DYNAMIC SELECTION OF PRIMITIVES IN THE METRIC MODEL

by

Sonya Dewi

TR90-059, November 1990

This is an unaltered version of the author's
M.Sc.(C.S.) Thesis

Faculty of Computer Science
University of New Brunswick
P.O. Box 4400
Fredericton, N.B. E3B 5A3

Phone: (506) 453-4566
Fax: (506) 453-3566

Abstract

This thesis has three main objectives. First, to initiate the investigation of the new degrees of freedom offered by the recently proposed pattern recognition model, specifically to the process of selection of the low-level primitives for structural pattern representation. Second, to suggest that not only should the latter process not be separated from the (high-level) recognition process, as has been practised so far, but, moreover, it should be driven by it. Third, to continue the investigation of the primitives selection model proposed by Muchnik (1974).

The metric model for pattern recognition recently proposed by Goldfarb(1990) makes use of the concept of distance to classify the patterns during the learning stage. The optimal value of the objective function used in the learning process can also be used in a feedback loop to measure the appropriateness of the low level primitives chosen by the Muchnik model. An implementation of this model for the handwritten digits is presented. The results of the experiments together with the suggestions for future work are also presented.

Contents

Abstract ii

List of Tables v

List of Figures vi

Acknowledgements xi

1 Introduction 1

2 The Muchnik Method of Primitive Selection 6

2.1 Introduction 6

2.2 Eye movement and fixation points 7

2.3 The Muchnik method of primitive selection 9

2.3.1 An image transformation 9

2.3.2 The search for image fragments 11

2.4 Experimental results 16

3 Levenshtein Transformation System in Metric Model 25

3.1 Introduction 25

3.2 Levenshtein Transformation System 26

3.3 Learning model for the transformation system 28

3.4 Weight learning for handwritten digit training patterns 35

3.4.1 Example one 36
3.4.2 Example two 36
3.4.3 Example three 40
3.4.4 Example four 43

4 Dynamic selection of primitives in the metric model 46
4.1 Introduction 46
4.2 The existing visual pattern recognition systems 47
4.3 Proposed dynamic system for primitives selection 48
4.4 On the reasons why the construction of the binary decision tree is possible 55
4.5 The Experimental Results 56

5 Conclusion and Future Work 69

References 71

Appendices

List of Tables

3.1 Clusters of fragments for the training sets of figure 3.8 36

3.2 Clusters of fragments for the training sets of figure 3.10 38

3.3 Clusters of fragments for the training sets in figure 3.12 42

3.4 Clusters of fragments for the training sets of figure 3.14 43

4.1 Clusters of fragments for the training sets of figure 4.7 and 4.8 60

4.2 Clusters of fragments for the training sets of figure 4.10 and 4.11 . . . 63

List of Figures

1.1 The general block diagram of syntactic pattern recognition [6], pp. 9). 3

2.1 Diagram of visual perception through eye movement theory [7] 8

2.2 (a) Defocusing mask DM of size 13×13 for equation 2.1 with $\alpha = 3$. The DM is always computed by discretizing and pixel reindexing at a necessary resolution the fixed size square $p \in (-3,-3) \times (3,3)$, $p_0 = (0,0)$. (b) Plot of (a). 10

2.3 Algorithm for applying the defocusing mask. 11

2.4 (a) The original image of digit 2, (b) The 3D plot of the defocused image of (a) using the mask of figure 2.2 , (c) The 3D plot of the smoothed image of (b) with $S = 9$ 12

2.5 Algorithm for searching the local extrema. 14

2.6 Two procedures for the algorithm in figure 2.5 15

2.7 (a) The original handwritten digit 2, (b) the functional image representation for (a) obtained by using the defocusing mask DM of size 13×13 , (c) the image and the corresponding fragments for the search window WL of size 9×9 17

2.8 (a) The original handwritten digit 3, (b) the functional image representation for (a) obtained by using the defocusing mask DM of size 13×13 , (c) the image and the corresponding fragments for the search window WL of size 9×9 18

2.9 (a) The original handwritten digit 5, (b) the functional image representation for (a) obtained by using the defocusing mask DM of size 13×13 , (c) the image and the corresponding fragments for the search window WL of size 9×9 19

2.10 (a) The original handwritten digit 8, (b) the functional image representation for (a) obtained by using the defocusing mask DM of size 13×13 , (c) the image and the corresponding fragments for the search window WL of size 9×9 21

2.11 (a) The original handwritten digit 8, (b) the functional image representation for (a) obtained by using the defocusing mask DM of size 17×17 , (c) the functional image representation for (a) obtained by using the defocusing mask DM of size 9×9 22

2.12 (a) The functional image representation for digit 8 obtained by using defocusing mask DM of size 17×17 , (b) the image of digit 8 and the corresponding fragments for the search window WL of size 17×17 , (c) the image of digit 8 and the corresponding fragments for the search window WL of size 13×13 23

2.13 Image of digit 8 with the corresponding fragments obtained by using the defocusing mask DM of size 17×17 and the search window WL of size 17×17 24

3.1 Examples of insertion and deletion operations. 26

3.2 Algorithm for computing the distance between two strings. 29

3.3 Algorithm for learning in the metric model. 32

3.4 Algorithm for optimizing $f(\omega)$ 33

3.5 Algorithm for computing the intrinsic intraclass distance. 34

3.6 Algorithm for computing the intrinsic interclass distance. 34

3.7 A binary decision tree generated by the weight learning process. . . . 35

3.8 (a) The images and the corresponding image fragments of the first training set, (b) the images and the corresponding image fragments of the second training set. Image fragments of (a) and (b) are obtained by using defocusing mask, searching window and cutting window of size 13 × 13 37

3.9 Binary decision tree for the training sets of figure 3.8 and table 3.1 . . 38

3.10 (a) The images and the corresponding fragments of the first training set,(b) the images and the corresponding fragments of the second training set. Image fragments of (a) and (b) are obtained by using defocusing mask of size 13 × 13 , searching window of size 5 × 5 and cutting window of size 17 × 17 39

3.11 Binary decision tree for the training sets of figure 3.10 and table 3.2 . 40

3.12 (a) The images and the corresponding image fragments of the first training set, (b) the images and the corresponding image fragments of the second training set. Image fragments of (a) and (b) are obtained by using defocusing mask, searching window and cutting window of size 13 × 13 41

3.13 Binary decision tree for the training sets of figure 3.12 and table 3.3 . 42

3.14 (a) The images and the corresponding image fragments of the first training set, (b) the images and the corresponding image fragments of the second training set. Image fragments of (a) and (b) are obtained by using defocusing mask, searching window and cutting window of size 13 × 13 44

3.15 Binary decision tree of training sets of figure 3.14 and table 3.4 45

4.1 Functional block diagram of an adaptive pattern recognition system [19] pp. 16 49

4.2 Block diagram of the dynamic selection of primitives [9] pp. 22. . . . 50

4.3 Tracking direction and the initial position of encoding of each four digit. 51

4.4 Algorithm for dynamic system of primitive selection. 53
4.5 Algorithm for object recognition. 54
4.6 Algorithm for computing the matrix distance among fragments. . . . 55
4.7 (a),(b) The images and the corresponding primitive fragments obtained by using defocusing mask of size 9×9 , search window of size 5×5 and cutting window of size 17×17 for training sets $D2$ and $D3$ respectively. 58
4.8 (a),(b) The images and the corresponding primitive fragments obtained by using defocusing mask of size 9×9 , search window of size 5×5 and cutting window of size 17×17 for training sets $D5$ and $D8$ respectively. 59
4.9 The final binary decision tree for the training sets of figure 4.8 60
4.10 (a),(b) The images and the corresponding primitive fragments obtained by using defocusing mask of size 13×13 , search window of size 13×13 and cutting window of size 13×13 for training sets $D2$ and $D3$ respectively. 61
4.11 (a),(b) The images and the corresponding primitive fragments obtained by using defocusing mask of size 13×13 , search window of size 13×13 and cutting window of size 13×13 for training sets $D5$ and $D8$ respectively. 62
4.12 An intermediate binary decision tree for the training sets of figure 4.10 and 4.11 63
4.13 (a) Three digits 2 with the corresponding primitive fragments obtained by using the final parameters for selection and the corresponding string representations, (b) The binary tree with the corresponding minimum distances. 65
4.14 (a) Three digits 3 with the corresponding primitive fragments obtained by using the final parameters for selection and the corresponding string representations, (b) The binary tree with the corresponding minimum distances. 66

4.15 (a) Three digits 5 with the corresponding primitive fragments obtained by using the final parameters for selection and the corresponding string representations, (b) The binary tree with the corresponding minimum distances. 67

4.16 (a) Three digits 8 with the corresponding primitive fragments obtained by using the final parameters for selection and the corresponding string representations, (b) The binary tree with the corresponding minimum distances. 68

Acknowledgements

I would like to thank my supervisor Prof. Lev Goldfarb for guiding me during my studies at UNB; without him I would not have accomplished this work. I will always be inspired by his spirit and honesty in his attitude to life and to science.

Sincere thanks are due to Mrs. Nan Doerksen, my host family, for the nights she spent proofreading my thesis.

I want to thank the following financial supporters who enabled me to start and finish my studies at UNB: the government of Indonesia, the University of New Brunswick, my beloved mother, and finally, the Canadian Bureau for International Education. My stay in New Brunswick was made pleasant by the warm welcome from the Doerksens, the Hicks and the Lees. Special thanks are due to Mini who takes care of my children better than I do; without her I would not have enough time to finish my work. Mbak In always keeps me in good spirits, which inspired me to finish my work. I also wish to thank Mbah Gun who taught me much about humanity and mathematics.

Finally, but by no means last, I wish to thank Ahmady, Cil and Dina, for their enthusiastic support, encouragement, and sacrifice.

Chapter 1

Introduction

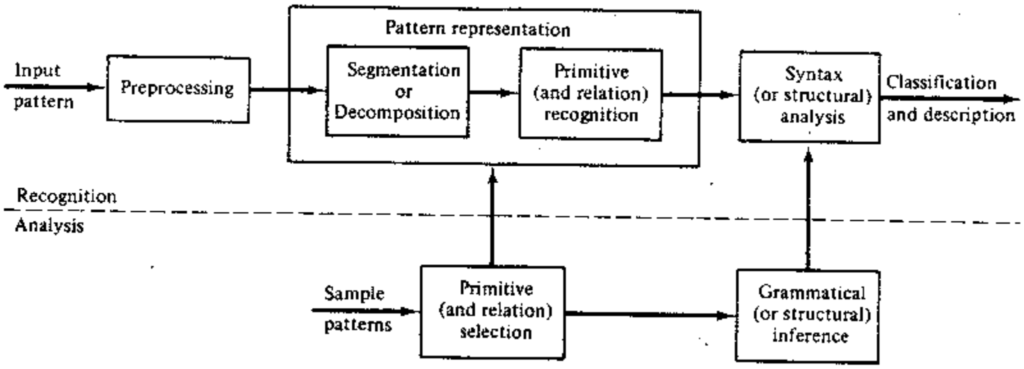
When you have two young, beautiful girls, one a Caucasian and the other Oriental, you may need to look at each girl only once to be sure that you can recognize which is which when you see them again. But, depending on your experience of Oriental people, you may need to look many times at two equally beautiful girls from the Orient in order to identify the various features that enable you to discriminate between them.

The ability to identify the features which distinguish one pattern from another is the main task of pattern recognition , a big sub-area in computer science. Pattern recognition is defined as "the categorization of input data into identifiable classes via the extraction of significant features or attributes of the data from a background of irrelevant details,..., however, in most pattern recognition problem ..., the determination of a complete set of discriminatory features is extremely difficult, if not impossible" [19]. In general, pattern recognition deals with all patterns such as waveforms, speech, voice, odours, images, etc. This thesis deals with pictorial pattern recognition, but the general learning model is applicable to other domains of pattern recognition.

That automatic pictorial pattern recognition cannot compete with the human visual perception is unquestionable. It is commonly believed that in order to successfully build a machine vision system with great ability, one needs to rely, in some sense, on the psychophysiology of human visual perception. By great ability, people may think of high speed, high accuracy and high capability in determining flexible recognition-criteria under different problem types [11]. Several existing methods possess these first two criteria but no system possesses the flexibility at a reasonable level. As an example, in handwritten character recognition, several available systems can yield nearly 90% accuracy of recognition with high speed computation, but the systems are not applicable for any other images. In some of the systems, the features to be extracted are fully determined by the designer [6]. Two main reasons of the failure to develop such a flexible system are the very ad-hoc method of the feature extraction (low-level) and the loose connections between different levels in a system. It is commonly known that the feature extraction methods are provided, somehow, by a-priori knowledge of the designers.

Those methods, clearly will not lead us close to a flexible, general system of pictorial pattern recognition. We need, as in our above illustration, a machine that can determine the set of features such as hair colour, shape of eyes, nose, lips, etc. to discriminate among the above girls without any human interference.

An even more difficult task for the machine is to select from the set of features which are necessary to be used in specific objects. From our illustration, the hair colour may be the only necessary feature to discriminate between a Caucasian girl and an Oriental girl, but it may not be sufficient as the only feature to discriminate between two Oriental girls. It is easy to see from the general block diagram of syntactic pattern recognition in figure 1.1 [6] that the system will, in no such way, have the capability to flexibly change the set of discriminatory features under different objects. Once the system has chosen the primitives (low-level) , it proceeds to the next levels until it reaches the final level. It ignores the possibility of getting incorrect features in the low-level and it does not have any freedom to adapt to the nature of the data. Consequently, the system may have low accuracy or it may be very uneconomical.



Pattern representation
Input pattern
Preprocessing
Segmentation or Decomposition
Primitive (and relation) recognition
Syntax (or structural) analysis
Classification and description
Recognition
Analysis
Sample patterns
Primitive (and relation) selection
Grammatical (or structural) inference

Figure 1.1: The general block diagram of syntactic pattern recognition [6], pp. 9).

A new, general approach to pattern recognition proposed by Goldfarb [9], which allows for an adaptive system of choosing primitives, seems to be capable of resolving the above problems. The system starts with a certain set of parameters for the primitive selection. The resulting primitives are then clustered and, based on the clusters the learning patterns are encoded. By assigning the appropriate weights to the selected primitives through the learning stage, the quality of the chosen primitives can be computed. If this quality is satisfactory, which means that the classes of pattern are well separated, then the learning process is terminated; otherwise the system goes back to the very beginning, i.e. to the primitive selection but with modified parameters. The steps are then repeated. The close connection between every level makes the system neither too self-confident nor too apathetic.

In this thesis, I closely follow the adaptive system of choosing primitives by using the Muchnik method as the primitive selection method [15] and the metric model as the learning model [8] for numeral hand-written recognition.

Basically, the Muchnik method is a transformation of an image representation into another representation through a pre-defined function and a searching procedure of local properties in the new image representation. The method has several parameters which decide the level of the detail of the image representation to be obtained. The more complicated the objects to be discriminated, the more detail the representation should contain. Throughout this method, the machine should be fully independent of human judgement in extracting the features. I further believe that the Muchnik method can also be applicable for texture feature extraction because it allows very raw image representation as the input. The fact that the method is supported by psychophysiology research, i.e. theory of eye movement and fixation points, makes the method even more trustworthy.

The metric model to be used as the learning model is a transformation system (TS), which generates a parametric distance function together with the learning model for the TS. The well-known Levenshtein TS which generates the Levenshtein (edit) string distance function is employed here and the learning model proposed by Goldfarb [8] is used. In our model the objects are represented by strings. By assigning the appropriate weights to the symbols in the alphabet during the learning stage, the system should be able to adapt very flexibly to the nature of the training patterns. The more *important* the role of a symbol in the alphabet in discriminating a class of strings from other classes, the heavier the weight assigned to the symbol. The above importance should be interpreted in terms of giving as small intraclass distance as possible and as big interclass distance as possible. At the higher level the metric model not only provides us with a more economic system but also with higher accuracy through the transformation system; it has the capability to correct, to some extent, errors which may occur at the lower level.

The objectives of this thesis are, first, to initiate the investigation of the new degree of freedom offered by the recently proposed pattern recognition model, second, to suggest that not only should the low-level not be separated from the high-level, as has been practised so far, but, moreover, it should be driven by the high-level, and, third, to continue the investigation of the primitive selection method proposed by Muchnik.

