding this procedure will slow down the thinning process, and make the system much more complicated. Fortunately, this is only a special case, which does not occur very often. Most of the checking to decide whether small branches should be deleted or not is unnecessary.
Fig 5.1 An example of the thinning process and small branch eliminating.

One could argue that perhaps it is not necessary to keep the small branch which disappeared in Fig. 5.1 (c). The image in Fig. 5.1 (c) can still be divided into two segments by finding a way to detect where the strokes change their directions abruptly. This method is not only applicable to the digit '9' but it can also find the abrupt change in the stroke direction in the middle of the digit '3', bottom part of '2', upper parts of '7' and '5' - see Fig. 5.2. It is not always easy to find branch points which are marked by small squares in Fig. 5.2. If a reasonably simple method can be developed for machines to detect these kinds of points, all the separated segments become just simple lines, arcs or circles.

The images will be broken at the place where the strokes change their directions abruptly as well as at branch points. A point like this will be called a "break point". The clustering process is expected to be simplified if only simple segments or segment primitives in the images are compared.
Fig 5.2 Some expected break points which might be difficult to find.

Detecting the break points in Fig 5.2 is included in the plans for future work. If a hardware approach utilizing parallel processing is considered, several $3 \times 3$ templates are used to find the break points. Unfortunately, all break points cannot be found by using this method. In addition, some regular points would appear the same as break points viewed from $3 \times 3$ templates. Perhaps, some larger templates or global methods are needed in order to extract break points.

The segmenting process discussed in Chapter 3 searches for black pixels which are connected one by one, and then stores these connected black pixels into an individual image. This is a fully sequential operation but the majority of segments contain at most 20 black pixels. Even the largest and the most complex segmenting operation will not
cost much additional time. The segmentation process is planned to be implemented by software.

The generation of distance fields for each segment in Chapter 3 is computationally expensive. It is the bottleneck stage in the distance measuring operation, and in the overall recognition process. For this reason, it is preferable to design a real time hardware to generate the distance field. In the design, the distance fields of all cluster center images' segments are restored in the memory of the network. The more cluster centers are required by the system, the more storage devices are needed to store the distance fields. The system needs only to generate distance fields for input images when it is operating. Therefore, the system operates as if it were generating only distance fields of the input images.

The matching process and the calculation of total average distance discussed at the end of Chapter 3 only processes under 100 numerical data. It is not a time consuming stage and neither is segmentation process. It is also planned to implement this stage in software.

The training data in this thesis do not cover all handwriting styles. More handwritten images for training and testing will help the research, and will make the system recognize more handwriting styles. It is also an interesting problem to discover how on earth is the degree of similarity between handwritten images defined in human's brain.
BIBLIOGRAPHY


[12] Kim, Y. S., Choi and S. W. Kim, S.W. High-Speed Thinning Processor for


[19] Starzyk, Januzs and Chai, Sinkuo. Supervised Learning with Potentials for Neural Network-based Object Recognition, Proceeding of 26th Southeastern Symposium on System Theory, March 1994, Ohio University, Athens, Ohio


APPENDIX I

Comparison of Thinning Algorithms' Result
In "A thinning algorithm by contour generation" by Paul C. K. Kwok, [13], there is comparison of five kinds of thinning algorithms. We copied the comparison results, Fig. I.i, I.iii and compared with our result Fig. I.ii, I.iv.

![Chinese Character](image)

**Fig. I.i**

Fig. I.i is a $128 \times 128$ Chinese Character consists of three disjoint objects (a) The original character consists (b) SPTA (c) Zhang / Seun Algorithm (d) Xu / Wang's CGT (e) Pavlidis's Contour Tracing (f) Contour Generation with chain codes.
Fig. I.i Thinning result of a $128 \times 128$ Chinese Character.
Fig. I.iii is a 128 × 128 Bold old English B (a) The original character consists of two disjoint objects, one with three holes (b) SPTA (c) Zhang / Seun Algorithm (d) Xu / Wang's CGT (e) Pavlidis's Contour Tracing (f) Contour Generation with chain codes.
Fig. I.iv Thinning result of a $128 \times 128$ Old-English B
APPENDIX II

Clustering Centers Selected
APPENDIX III

Thinned Images of Cluster Centers