This thesis will be organized in the following way: The psychophysiology research related to the Muchnik method, the Muchnik method of primitive selection, the implementation and the result of experimentation, will be presented in Chapter 2. The metric model will be described in simplified form in Chapter 3. The interested readers are referred to the original reference [8]. Also several examples taken from the experiments are given. Chapter 4 will be devoted to the adaptive system and the numeral hand-written recognition system. And the last chapter will contain the conclusion and the suggestions for future work.

Chapter 2

The Muchnik Method of Primitive Selection

2.1 Introduction

From several major problem areas in pattern recognition summarized by Tou and Gonzalez [19], the extraction of the discriminatory/characteristic features from input data and the reduction of the dimensionality of pattern vectors are recognized to be the central part. The problems usually are complicated since the system has to decide what discriminatory features to extract and how to extract them. From the binary or any non-vector feature representation, specifically from the images, the common way to select the features is known as the image segmentation and texture extraction. In the image segmentation methods proposed so far, the dimensionality reduction is not taken into consideration. Several existing methods can be categorized into three classes : (1) characteristic feature thresholding/clustering, (2) edge detection, and, (3) region extraction. Unfortunately, almost all of them are very ad-hoc in nature, therefore there are no general methods available for all images. As can be observed from [6,5,12], the mathematical method must be supplied by heuristics. The involvement of semantics and a-priori knowledge is considered to be acceptable since the image segmentation is a result of psychophysiological perception [17].

I.B. Muchnik from the Institute of the Control Sciences (Automatica and Telemechanika), Moscow, U. S. S. R. also agreed that primitive selection is a part of psychophysiological perception, but instead of supplementing his method with a-priori knowledge, he took the research and the theory in psychophysiology into account when he designed his method. This method, which has the capability to determine the geometrical features to be extracted without any human intervention, is, for some unknown reason, not sufficiently known. In the only paper published in English [15], the algorithm and the experimental results are not presented in an adequately detailed form.

This thesis, to some extent, continues the investigation of the method. The input data are handwritten digits, which are digitized using software Al-Hazen on an IBM PS/2 work station into a 128 × 128 gray level images, the binarization of these images is done by using the minimum-error thresholding method [13] implemented in Pascal; the resulting images have 0 values for the blank pixels and 1 values for the black pixels. All of the remaining computations are done on IBM 3090 mainframe computer in PL/I language.

This chapter will be organized as follows: the human visual perception aspects related to the Muchnik method, i.e. eye movement and fixation points will be presented in section 2.2, the Muchnik method and the algorithm will be described in section 2.3, and the results of the investigation of the method will be given in section 2.4.

2.2 Eye movement and fixation points

It is an interesting fact that "the visual system is a discontinuous processing system where the discontinuities are usually marked by the occurrence of jumping eye movement" [7], pp. 63. Study in psychophysiology shows that from an image we look at, we only gather some *important* information about the image. The internal representation or memory of an image is a piecemeal affair : a composite of features [16]. The features that are picked out of an image are dependent upon the eye movement, which is adjusted by the brain (figure 2.1).

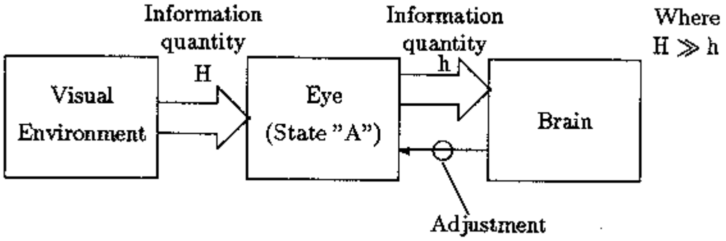


Figure 2.1: Diagram of visual perception through eye movement theory [7].

During viewing, the eye jumps from one stable position to another. The stable positions are called fixation points; the intermediate rapid movements when the eye jumps are called saccade. There are also three important facts related to fixation points. "First, the main share of information processing by the vision system falls on fixation points; during the eye jumps(saccade) the human almost does not perceive the image. Second, under normal conditions, the positions of the fixation points coincide with a place where the viewer's attention is concentrated. Third, fixation points, and therefore the places of attention concentration, are not arbitrarily positioned in the image; there exist image fragments that as a rule attract attention of any viewer" (translation of [21] pp. 9 taken from [3]).

Research in psychophysiology indicates that the eye fixations are in areas with highest spatial frequencies and in the longest durations around the angles of polygons [2], corners [1], unpredictable contours, change directions and unusual details [14].

2.3 The Muchnik method of primitive selection

The method of primitive selection proposed by Muchnik is 1) a process of transforming an image representation into another representation which possesses a better quantitative measure of information, and 2) a process of search for locations in the new image representation which contain high information. For the transformation, Muchnik proposes three methods; the computationally most efficient one , the defocusing method, is chosen to be implemented in this thesis.

2.3.1 An image transformation

A defocusing mask DM, i.e. function $d(a,b)$ defined on an image fragment of size $M \times M$, is simply a fixed discretized potential function of the following form :

$$K(p_0, p) = K_{p_0}(p) = \frac{1}{1 + \alpha \parallel p_0 - p \parallel^2} \tag{2.1}$$

where $p_0 = (p_0^1, p_0^2)$ is the central point and $p = (p^1, p^2)$ is the variable pixel (see fig. 2.2).

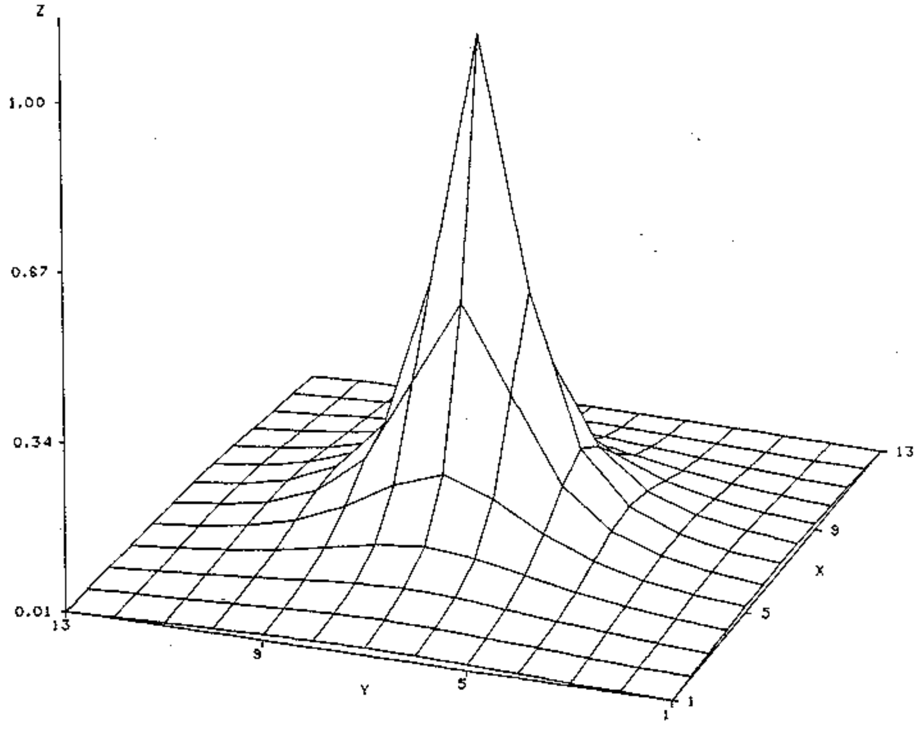
Given an input image I, i.e. function $f(x,y)$ defined on image of size $N \times N$, and a defocusing mask DM of size $M \times M$, one can generate a defocused image DI, i.e. function $g(x,y)$ computed for each pixel (x_0, y_0) as follows :

$$g(x_0, y_0) = \sum_{(x,y) \in IF(x_0,y_0)} f(x,y) \times d(x,y), \qquad \forall \ (x_0, y_0) \tag{2.2}$$

where $IF(x_0, y_0)$ is an image fragment of size $M \times M$ centred at (x_0, y_0), $f(x,y)$ is the image value for the corresponding pixel, and $d(x,y)$ is the corresponding value of the mask DM also centred at pixel (x_0, y_0). The summation is over all pixels in the image fragment of size $M \times M$. In order to compute the values of g for the boundary image pixels, the image is padded with blank pixels. The reader might be familiar with the defocusing transformation in the case when the function defined over some neighbourhoods is a filtering function in spatial domain. The algorithm for computing g is presented in figure 2.3.

0.013	0.016	0.018	0.021	0.024	0.026	0.027	0.026	0.024	0.021	0.018	0.016	0.013
0.016	0.019	0.023	0.028	0.033	0.037	0.038	0.037	0.033	0.028	0.023	0.019	0.016
0.018	0.023	0.030	0.038	0.047	0.055	0.058	0.055	0.047	0.038	0.030	0.023	0.018
0.021	0.028	0.038	0.052	0.071	0.090	0.100	0.090	0.071	0.052	0.038	0.028	0.021
0.024	0.033	0.047	0.071	0.111	0.166	0.200	0.166	0.111	0.071	0.047	0.033	0.024
0.026	0.037	0.055	0.090	0.166	0.333	0.250	0.333	0.166	0.090	0.055	0.037	0.026
0.027	0.038	0.058	0.100	0.200	0.500	1.000	0.500	0.200	0.100	0.058	0.038	0.027
0.026	0.037	0.055	0.090	0.166	0.333	0.250	0.333	0.166	0.090	0.055	0.037	0.026
0.024	0.033	0.047	0.071	0.111	0.166	0.200	0.166	0.111	0.071	0.047	0.033	0.024
0.021	0.028	0.038	0.052	0.071	0.090	0.100	0.090	0.071	0.052	0.038	0.028	0.021
0.018	0.023	0.030	0.038	0.047	0.055	0.058	0.055	0.047	0.038	0.030	0.023	0.018
0.016	0.019	0.023	0.028	0.033	0.037	0.038	0.037	0.033	0.028	0.023	0.019	0.016
0.013	0.016	0.018	0.021	0.024	0.026	0.027	0.026	0.024	0.021	0.018	0.016	0.013

(a)



(b)

Figure 2.2: (a) Defocusing mask DM of size 13×13 for equation 2.1 with $\alpha = 3$. The DM is always computed by discretizing and pixel reindexing at a necessary resolution the fixed size square $p \in (-3,-3) \times (3,3)$, $p_0 = (0,0)$. (b) Plot of (a).

```
procedure DEFOCUS $(I, DM, DI)$.
/*
This procedure applies the defocusing mask $DM$ of size $M \times M$ to an image $I$ and stores
the defocused image in $DI$.
*/

Pad $I$ with blank pixels of width $(M-1)/2$.
For every pixel $pix \in I$ do:
    Begin.
    Extract a fragment $IF$ from $I$ whose centre is at $pix$ and whose size is $M \times M$.
    Compute $g(pix)$ using equation 2.2.
    End.
End of Algorithm.
```

Figure 2.3: Algorithm for applying the defocusing mask.

Once a defocused image DI is obtained from algorithm in figure 2.3, a smoothing procedure is then applied to get another representation SI, i.e. function $h(x, y)$. A smoothing procedure used here is simply a neighbourhood averaging procedure. Each value of the smoothed image SI of size $N \times N$ is obtained by averaging the values of the pixels of DI contained in a pre-defined neighbourhood P of (x_0, y_0) of size $S \times S$:

$$h(x_0, y_0) = \frac{1}{S \times S} \sum_{(x,y) \in P} g(x, y), \qquad 1 \leq x, y \leq N$$

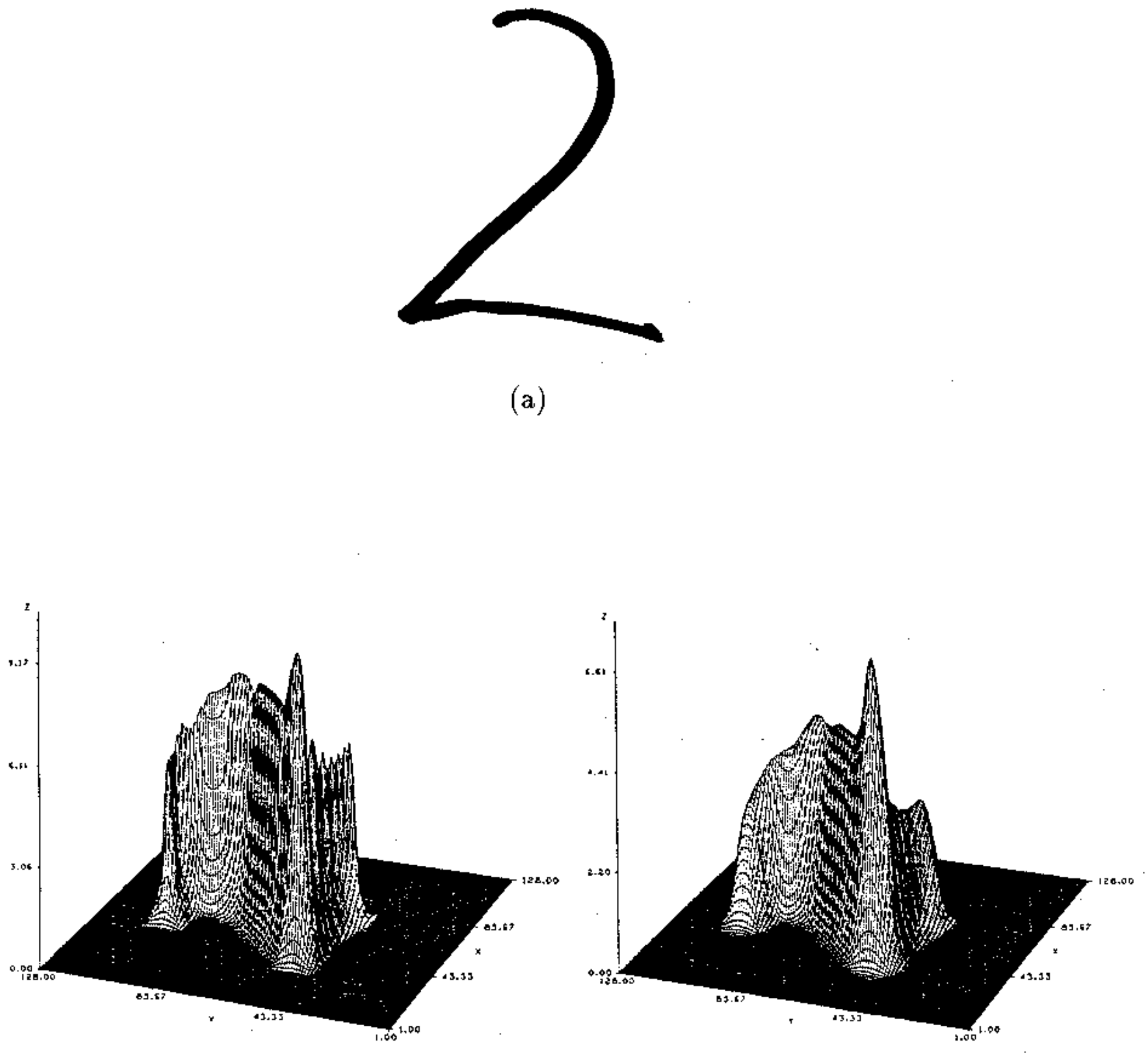
where $S \times S$ is the number of pixels in the neighbourhood P.

The smoothing procedure is applied in order to eliminate numerous small peaks. In figure 2.4 an example of image representation before and after the smoothing is shown. From now on the final image representation SI will be called *the functional image representation.*

2.3.2 The search for image fragments

What remains to be done is the search for image locations that may have image fragments of high information in the functional image representation. First, let us define the local extremum to be searched.

For a search window of size $L \times L$ and an inside window of size $R \times R$, $R \leq L$, pixel $(x_0, y_0) = pix_0 \in SI$ is defined as a local maximum if and only if it fulfils the following requirements :



(a)

(b)

(c)

Figure 2.4: (a) The original image of digit 2, (b) The 3D plot of the defocused image of (a) using the mask of figure 2.2, (c) The 3D plot of the smoothed image of (b) with $S = 9$.

- $h(pix_0) \geq h(pix)$ $\forall$ pixels $pix \in WR$, where WR is the set of pixels in the neighbourhood of pix_0 of size $R \times R$.

- $h(pix_0) > h(pix)$ $\forall$ pixels $pix \in WL$, where WL is the set of pixels in the neighbourhood of pix_0 of size $L \times L$, $WL \supseteq WR$.

The above definition is also true for a local minimum if the sign of SI to be searched is reversed, $h(x, y) = -h(x, y)$.
The search algorithm for the above local extrema given in figure 2.5 is a gradient search method. The search procedure starts from the upper left corner of the image representation and moves along the steepest direction. Observe from figure 2.5 that a 2D array $FLAG$ is used to increase the efficiency of the search algorithm and also to guarantee that all pixels have been processed.

After obtaining the set of locations of extrema from the algorithm in figure 2.5, then the set of image fragments corresponding to it can be obtained. This set is referred to in this thesis as the set of *primitive image fragments* of I, and is easily extracted by positioning the centre of a cutting window of a specified size on the functional image representation in the locations of extrema.

```
procedure SEARCH($SI, L, ILCF$).
/*
This procedure searches for the location of maxima of $SI$ of size $N \times N$ with the search window
of size $L \times L$ and the inside window of size $R \times R$. It calls procedures FINDIP and FLAGING
(figure 2.6). The number of locations of extrema is denoted by $NLCF$. The flags of the
pixels in padded image, a 2D array of size $(N + L - 1)^2$, is denoted by $FLAG$,.
*/

Set $R2 = (R - 1)/2, L2 = (L - 1)/2, NLCF = 0$.
Pad $SI$ with very small valued pixels of width $L2$.
Set $FLAG$ to $true$ in the padded pixels and $false$ otherwise.
Set $FLAGMAX = (N + L - 1)^2$.
Set $NFLAG$ to be the number of padded pixels.
Set $x = 1,\ y = 1,\ CP = SI(x, y)$.
While $(NFLAG < FLAGMAX)$ do:
        Begin. Set $WR$ to be a set of pixels in the $R \times R$ neighbourhood of $(x, y)$.
        Set $TEMPR = CP$.
        Compare $CP$ to every $PR = SI(xr, yr) \in WR$.
        If $PR \geq TEMPR$ then set $MAX = PR,\ xmax = xr,\ ymax = yr,\ TEMPR = PR$.
        If there is at least one $PR \in WR < CP$ then do:
           Begin.
           Call FLAGING($xmax, ymax, r$).
           If $NFLAG \geq FLAGMAX$ then go to end of algorithm.
              Else if $FLAG(xmax, ymax) = true$ then call FINDIP ($xmax, ymax$).
           Go to end of while.
           End.
        Else do:
             Begin.
             For every $PR = SI(xr, yr) \in WR, xr \neq x, yr \neq y$ do:
                  Begin.
                  If $FLAG(xr, yr) \neq true$ then do:
                     Begin.
                     Set $FLAG(xr, yr) = true$
                     Set $NFLAG = NFLAG + 1$
                     End.
                  End.
             Set $WL$ to be a set of pixels in the $L \times L$ neighbourhoods of (x,y)
             excluding the $R \times R$ neighbourhoods.
             Set $TEMPW = CP$.
             Compare $CP$ to every $PW = SI(xw, yw) \in WL$.
             If $PW > TEMPW$ then set $xmax = xw, ymax = ywTEMPW = PW$.
             If there is at least one $PW \in WL \leq CP$ then do:
                Begin.
                Call FLAGING($xmax, ymax, L$).
                If $NFLAG \geq FLAGMAX$ then go to end of algorithm.
                   Else if $FLAG(xmaxymax) = true$ then call FINDIP($xmax, ymax$).
                Go to end of while.
                End.
             Else do:
                  Begin.
                  Set $NLCF = NLCF + 1,\ ILCF(NLCF).xl = x,\ ILCF(NLCF).yl = y$.
                  For every $PW = SI(xw, yw) \in WL$ do:
                       Begin.
                       If $FLAG(xw, yw) \neq true$ then do:
                          Begin. Set $FLAG(xw, yw) = true$.
                          Set $NFLAG = NFLAG + 1$.
                          End.
                       End.
                  Set $x = x + L2 + 1$.
                  Set $CP = SI(x, y)$.
                  If $FLAG(x, y) = true$ then call FINDIP($x, y$).
                  End.
             End.
       End of while.
End of algorithm.
```

Figure 2.5: Algorithm for searching the local extrema.

```
procedure FINDIP($x, y$).
/*
This procedure finds the pixel $(x, y)$ whose $FLAG(x, y) \neq true$.
*/

While $FLAG(x, y) = true$ do:
        Begin.
        If$(x, y)$ is in the lower boundary of $SI$ and not in the right boundary of $SI$ then set $x = 1, y = y + 1$.
          Else if $(x, y)$ is in the lower boundary of $SI$ and in the right boundary of $SI$ then set $x = 1, y = 1$.
              Else $x = x + 1$.
        End.
End of algorithm.
```

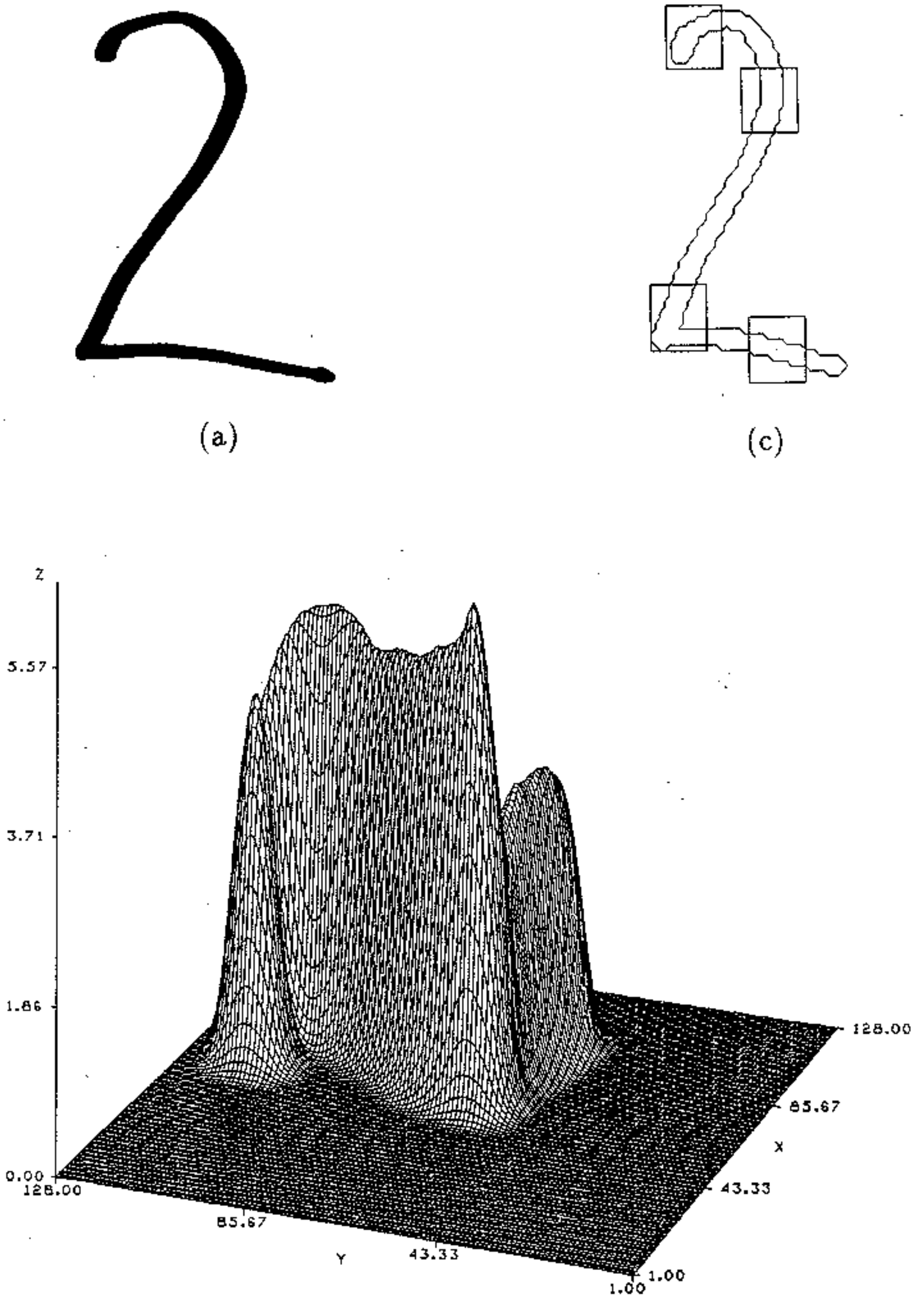
```
procedure FLAGING($i, j, Z$).
/*
This procedure sets the flags of pixel$(i, j)$ as well as the the $Z \times Z$ neighbouring pixels of $(i, j)$ to $true$
subject to the following condition : those pixels no longer have the potential of being local maxima or those
pixels have already been processed.
*/

Set $ZN$ to be a set of pixels in the neighbourhood of $(i, j)$ of size $Z \times Z$.
Set $WZN$ to be a set of pixels $(xwz, ywz) \in ZN$ & $(xwz, ywz) \in WL$.
For all pixels $(xwz, ywz) \in WZN$,
If $FLAG(xwz, ywz) \neq true$ then do:
  Begin.
  Set $FLAG(xwz, ywz)$ to $true$.
  Set $NFLAG = NFLAG + 1$.
  End.
End of algorithm.
```

Figure 2.6: Two procedures for the algorithm in figure 2.5.

2.4 Experimental results

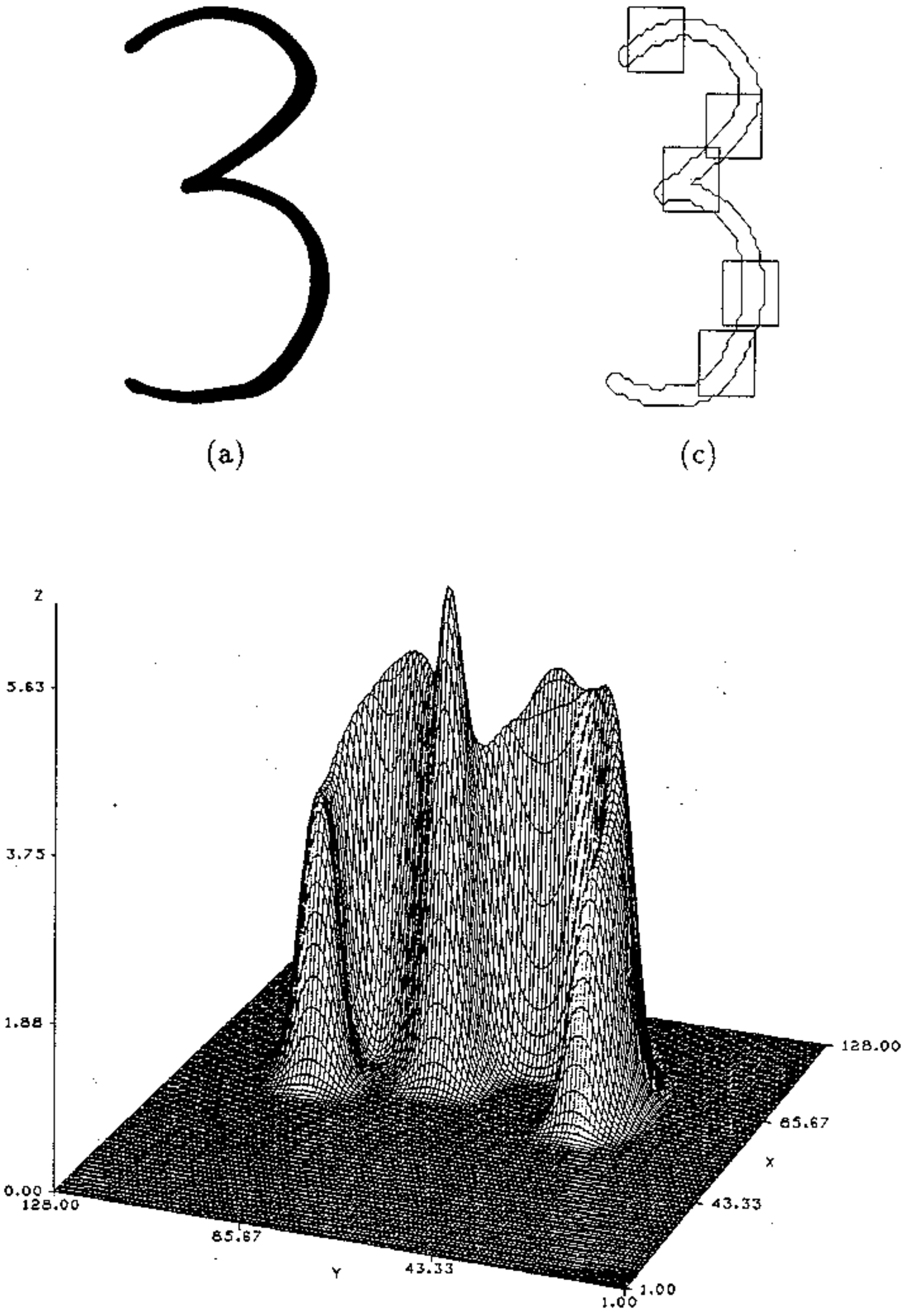
The functional image representations point to the locations of the pronounced changes of geometric forms (corner, intersection, angle, etc.) (figure 2.7, 2.8, 2.9, 2.10). In other words, the locations of important geometric features coincide with the locations of the maxima. The examples of the primitive selections of each digit will be presented. In displaying the image and the corresponding fragments, only the contour of the image is blackened in order to get the clearer picture of the fragments. The reader should not think that any edge detection method is applied; in reality, all the computation is done in the whole image inside the contour. For any figure displaying digit 8, the functional representation of digit 8 is presented in a 90 degrees rotated viewpoint, so that the peaks in the intersection can easily be seen.

The effect of the change of the defocusing mask size on a relatively complex image, such as digit 8, is evident from figure 2.11. The defocusing mask of size 17×17 can not produce a functional image representation that points to two intersections that are presents in digits 8 (figure 2.11(b)). The smaller the mask size, i.e. 9×9 the more detailed information from the image is gathered (figure 2.11(c)). This leads to the assumption that the more complicated the image under study, the smaller the size of the defocusing mask is needed to extract the primitive image fragments, and possibly, the greater the number of primitive image fragments will be obtained.


(a)
(c)

(b)

Figure 2.7: (a) The original handwritten digit 2, (b) the functional image representation for (a) obtained by using the defocusing mask DM of size 13×13, (c) the image and the corresponding fragments for the search window WL of size 9×9.

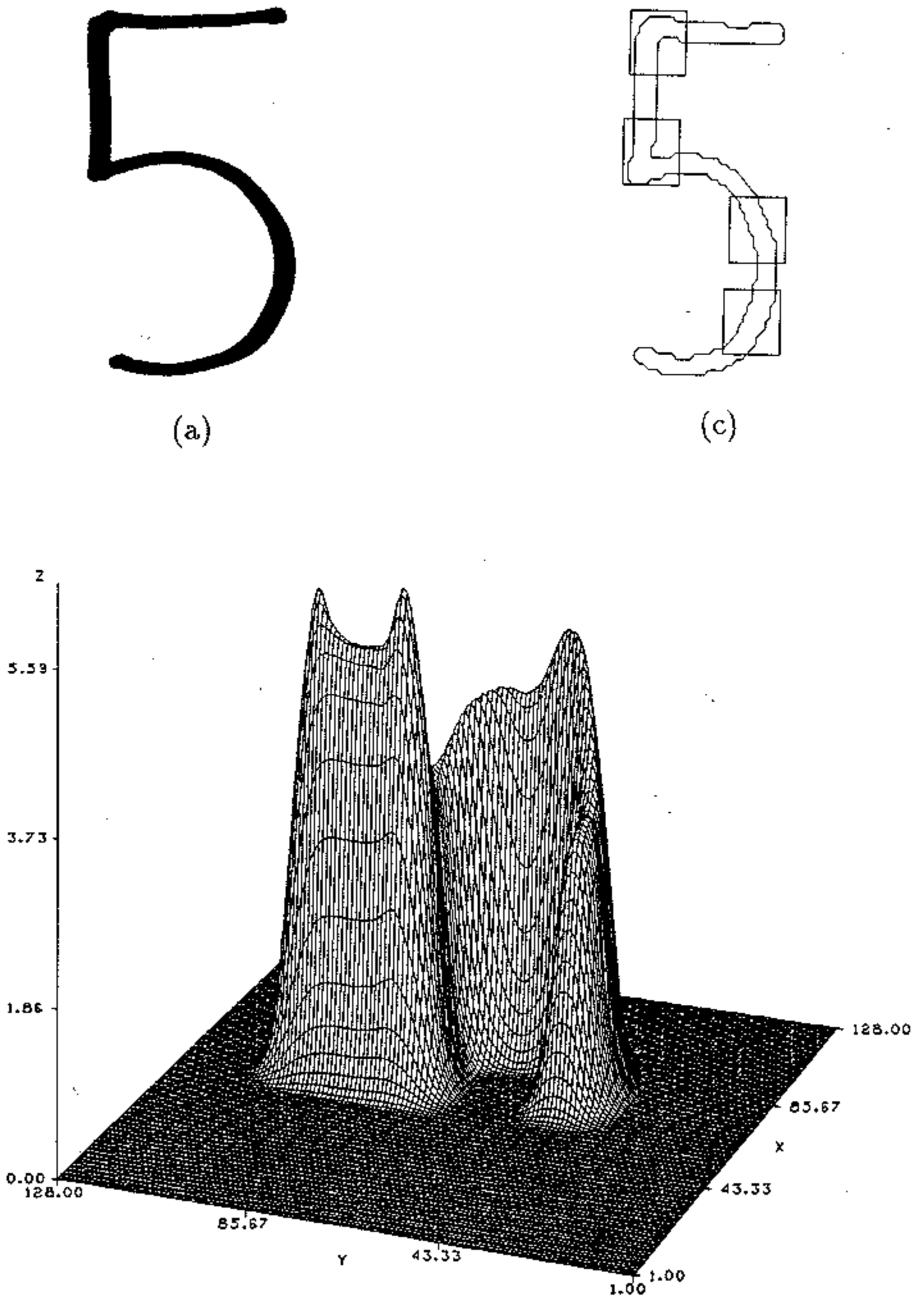


(a)

(c)

(b)

Figure 2.8: (a) The original handwritten digit 3, (b) the functional image representation for (a) obtained by using the defocusing mask DM of size 13×13, (c) the image and the corresponding fragments for the search window WL of size 9×9.



(a)

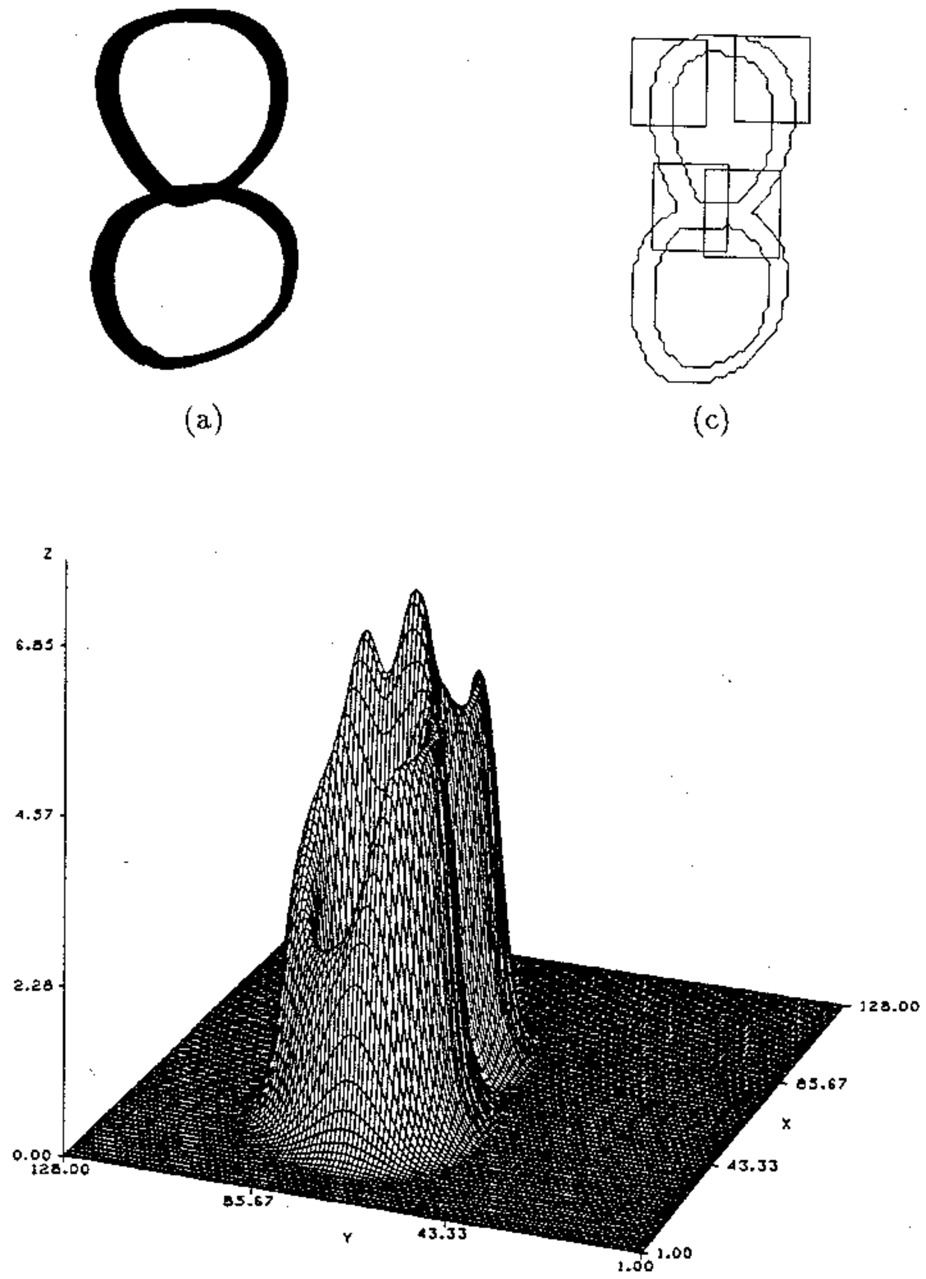
(c)

(b)

Figure 2.9: (a) The original handwritten digit 5, (b) the functional image representation for (a) obtained by using the defocusing mask DM of size 13×13, (c) the image and the corresponding fragments for the search window WL of size 9×9.

In many cases, even if the functional image representation has the necessary information, the search procedure may not be able to extract all the important fragments if the search window is not of an appropriate size (figure 2.12). By using a search window of size 13×13, the location of the right intersection of digit 8 in figure 2.12 can not be detected as a location of maxima, even though the functional image representation clearly shows two big peaks for both intersections. Similar to the situation with the defocusing, the smaller the size of the search window applied, the higher the level of detail of the primitive image fragments extracted.

The only fragments used as primitives in this thesis are those corresponding to the local maxima. However, that does not mean that the features corresponding to the local minima are not important features. In fact, by applying a relatively large defocusing mask, the resulting local minima indicate another pronounced geometric feature which is not detected by the local maxima, i.e. holes for digit 8 (figure 2.13).



(a)

(c)

(b)

Figure 2.10: (a) The original handwritten digit 8, (b) the functional image representation for (a) obtained by using the defocusing mask *DM* of size 13 × 13, (c) the image and the corresponding fragments for the search window *WL* of size 9 × 9.

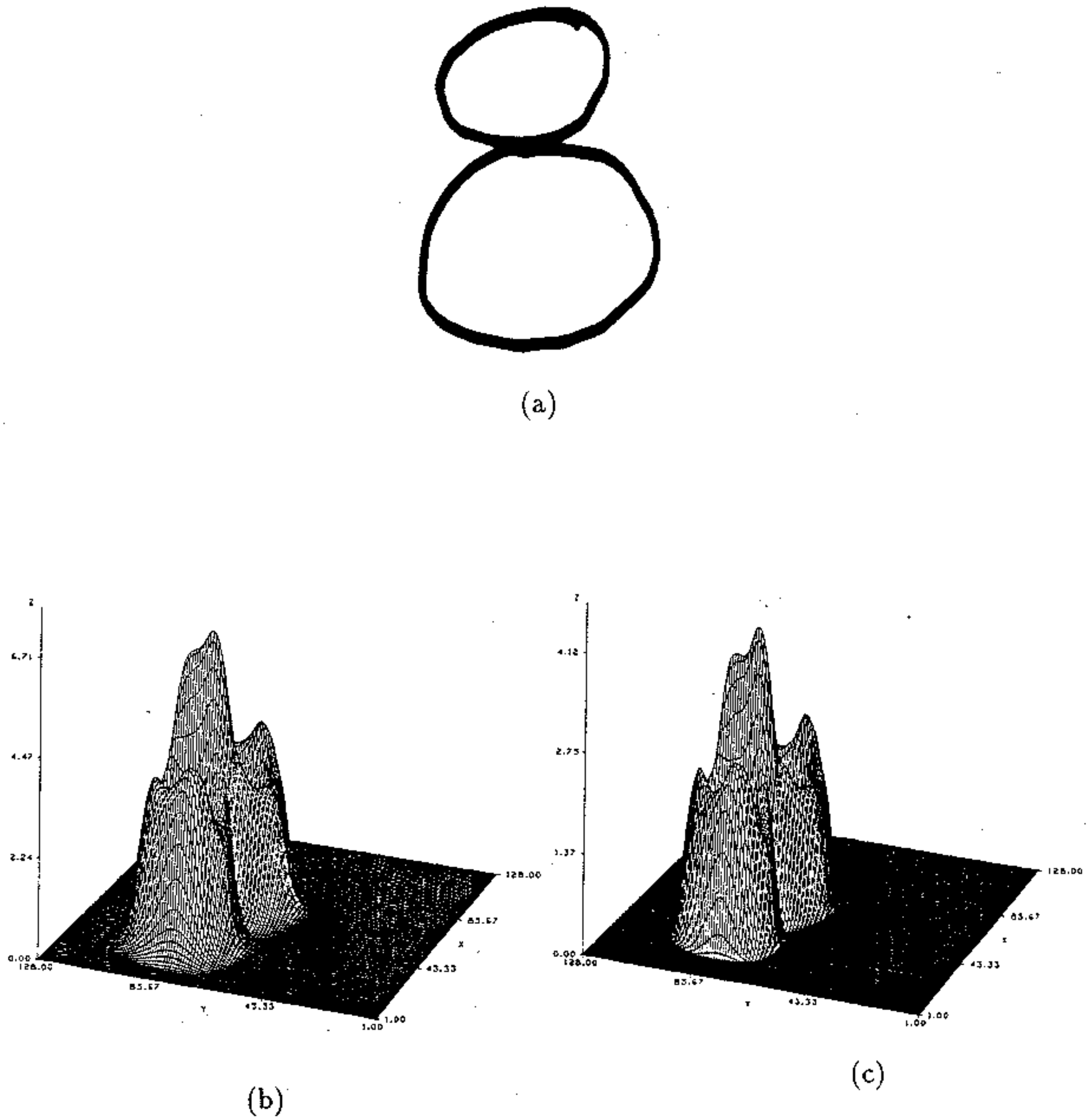
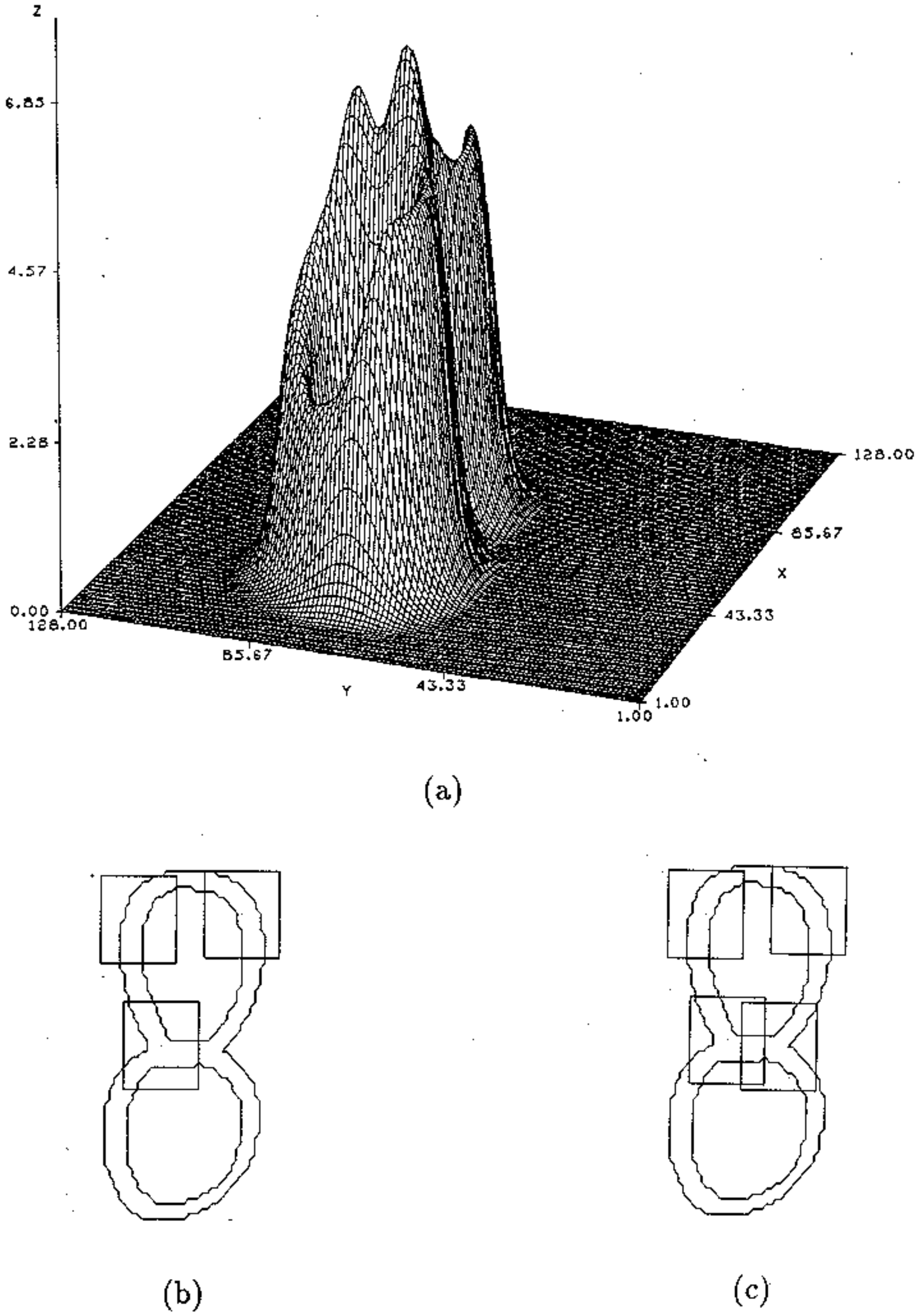


Figure 2.11: (a) The original handwritten digit 8, (b) the functional image representation for (a) obtained by using the defocusing mask DM of size 17×17, (c) the functional image representation for (a) obtained by using the defocusing mask DM of size 9×9.



(a)

(b)

(c)

Figure 2.12: (a) The functional image representation for digit 8 obtained by using defocusing mask DM of size 17×17, (b) the image of digit 8 and the corresponding fragments for the search window WL of size 17×17, (c) the image of digit 8 and the corresponding fragments for the search window WL of size 13×13 .

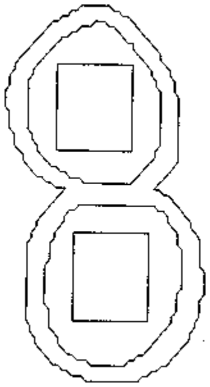


Figure 2.13: Image of digit 8 with the corresponding fragments obtained by using the defocusing mask DM of size 17×17 and the search window WL of size 17×17

Chapter 3

Levenshtein Transformation System in Metric Model

3.1 Introduction

The metric model proposed by Goldfarb [8] is a new, general, evolving model for pattern learning which unifies existing methods. The reader who is interested in the complete concept and exposition of this general mathematical model should read [8], since here I will only present a specific application of the general model. All of the definitions and notations are taken from [8] and [9].

The metric model is a transformation system $\mathbf{T} = (\mathbf{O}, \mathbf{S})$ where $\mathbf{O}$ is a set of homogeneously structured objects and $\mathbf{S}$ is a finite set of permissible rules for transforming one object into another, together with a learning model for the transformation system.

The Levenshtein transformation system ,which is an example of the general transformation system, is employed in this thesis and will be described in section 3.2. The learning model for the transformation system will be presented in section 3.3, and the experimental results will be given in section 3.4.

3.2 Levenshtein Transformation System

The Levenshtein Transformation System $\mathbf{T_L}$ is a set of strings $\mathbf{O}$ over alphabet $\mathcal{A}$ as the object representations together with a finite set of symbol insertions and symbol deletions $\mathbf{S}$ which specify permissible transformation operations for transforming one string into another.

The intrinsic Levenshtein (edit) distance function in $\mathbf{T_L}$ is the function Δ_L defined on $\mathbf{O} \times \mathbf{O}$ as follows :

$$\Delta_L : \mathbf{O} \times \mathbf{O} \rightarrow \mathbf{N},$$

where

$\mathbf{N} = \{0, 1, 2, \ldots\}$, $\Delta_L(o_1, o_2)$ is the smallest number of transformation operations required to derive o_2 from o_1.

$$\begin{array}{rcl} x\,a\,y & \xrightarrow{s_1} & x\,y \\ x\,y & \xrightarrow{s_2} & x\,b\,y \end{array}$$

Figure 3.1: Examples of insertion and deletion operations.

In the above example, s_1 is a deletion operation of symbol a and s_2 is an insertion operation of symbol b.

The next step is to define a weighted Levenshtein distance. The weighted Levenshtein distance in the transformation system $\mathbf{T_L}$ is defined similarly to the intrinsic Levenshtein distance except for the fact that each operation has now a weight associated with it, i.e. it is equal to the shortest weighted path between o_1 and o_2. Next we will describe a dynamic programming algorithm for computing the weighted Levenshtein distance (for detail see [20]).

A cost function γ is defined as a function which assigns a non negative real number to each of the operation in $\mathbf{S}$. Let o_1 and o_2 be finite strings in $\mathbf{O}$, $o_1(i)$ is the ith symbol in string o_1, $o_1\langle i : j\rangle$ is a substring composed of the ith through jth symbols of o_1 and $\mid o_1 \mid$ is the length of string o_1.

Now, let a trace $\mathcal{T}$ be a set of operations in **S** which transforms o_1 to o_2 (ignoring the order of the operations). As an example, let us take two strings o_1 and o_2 ; then a trace $\mathcal{T}$ from o_1 to o_2 can be given as follows :

$$\begin{array}{lcl} o_1 & : & a\,b\,c\,a\,a\,a\,d\,b\,a \\ & & \;/\,/\;\;\,/\;\,/\,/ \\ o_2 & : & b\,c\,d\,a\,d\,b\,c\,d \end{array}$$

The two same symbols from o_1 and o_2 are joined by lines meaning that no operations are required. Any symbol in o_1 which is untouched by a line means that the symbol is deleted, while any untouched symbol in o_2 means that the symbol is inserted. The cost of $\mathcal{T}$ is defined as follows :

$$cost(\mathcal{T}) = \sum_{i \in I} \gamma(o_1\langle i\rangle \to \Lambda) + \sum_{j \in J} \gamma(\Lambda \to o_2\langle j\rangle),$$

where I and J are the sets of position in o_1 and o_2 not touched by any line in $\mathcal{T}$, Λ is the null string.

Now, in order to compute the minimum cost trace $\Delta_L(o_1, o_2)$among all the possible traces from o_1 to o_2, the set of the traces is split to get the inductive solution. A trace $\mathcal{T}$ from o_1 to o_2 can be split into two traces $\mathcal{T}_1$ from o_{11} to o_{21} and $\mathcal{T}_2$ from o_{12} to o_{22}, if and only if there is no line in $\mathcal{T}$ connecting symbol in o_{1i} to symbol in o_{2j} for $i \neq j$, $i, j \in \{1, 2\}$. In any case, trace $\mathcal{T}$ always can be split into two traces $\mathcal{T}_1$ and $\mathcal{T}_2$ such that $\mid o_{12} \mid$ and $\mid o_{22} \mid \leq 1$, and at least one of $\mid o_{12} \mid, \mid o_{22} \mid$ is not zero. Several more notations are introduced as follows :

- $o_1(i) = o_1\langle 1 : i\rangle$.
- $o_2(j) = o_2\langle 1 : j\rangle$.
- $D(i, j) = \delta(o_1(i), o_2(j)), 0 \leq i \leq\mid o_1 \mid, 0 \leq j \leq\mid o_2 \mid$, where $\delta(o_1(i), o_2(j))$ is the minimum cost trace from $o_1(i)$ to $o_2(j)$.

The $D(i,j)$ is computed as follows :

$$D(i,j) = \min\{D(i-1,j) + \gamma(o_1(i) \rightarrow \Lambda), D(i,j-1) + \gamma(\Lambda \rightarrow o_2(j))\},$$

for all $i, j, 1 \leq i \leq \mid o_1 \mid, 1 \leq j \leq \mid o_2 \mid$.

Figure 3.2 gives the detailed algorithm for the weighted Levenshtein string distance.

The function Δ_L satisfies the following three parametric axioms (the proof can be found in [8], pp. 600).

- $\Delta_L(o_1, o_2) = 0$ if and only if $o_1 = o_2$.
- $\forall (o_1, o_2) \in \mathbf{O} \times \mathbf{O} \quad \Delta_L(o_1, o_2) = \Delta_L(o_1, o_2) = \Delta_L(o_2, o_1)$.
- $\forall (o_1, o_2), (o_2, o_3) \in \mathbf{O} \times \mathbf{O} \quad \Delta(o_1, o_2) + \Delta(o_2, o_3) \geq \Delta(o_1, o_3)$.

3.3 Learning model for the transformation system

The fundamentally new concept in the learning model for the transformation system is the concept of *the parametric family of distance function* $\Delta_{L_\omega} = \Delta_\omega$, $\omega \in \Omega$, and Ω is defined as follows :

$$\Omega = \{\omega \in \mathbf{R^m} \mid \omega = (w^1, w^2, \ldots, w^m), w^i \geq 0, \sum_{i=1}^{m} w^i = 1\},$$

where each member of Δ_ω is obtained from the intrinsic distance function Δ_L by assigning to each operation $s_i \in \mathbf{S}$ a non-negative weight w^i.

Before we come to the discussion of the learning model itself, we should define on the weight space $\mathbf{R^m}$ two functions which are the main tools in the model.

Let $\overline{O}$ be a set of disjoint classes $\overline{O}_i (i = 1, 2)$, and let O_i be a finite set of the corresponding training objects; $O_i = \{o_{i1}, o_{i2}, \ldots, o_{ik_i}\}$, and Π be any distance function defined on $\overline{O}$. The *average intraclass* Π*- distance for training class* O_i is defined as

```
**function** LEV_DIST($OB1, OB2, W$).
/*
This function computes the distance between string $OB1$ of length $L1$ and string $OB2$ of
length $L2$. $EA$ is the number of symbols in the alphabet. $W$ is the weighting scheme of
insertion/deletion operations and $RW$ is the weighting scheme of substitution operations.In
this implementation, since substitution operations are not considered, the weights of substitution operations are assigned with very large values .
*/

Set $D(0,0) = 0$.
For $i$ from 1 to $EA$ do:
    Begin.
    For $j$ from 1 to $EA$ do:
        Begin.
        If $i \neq j$ then set $RW(i,j) = bigvalue$.
           Else $RW(i,j) = 0$.
        End.
    End.
For $i$ from 1 to $L1$ do:
    Begin.
    /* Initialize the cost of deletion of string $OB1$ into the $null$ string*/.
    Set $D(i,0) = D(I-1,0) + W(OB1(i))$.
    End.
For $j$ from 1 to $L2$ do:
    Begin.
    /* Initialize the cost of insertion of string $OB2$ from the $null$ string */.
    Set $D(0,j) = D(0,j-1) + W(OB2(j))$.
    End.
For $i$ from 1 to $L1$ do:
    Begin.
    For $j$ from 1 to $L2$ do:
        Begin.
        $m1 = D(i-1,j-1) + RW(OB1(i),OB2(j))$.
        $m2 = D(i-1,j) + W(OB1(i))$.
        $m3 = D(i,j-1) + W(OB2(j))$.
        Set $D(i,j)$ = minimum value over $m1, m2, m3$.
        End.
    End.
Return $(D(OB1(L1), OB2(L2))$.
End of algorithm.
```

Figure 3.2: Algorithm for computing the distance between two strings.

follows :

$$\alpha_{\Pi}(O_i) = \frac{2}{k_i(k_i-1)} \sum_{k=1}^{k_i-1} \sum_{l=k+1}^{k_i} \Pi(o_{ik}, o_{il}),$$

for $k_i > 1$, and $\alpha_{\Pi}(O_i) = 0$ for $k_i = 1$. The *minimum interclass* Π-*distance for training class* O_1 *to* O_2 is defined as follows :

$$\beta_{\Pi}(O_1, O_2) = \min_{\substack{o_i \in O_1 \\ o_j \in O_2}} \Pi(o_i, o_j).$$

The average intraclass Π-distance for O_i is the average distance between the objects in O_i and the minimum interclass Π-distance between O_1 and O_2 is the minimum distance between the objects in O_1 and O_2.

In our specific application, Π is the function Δ_L defined in section 3.2. From now on the average intraclass Δ_L-distance will be called the *intraclass distance* and the minimum interclass Δ_L-distance will be called the *interclass distance.*

Now we are ready to discuss the learning model. The learning model used in this thesis is the hierarchical learning model ([8], pp. 603-605) which generates a binary decision tree at each node of which the following optimization problem is being solved :

$$\max f(\omega),\ \omega \in \Omega,\ f(\omega) = \frac{f_1(\omega)}{c + f_2(\omega)},$$

where

$$f_1(\omega) = \beta_{\Delta_\omega}(O_1, O_2),$$

$$f_2(\omega) = \alpha_{\Delta_\omega}(O_1) + \alpha_{\Delta_\omega}(O_2),$$

and c is a small positive constant to avoid overflow (when the values of f_2 approach zero).

In the above formulas, O_1 or O_2 denote the union of several training classes and not necessarily one training class. Each child node in the binary decision tree corresponds either to the set O_1 or O_2 and for each of them the optimization process is repeated. The algorithm for the weight learning is presented in figure 3.3.

The weight learning process tries to make classes O_1 and O_2 as compact as possible and to move class O_1 as far as possible from class O_2. The solution set for the above optimization problem will be denoted by $\Omega_{\max}$. Each weight in the weighting scheme $\omega^* \in \Omega_{\max}$ indicates the importance of the corresponding insertion/ deletion operation for the recognition of the pattern.

In the next section and next chapter we will see how $f(\Omega_{\max})$ acts as the evaluation measure in the learning process. The classes are considered to be well-separated if at least the interclass distance is equal to the sum of the two intraclass distances, or $f(\Omega_{\max}) \geq 1$ for the terminal nodes. For the nonterminal nodes, due to the fact that O_1 and O_2 represent union of several classes, the value f_2 may not be sufficiently small and therefore $f(\Omega_{\max}) \geq 0.5$ is acceptable. In this situation, the values $f(\Omega_{\max})$ are *satisfactory*.

This learning model is a static version of the general learning model proposed in [8, 10], i.e. the set of operation **S** remains unchanged during the learning process.

The training sets to be dealt with in this thesis are $D2$, $D3$, $D5$, $D8$. which are sets of strings, each string represents a handwritten digit 2, 3, 5 or 8 respectively. These string representations are obtained by first, finding primitive image fragments, clustering them (forming the alphabet for the primitives), and tracing the images in certain directions. In section 4.2 the corresponding details will be presented. In the case of those four classes are the training classes, the binary decision tree will look as it in figure 3.7.

```
procedure LEARN($C, sat, DT$).
/*
This procedure performs the learning process in the metric model for the training set as in the example in section 3.3. In the first level of the binary decision tree, the algorithm attempts to separate the four classes, i.e. $D2, D3, D5, D8$ into two union classes $O1 = D2 \cup D3$ and $O2 = D5 \cup D8$. In the second level, each single class is to be separated. This procedure calls the procedure OPTI (figure 3.4).
*/

Read $D2, D3, D5, D8$ from $C$.
Set $OR = D2 \cup D3 \cup D5 \cup D5$.
Set $O1 = D2 \cup D3$ and $O2 = D5 \cup D8$.
Set $DT$ to be a pointer pointing to a node containing $M, L, E$ and the label of $OR$.
Set $DT \rightarrow L$ and $DT \rightarrow R$ to be pointers pointing to the left and right child of the node pointed by $DT$, i.e. the nodes containing the labels of $O1$ and $O2$ respectively.
Set $sat = true$.
Call OPTI($DT, sat$).
Call OPTI($DT \rightarrow L, sat$).
Call OPTI($DT \rightarrow R, sat$).
Set $WMAX = dummy$ for all leaf nodes.
End of algorithm.
```

Figure 3.3: Algorithm for learning in the metric model.

```
procedure OPTI($P$).

/*
This procedure performs the optimization of $f(\omega)$. Two functions INTERDIST (figure 3.5)
and INTRADIST (figure 3.6) are called.
*/

Set $CONST$ to be a very small value.
Set $FMAX = 0$.
For the weighting scheme $W(i)$ in the unit simplex do:
    Begin.
    $F1 = \text{INTERDIST}(P \rightarrow L, P \rightarrow R, W)$.
    $F2 = \text{INTRADIST}(P \rightarrow L, W)$.
    $F3 = \text{INTRADIST}(P \rightarrow R, W)$.
    $F = F1/(CONST + F2 + F3)$.
    If $F > FMAX$ then do:
        Begin.
        Set $FMAX = F$.
        Set $WMAX = W$.
        End.
    End.
If $P$ points to the root of the tree then do:
  Begin.
  If $FMAX \geq 0.5$ then $sat = true \,\&\, sat$.
     Else $sat = false \,\&\, sat$.
  End.
  Else do:
        Begin.
        If $FMAX \geq 1$ then $sat = true \,\&\, sat$.
           Else $sat = false \,\&\, sat$.
        End.
Store $WMAX$ and $FMAX$ in the node pointed to by $P$.
End of algorithm.
```

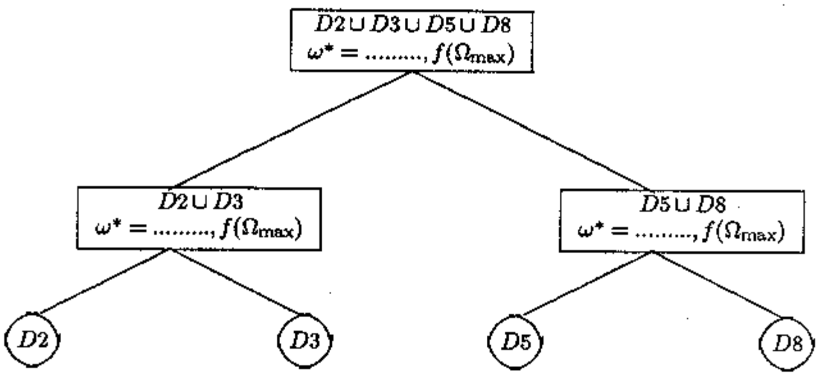
Figure 3.4: Algorithm for optimizing $f(\omega)$.

```
function INTRADIST($C1, W$).
/*
This function computes the intraclass distance of $C1$ using $W$ as the weighting scheme; $NS$
is the number of strings in $C1$.
*/

Set $sum = 0$.
For every pair of strings $(STR1, STR2)$; $STR1, STR2 \in C1$ do:
    Begin.
    Set $tempdist$ = LEV_DIST($STR1, STR2, W$).
    Set $sum = sum + tempdist$.
    End.
Set $IF2 = (2 \times sum)/(NS(C1) \times (NS(C1) - 1))$.
Return $(IF2)$.
End of algorithm.
```

Figure 3.5: Algorithm for computing the intrinsic intraclass distance.

```
function INTERDIST($C1, C2, W$).
/*
This function computes the interclass distance from $C1$ to $C2$ with the
weighting scheme $W$.
*/

Set $min = bigvalue$.
For every pair of strings $(STR1, STR2)$; $STR1 \in C1$, $STR2 \in C2$ do:
    Begin.
    Set $tempdist$ = LEV_DIST($STR1, STR2, W$).
    If $tempdist < min$ then $min = tempdist$.
    End.
Return $(min)$.
End of algorithm.
```

Figure 3.6: Algorithm for computing the intrinsic interclass distance.



$D2 \cup D3 \cup D5 \cup D8$
$\omega^* = ..., f(\Omega_{\max})$
$D2 \cup D3$
$\omega^* = ..., f(\Omega_{\max})$
$D5 \cup D8$
$\omega^* = ..., f(\Omega_{\max})$
$D2$
$D3$
$D5$
$D8$

Figure 3.7: A binary decision tree generated by the weight learning process.

3.4 Weight learning for handwritten digit training patterns

Several examples taken from the experimental results are presented in order to show how the weight learning works. All of the examples have training sets consisting of only two classes of strings representing handwritten digits. In each example, the images, the image fragments, and the clusters of the fragments are provided. The description of the process of obtaining the string representations will be presented in chapter 4. In this section the word *primitive* is used for describing the symbol/label of cluster.

3.4.1 Example one

In this example, the training sets are $D3$ and $D8$. Figure 3.8 shows the images and the corresponding fragments of the two training sets of digits. Image fragments are obtained by using a defocusing mask, a searching window and a cutting window of size 13×13, table 3.1 contains the clusters of all the fragments in the training set (see section 3.3). In the notation of the previous section we have $O1 = D3$, $O2 = D8$.

CLUSTER	FRAGMENTS
a	1,6,8,12,17,20,21,24,26,30,33
b	2,3,4,5,7,9,10,11,13,14,15,16,18,19,22,23, 25,27,28,29, 31,32,34

Table 3.1: Clusters of fragments for the training sets of figure 3.8.

$D3 = \{bbabb, ab, abb, babb\}$

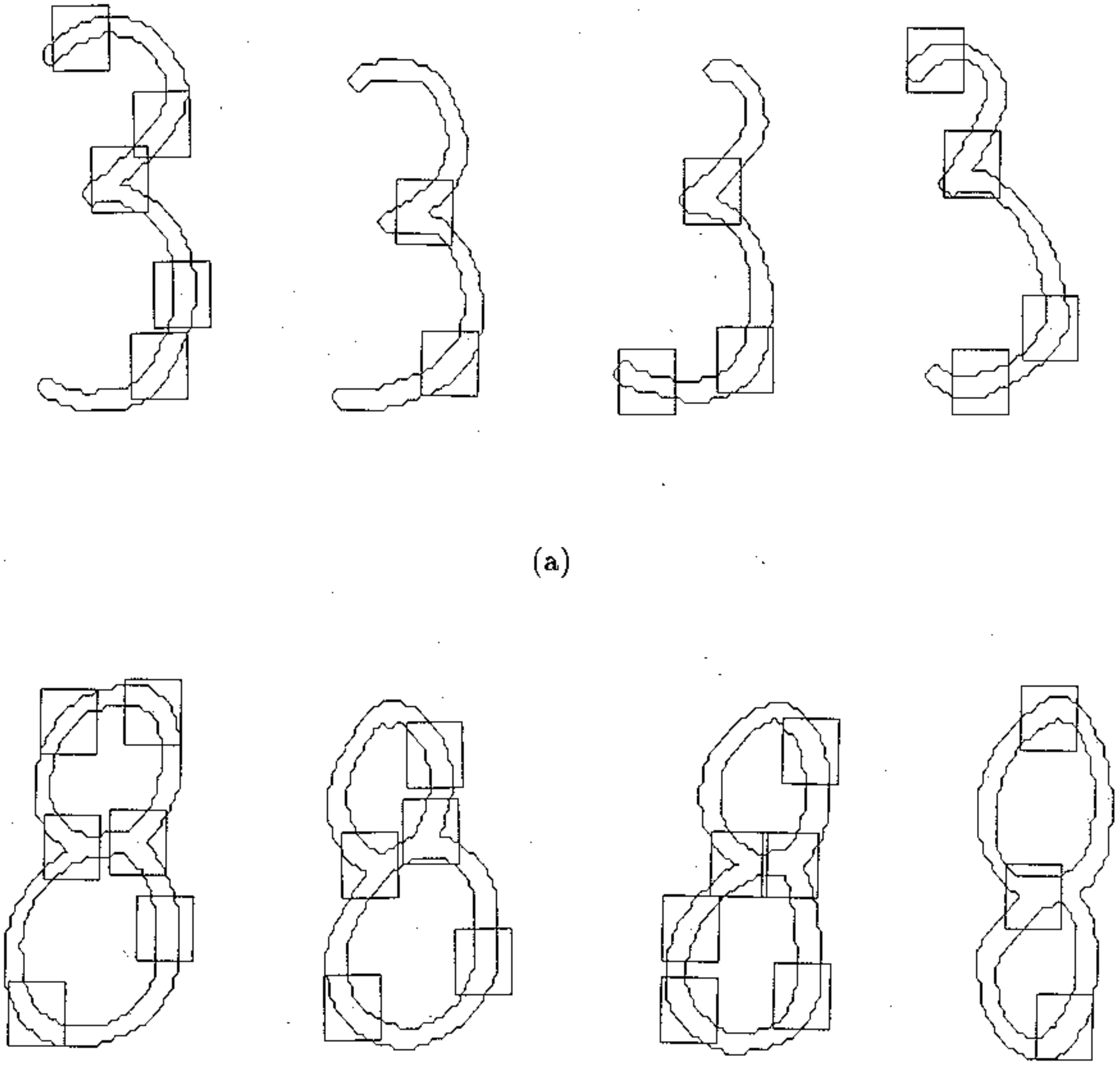
$D8 = \{abbabb, ababb, ababbb, abb\}$

The binary tree in figure 3.9 is produced by using the weight learning scheme described in the last section.

The $f(\Omega_{max})$ is equal to zero since the numerator is zero, i.e. the corresponding interclass distance is zero. This distance is zero because of the existence of the two identical strings each belonging to a different class (the third string in $D3$ and the fourth string in $D8$). Obviously, this situation occurred because the second intersection of the corresponding digit 8 was not detected by the machine (figure 3.8). To summarize, because there is a serious error that occurred in the process of primitive selection, the weight learning can not correct it.

3.4.2 Example two

The training sets for this example also consist of $D3$ and $D8$. The images and the corresponding fragments of the two training sets of digits are presented in figure 3.10.



(a)

(b)

Figure 3.8: (a) The images and the corresponding image fragments of the first training set, (b) the images and the corresponding image fragments of the second training set. Image fragments of (a) and (b) are obtained by using defocusing mask, searching window and cutting window of size 13×13.

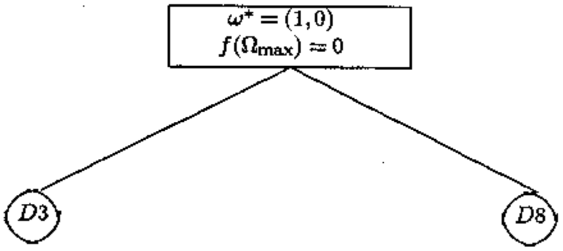


Figure 3.9: Binary decision tree for the training sets of figure 3.8 and table 3.1.

The image fragments are obtained by using defocusing mask of size 13×13, searching window of size 5×5 and cutting window of size 17×17. The clusters of fragments for these training sets are presented in table 3.2. In the notation of the previous section we have $O1 = D3$, $O2 = D8$.

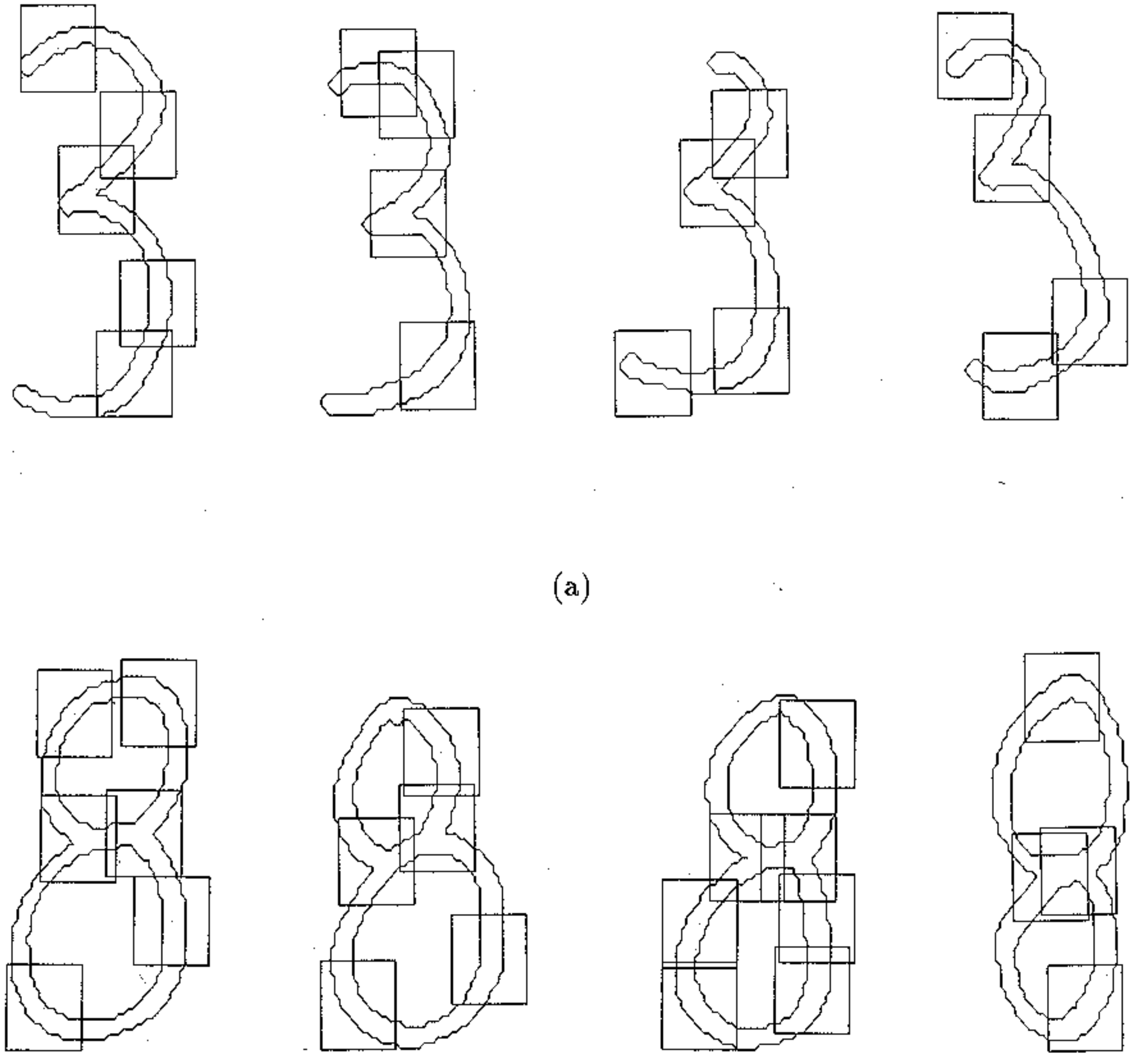
$D3 = \{bcabc,\ bcac,\ cacb,\ bacb\}$

$D8 = \{accabc,\ acacc,\ acacccc,\ abac\}$

CLUSTER	FRAGMENTS
a	1,8,10,15,20,23,24,27,29,33,37,39
b	2,5,6,12,14,17,21,36
c	3,4,7,9,11,13,16,18,19,22,25,26,28,30, 31,32,34,35,38

Table 3.2: Clusters of fragments for the training sets of figure 3.10.

Observe from figure 3.11 that the $f(\Omega_{max})$ has a very big value due to the fact that both intraclass distances are zero. The only nonzero weight value in the weighting scheme is associated with the insertion and deletion operation of primitive a which represents the intersection (or corner) (see table 3.2. In other words, the rest of the primitives (b and c) are disregarded in recognizing the digits 3 and 8. Digit 3 is



(a)

(b)

Figure 3.10: (a) The images and the corresponding fragments of the first training set,(b) the images and the corresponding fragments of the second training set. Image fragments of (a) and (b) are obtained by using defocusing mask of size 13 × 13, searching window of size 5 × 5 and cutting window of size 17 × 17.

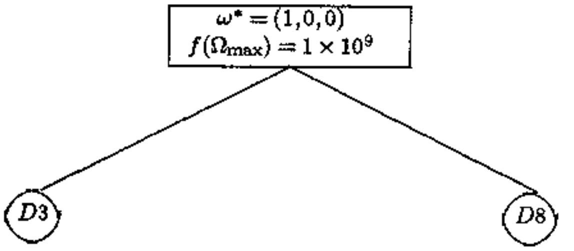


Figure 3.11: Binary decision tree for the training sets of figure 3.10 and table 3.2.

recognized by the presence of a single primitive a, which yields the intraclass distance for class $D3$ to the zero, and digit 8 is recognized by two primitives a, which also yields the intraclass distance for class $D8$ to the zero. This shows us how nicely the weight learning chooses the discriminatory fragments by assigning the big weights to insertion/deletion operation of *important* primitives. In addition, if we compare the examples one and two, which are based on the same training sets, we can see that the sizes of the defocusing mask, searching window and cutting window are important parameters in the primitive selection which will be dealt with in the next chapter.

3.4.3 Example three

In this example, the training sets are $D2$ and $D3$. Figure 3.12 shows the images and the corresponding fragments of the two training sets of digits. Image fragments are obtained by using defocusing mask, searching window and cutting window of size 13×13, table 3.3 contains the clusters of all the fragments for that training sets.

The $O_1 = D2$ and $O_2 = D3$ is the following.

$D2 = (a\,c,\ b\,a\,c,\ b\,b\,a\,c,\ b\,b\,a\,c\,c)$

$D3 = (b\,b\,a\,b\,b,\ a\,b,\ a\,b\,b,\ b\,a\,b\,b)$

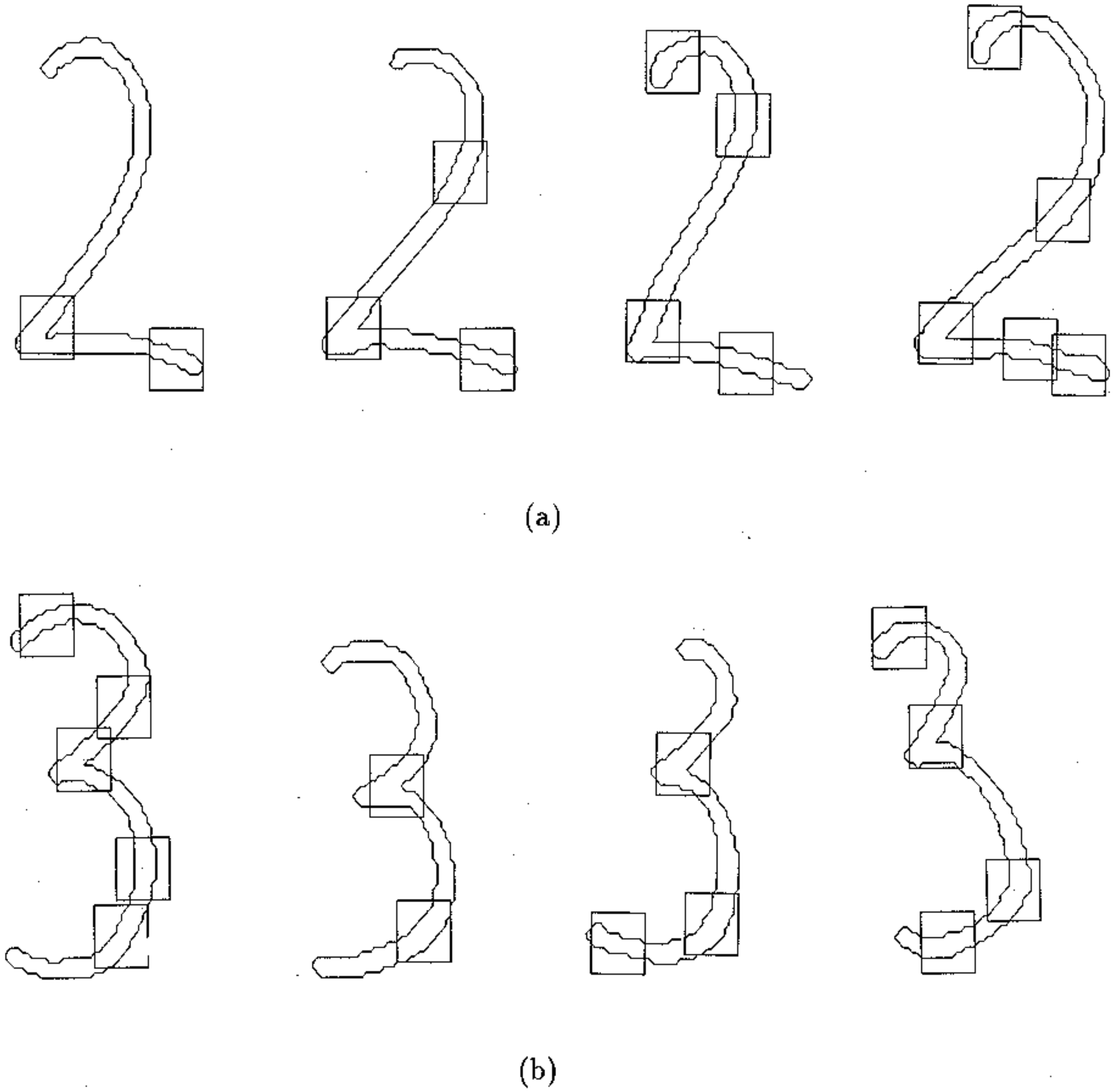


Figure 3.12: (a) The images and the corresponding image fragments of the first training set, (b) the images and the corresponding image fragments of the second training set. Image fragments of (a) and (b) are obtained by using defocusing mask, searching window and cutting window of size 13×13.

CLUSTER	FRAGMENTS
a	1,4,6,12,15,20,22,26
b	3,8,9,10,11,16,17,18,19,21,23,24,25, 27,28
c	2,5,7,13,14

Table 3.3: Clusters of fragments for the training sets in figure 3.12.

The binary tree in figure 3.13 is the result achieved by using the described weight learning. The only nonzero value in the weighting scheme is assigned to the insertion/deletion operation for primitive c, which is present only in the strings of class $D2$. With that weighting scheme, of course, the intraclass distance for class $D3$ is zero. In class $D2$, the fourth string, contrary to the rest of the strings in the class, has two primitives c. This results in a nonzero value 0.5 for the intraclass distance for $D2$ and in value 2 for the $f(\Omega_{max})$. Here, we can see again that the weight learning process can decide which primitive(s) are *important* in recognizing different classes of strings. In this example the direction of a primitive fragment is important, i.e. primitives b and c differ only in the direction.

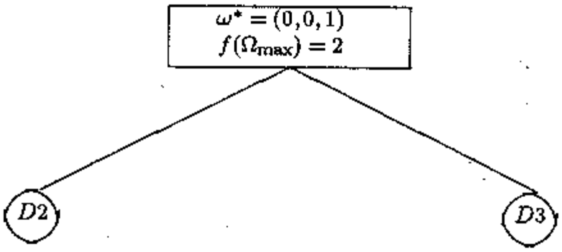


Figure 3.13: Binary decision tree for the training sets of figure 3.12 and table 3.3.

3.4.4 Example four

To show, once more, how the weight learning chooses the discriminatory primitive(s), let us take $D2$ and $D5$ as the training sets. Figure 3.14 shows the images and the corresponding fragments of the two training sets of digits. Image fragments are obtained by using defocusing mask, searching window and cutting window of size 13×13, table 3.4 contains the clusters of all the fragments in the training sets.

CLUSTER	FRAGMENTS
a	1,4,6,12,15,17,19,20,23,26,27,28
b	3,8,9,10,11,16,18,21,22,24,25,29
c	2,5,7,13,14

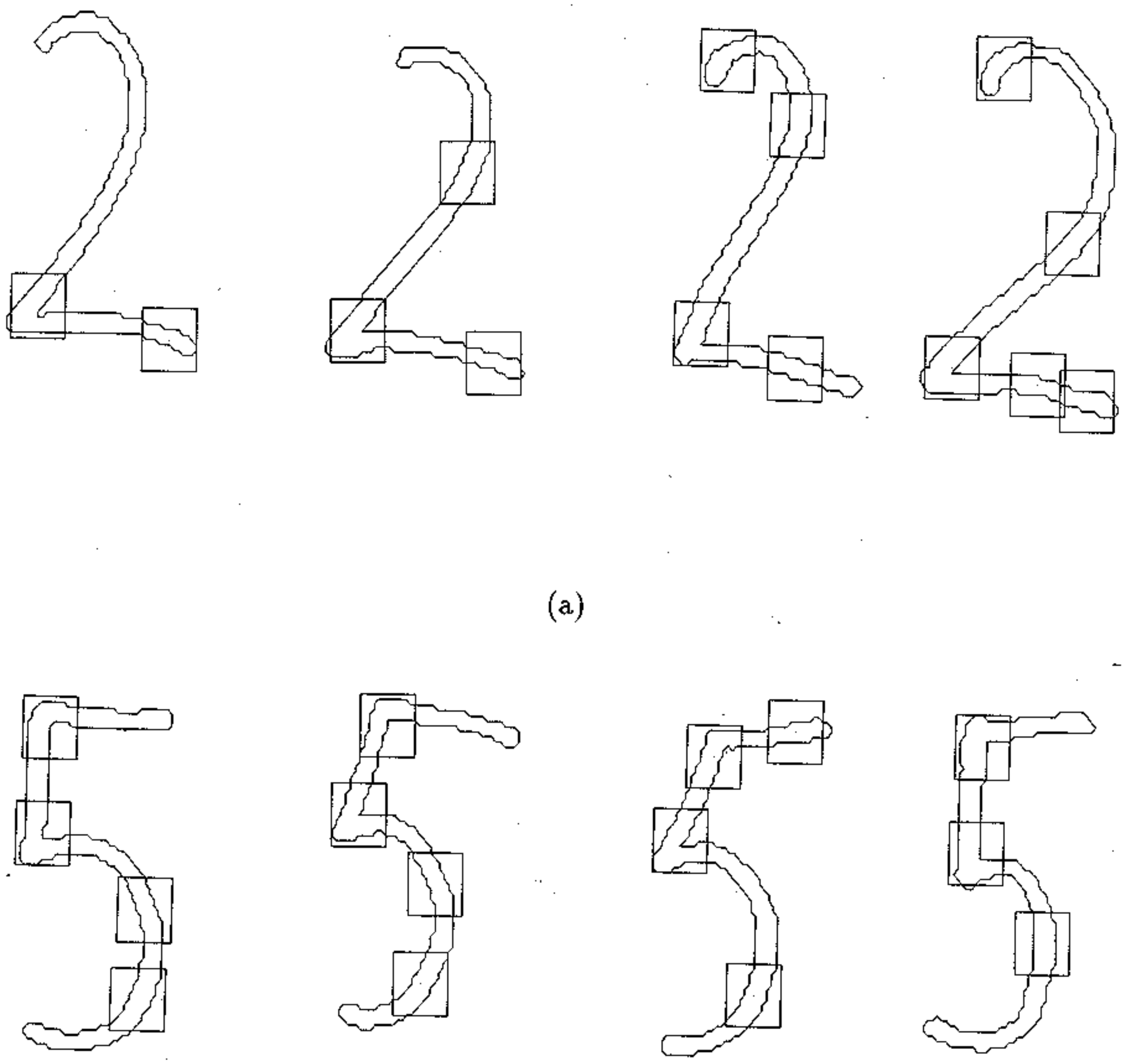
Table 3.4: Clusters of fragments for the training sets of figure 3.14.

The training sets $O1 = D2$ and $O_2 = D5$ are the following.

$D2 = (a\,c,\ b\,a\,c,\ b\,b\,a\,c,\ b\,b\,a\,c\,c)$

$D5 = (a\,a\,b\,b,\ a\,a\,b\,b,\ b\,a\,a\,b,\ a\,a\,b)$

The binary decision tree in figure 3.15 shows us that the discriminatory primitive is a. Observe that even though the primitive c only constructs the string that is present in class $D2$, the weight learning does not assign any weight to the insertion/ deletion operation corresponding to primitive c. If it would, the intraclass distance would not be zero, because the fourth string in class $D2$ would have two primitives c instead of one as in the rest of the strings in the same class. The fragments corresponding to primitive a are the corners. All digits 2 have only one corner, while all digits 5 have two corners.



(a)

(b)

Figure 3.14: (a) The images and the corresponding image fragments of the first training set, (b) the images and the corresponding image fragments of the second training set. Image fragments of (a) and (b) are obtained by using defocusing mask, searching window and cutting window of size 13 × 13.

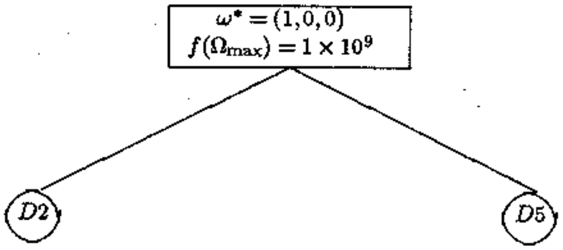


Figure 3.15: Binary decision tree of training sets of figure 3.14 and table 3.4.

Chapter 4

Dynamic selection of primitives in the metric model

4.1 Introduction

When designing an automatic system of visual pattern recognition it is important to also take into consideration the psychophysiological research of the human visual system, since both the feature extraction process and, indeed, the whole system of visual pattern recognition are based on perception. A system of signs [11], which developed through the experimental investigations of visual perception in adults, children and mentally-ill or brain damage persons, has some successive distinct stages which are most likely present in the human visual system.

The proposed dynamic primitive selection system which will be described in 4.3 has similar levels to those of [11]. This dynamic system of primitives selection provides a flexible method for choosing pattern primitives for different problem types. Unlike the existing pattern recognition systems (see section 4.2), all levels in the proposed system are closely connected. Moreover, the high level gives feedback to the low level, which enables the system to reselect the primitives. The construction of the binary decision tree in the dynamic system is discussed in section 4.4. The results of the experiments done with the handwritten digits as the images are given in 4.5.

4.2 The existing visual pattern recognition systems

In general, an automatic pattern recognition system has several levels. The first level is the acquisition of the objects under study (sensing problem). The second level is the feature extraction and the dimensionality reduction, and the third one is the determination of optimum decision procedures for classification and identification [19]. Here I only want to consider the last two levels, the second level is referred to as the low level and the third level is referred to as the high level. All existing systems so far proceed through those levels when designing(updating) the pattern recognition system. The low level gives only the input to the high level and there is no other interrelationships between the levels.

In principle, there are three main categories of methodologies for automatic pattern recognition: heuristic, mathematical, and syntactic. The heuristic methods generally are very ad-hoc and developed for very specialized recognition problems. In term of generalization and automation, these methods are not useful. The mathematical methods have two subdivisions : deterministic and statistical. Both types of the approaches are based on the mathematical framework except that in the statistical approach the statistical properties are employed. In the third group of methods, the syntactic methods, a pattern is described by its primitives (subpatterns) and the relationships among them. These methods are capable of recognizing the complex pattern which can not be adequately represented in a vector form.

Let us recall the syntactic pattern recognition diagram (figure 1.1). The diagram views the recognition system as being partitioned into two major parts : analysis and recognition. In the analysis part the system works with the training set, i.e. selects the primitives and builds the grammatical (structural) inference. The primitive selection is "largely influenced by the nature of the data, the specific application in question, and the technology available for implementing the system. There is no general solution for the primitive selection problem at this time" ([6], pp. 79). For each specific application, the designer selects beforehand the primitives to be used (e.g. stroke segments and the corresponding relations among them are chosen to be the primitives in the chinese character recognition system [6]). The second step is the construction of grammars which generate the language for describing the patterns under study. Let us once again quote the comments of K.S. Fu, the most prominent scientist who worked in the syntactic pattern recognition area. "Ideally, it would be nice to have a grammatical inference machine that would infer a grammar from a set of given strings describing the patterns under study. Unfortunately, such a machine has not been available except for some very special cases. In most cases so far, the designer constructs the grammar based on the a-priori knowledge available and his or her experience" ([6], pp. 93). In his hand-written character recognition system, Fu selected the primitives first and then constructed manually the grammar for each character under study . We can see from the above diagram how big the syntactic system's dependency is on the human designer, how big the inflexibility of the system for different problem types is, and, consequently, how poor the degree of automation of the system is.

4.3 Proposed dynamic system for primitives selection

In order to automate more systems we have to give more freedom to the machine to select the primitives, to learn from the data, to evaluate the learning process, and to make it self-adjusting based on the learning process evaluation. In other words, the system checks how well the primitives are capable of representing the patterns in the training set from the discriminatory point of view. The need of such an adaptive system of pattern recognition has been noticed in the early 1970's. "When we wish to design a pattern recognition system which is resistant to distortions, flexible under large pattern deviations, and capable of self-adjustment, we are confronted with the adaptation problem" ([19], pp. 15). Even a functional block diagram of such a system has been suggested (figure 4.1). There is no automatic pattern recognition system yet developed based on the diagram, simply because there are no available tools for some of the levels. Referring to the diagram in figure 4.1 , the separation between the "contextual analysis" and "estimation, adaption, learning" should not be present in the corresponding system.

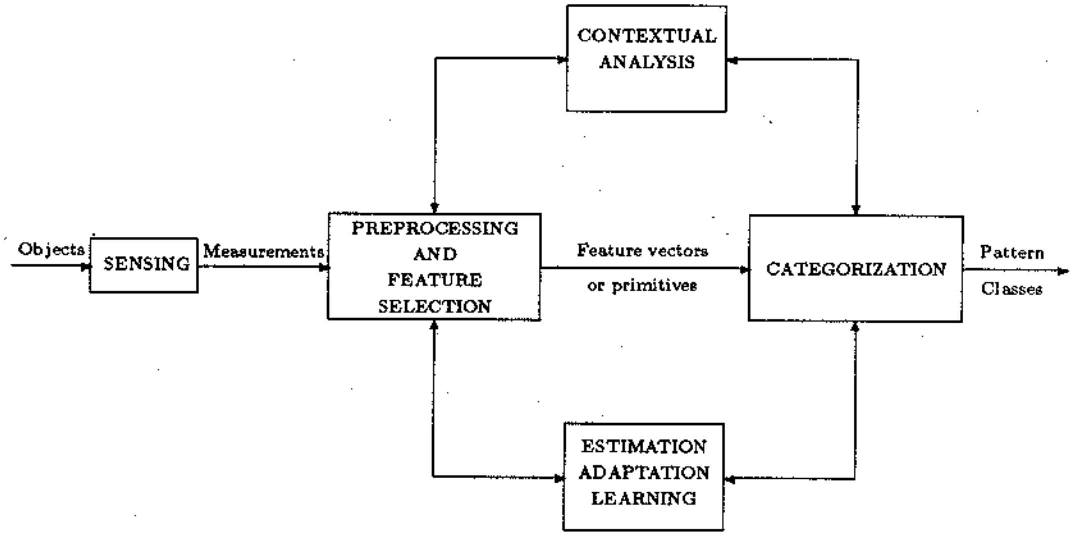

CONTEXTUAL ANALYSIS
Objects
SENSING
Measurements
PREPROCESSING AND FEATURE SELECTION
Feature vectors or primitives
CATEGORIZATION
Pattern Classes
ESTIMATION ADAPTATION LEARNING

Figure 4.1: Functional block diagram of an adaptive pattern recognition system [19] pp. 16

The dynamic selection of primitives proposed in [9] is conceptually different from the above. The difference mainly comes because of the existence of the evaluation function $f(\omega)$ in the learning process (figure 4.2) which allows the system to self-adjust. In this model, the high level drives the low level by means of the evaluation measure: the low level changes the parameters and reselects the primitives if the measure is low. Also, there is no separation between contextual analysis and the adaptation learning; the learning model manages both tasks.

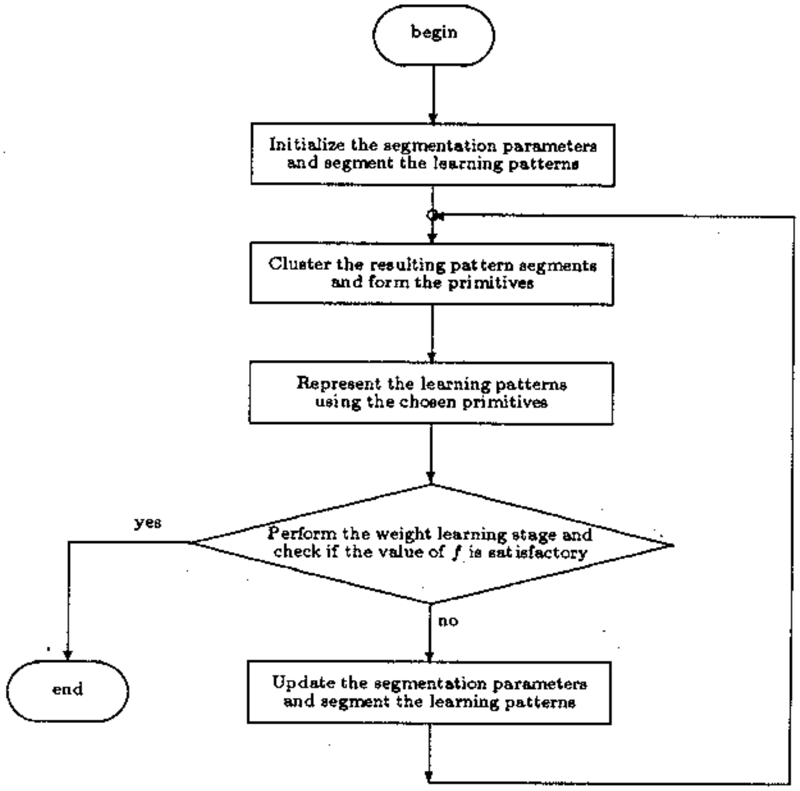

begin
Initialize the segmentation parameters and segment the learning patterns
Cluster the resulting pattern segments and form the primitives
Represent the learning patterns using the chosen primitives
Perform the weight learning stage and check if the value of f is satisfactory
yes
no
end
Update the segmentation parameters and segment the learning patterns

Figure 4.2: Block diagram of the dynamic selection of primitives [9] pp. 22.

This thesis follows exactly the diagram in figure 4.2 and uses the Muchnik method as the primitive selection method (see chapter 2). The learning model used is discussed in chapter 3.

Once the set of primitive fragments from the training images is obtained (by Muchnik method), the city-block distances between each pair of primitive fragments are computed to obtain a matrix of distance(figure 4.6). Using the average linkage cluster analysis provided by SAS software with the distance matrix as the input, the clusters of the primitive fragments are obtained. Next, the primitive fragments are labelled according to their clusters.

Each image is encoded as a string of cluster labels corresponding to each primitive fragment and is generated by tracing the image starting from a certain initial position (figure 4.3). In this thesis, we are not concerned with the algorithm for generating string representation when the primitive fragments have been chosen since this topic has already been addressed in the pattern recognition literature.

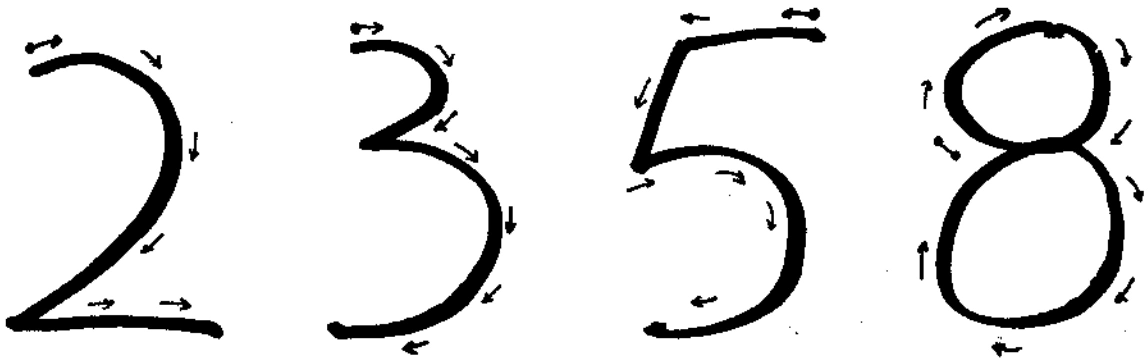


Figure 4.3: Tracking direction and the initial position of encoding of each four digit.

The obtained representations become inputs for the learning process discussed in chapter 3. If the evaluation measures are satisfactory then the resulting binary tree is the binary decision tree for the pattern recognition system, otherwise, the three parameters in the Muchnik method of primitive selection are changed and the primitive selection process is repeated until the evaluation measures are satisfactory (figure 4.4). Section 4.4 will discuss the construction of the binary decision tree. The pattern recognition system is completed once we have the final binary decision tree.

Next, we will discuss about the recognition stage. At the low level, the primitive fragments of the digit to be recognized are selected by the Muchnik method using the fixed defocusing mask size, fixed search window size and fixed cutting window size stored at the root of the final binary decision tree. Then, following the same procedure as for the training sets, the digit is encoded into the string. At each node of the binary decision tree, using the weighting scheme corresponding to this node weighting scheme (stored at this node), the string representing the input digit is classified using the corresponding distance measure by the minimum distance classification rule. The digit is recognized to be in the class corresponding to the leaf node at which the recognition process terminates. The algorithm is presented in figure 4.5.

```
procedure MAIN($NI, UM, LM, UL, LL, UE, LE$);
/*
This main procedure generates the binary decision tree whose root is pointed by a pointer $DT$ for the training sets of $NI$ number of images of $I$ each of size $N \times N$ with the $UM, LM, UL, LL, UE, LE$ are the upper and lower range of $RANGE$ of the size of the defocusing mask, search window and cutting window respectively. It calls procedures DEFOCUS(figure 2.3), SEARCH(figure 2.5), FR_DIST(figure 4.6) and LEARN(figure 3.3). $DM$ is the de-focusing mask, $DI$ is the de-focused image and $SI$ is the smoothed-image. The positions of local maxima are stored in $LCF$ and the extracted fragments are stored in $FRAGMENT$. The distance matrix between the fragments in the training set is stored in $MATRIX$. $C$ denotes the set of strings representing the images; the value of $sat$ indicates whether the values $F$ in the learning process are satisfactory or not. For $i = 0$ or $j = 0$, $W(i, j)$ denotes the weight of the insertion-deletion operation.
*/

For $a$ from 1 to $NI$ do: read $I(a)$.
For $M$ from $UM$ to $LM$ by $RANGE$ do:
    Begin.
    Generate defocusing mask $DM$ of size $M \times M$ (equation 2.1).
    For $a$ from 1 to $NI$ do:
        Begin.
        Call DEFOCUS($I(a), DM, DI(a)$).
        Apply the smoothing procedure on $DI$ and get $SI$ as the result.
        End.
    For$L$ from $UL$ to $LL$ by $RANGE$ do:
        Begin.
        For $a$ from 1 to $NI$ call SEARCH($SI(a), M, LCF(a)$).
        For $E$ from $LE$ to $UE$ do:
            Begin.
            For $a$ from 1 to $NI$ do:
                Begin.
                Extract fragments from $SI(a)$ whose centers are positioned
                in $LCF(a)$ of size $E \times E$, and store them in $FRAGMENT$.
                End.
            Call FR_DIST($FRAGMENT, MATRIX$).
            Cluster the fragments based on the distance matrix $MATRIX$ by
            using the average linkage method in $SAS$ and label each cluster.
            For $a$ from 1 to $NI$ do:
                Begin.
                Trace $I(a)$ according to the tracking direction presented in figure 4.3 and
                encode it as $C(a)$ according to the labels corresponding to its primitive fragments.
                End.
            Call LEARN($C, sat, DT$).
            If $sat = true$ then go to end of algorithm.
            End.
        End.
    End.
End of Algoritm.
```

Figure 4.4: Algorithm for dynamic system of primitive selection.

```
procedure RECOG($DT, OBJECT, RO$);
/*
This procedure recognizes $OBJECT$ using the binary decision tree whose root is pointed to by $DT$ as class $RO$. It calls procedures DEFOCUS(figure 2.3) and SEARCH(figure 2.5). $M, L$ and $E$ are read from the vertex pointed to by $DT$ and are the proper sizes of the defocusing mask, search window and cutting window respectively.
*/

Read $M, L$ and $E$ from the root of the binary decision tree.
Call DEFOCUS($OBJECT, M, DOBJECT$).
Apply the smoothing procedure on $DOBJECT$ and get $SOBJECT$ as the result.
Call SEARCH($SOBJECT, L, OLCF$).
Extract the fragments of size $E \times E$ from $SOBJECT$ in position of $OLCF$ and store it in $OFRAGMENT$. Compute and find the nearest fragment in the training set $FRAGMENT$ from each fragment in $OFRAGMENT$ by using the same method of distance computation, i.e. the distance between two fragments is given by the procedure FR_DIST (figure 4.6).
Label the fragment in $OFRAGMENT$ according to the label of its nearest fragment in $FRAGMENT$.
Encode $OBJECT$ as $OSTRING$.
Set $CE = DT$.
Set $CN$ as the node pointed to by $CE$.
While ($CE \neq null$ and $WMAX$ in $CN \neq dummy$) do:
    Begin.
    Using the weighting scheme $WMAX$, compute the string distance between $OSTRING$ and each string in subtree of $CN$, and find $mins$, i.e. the string whose distance to $OSTRING$ is a minimum.
    Set $CE$ to $mins$.
    Set $CE$ to be a pointer pointing to the child of a node pointed to by $CE$.
    End.
Set $RO = mins$
$OBJECT$ is recognized to be in the same class of $RO$.
End of algorithm.
```

Figure 4.5: Algorithm for object recognition.

```
procedure FR_DIST($FRAGMENT, MATRIX$).
/*
This procedure computes the distance between every pair of fragments in $FRAGMENT$
and store the distance values in $MATRIX$.
*/

For every pair $(FR1, FR2)$; $FR1, FR2 \in FRAGMENT$ do:
    Begin.
    Pad $FR1$ with blank pixels of width 1.
    Set $min = bigvalue$.
    For the center of $FR2$ coincident with that of $FR1$ as well as the center of $FR2$
    shifted in all 8 directions by a radius of 1 pixel from the center of $FR1$ do:
        Begin.
        Compute the distance $dist$ by adding the absolute value of the difference between
        each corresponding pixel in $FR1$ and $FR2$.
        If $dist < min$ then $min = dist$.
        For rotation of $FR2$ by 90, 180, 270 and 360 degrees do:
            Begin.
            Compute the distance $dist$ by adding the absolute value of the difference between
            each corresponding pixels in $FR1$ and $FR2$.
            If $dist < min$ then $min = dist$.
            End.
        End.
    Store $min$ as the distance value of $(FR1, FR2)$ in $MATRIX$.
    End.
End of algorithm.
```

Figure 4.6: Algorithm for computing the matrix distance among fragments.

4.4 On the reasons why the construction of the binary decision tree is possible

It is important to remember that during the construction of the binary decision tree the optimization problem is solved separately for each consecutive node, i.e. the sizes of the three window - defocusing mask, search window, cutting window - are adjusted within the corresponding loop (figure 4.4) for the classes associated with the node.

Once the parameters for all parents of the leafes are obtained, the final parameters are set as follows : we choose the smallest defocusing mask, the smallest search window, and the largest cutting window. With these parameters, the new string representation of all the digits as well as the final optimal weighting schemes for each of the tree nodes are constructed.

Please note that the loop for defocusing mask decreases the size, the loop for the search window also decreases the size, and the loop for the cutting window increases the size. The reasons for these are as follows : with the decrease of the defocusing mask size and the search window size, no geometric features are lost, only new could be generated (recall section 2.4); and with the increase of the cutting window size only relative distances between the similar features decrease, such that the better clusters are obtained.

4.5 The Experimental Results

In this section, the final binary decision tree generated by the dynamic system of primitives selection is presented as well as the results for the test patterns. The final binary decision tree correponds to the following parameters for the primitives selection : defocusing mask of size 9×9, search window of size 5×5, and cutting window of size 17×17. The images together with the corresponding primitive fragments obtained by using the above parameters are presented in figures 4.7 and 4.8 and the final binary decision tree is presented in figure 4.9. Table 4.1 shows the corresponding primitive fragment clusters. The training sets are the following :

$D2 = \{efbd,\ eebd,\ dfebgg,\ debgd\}$

$D3 = \{debfe,\ gebe,\ ebeh,\ dbeh\}$

$D5 = \{hccee,\ cbee,\ hcbfe,\ hccfh\}$

$D8 = \{aeeafe,\ aebfe,\ aeaeeee,\ afae\}$

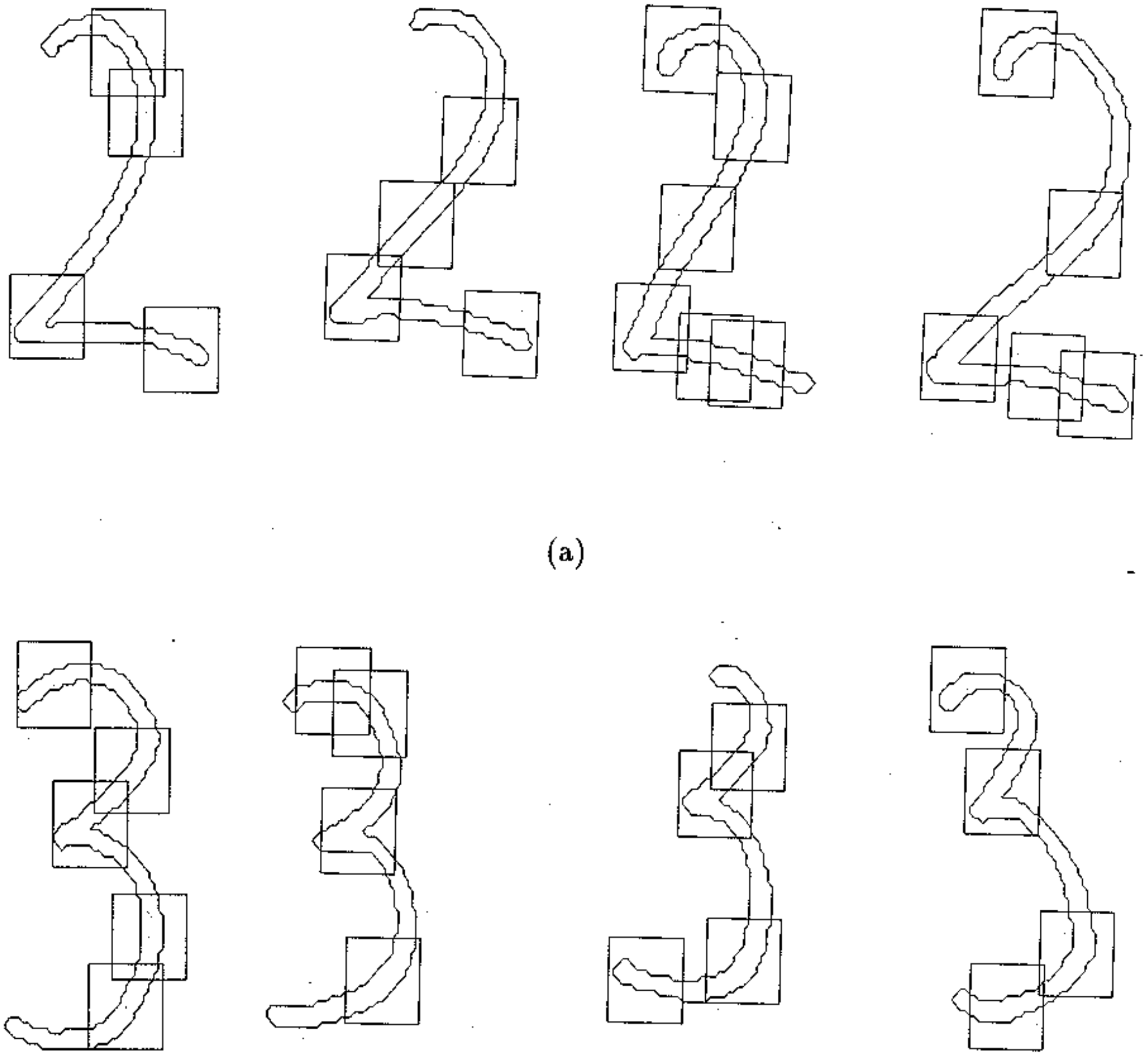
In order to show how the dynamic system works, an intermediate binary decision tree and the corresponding primitive fragments are given (figures 4.10,4.11 and 4.12 and table 4.2). The training sets are listed as follows:

$D2 = \{ad,\ gad,\ bgad,\ bgadd,\}$

$D3 = \{bgagg,\ ag,\ agf,\ bagb,\}$

$D5 = \{cceg,\ cceg,\ bcag,\ ccg,\}$

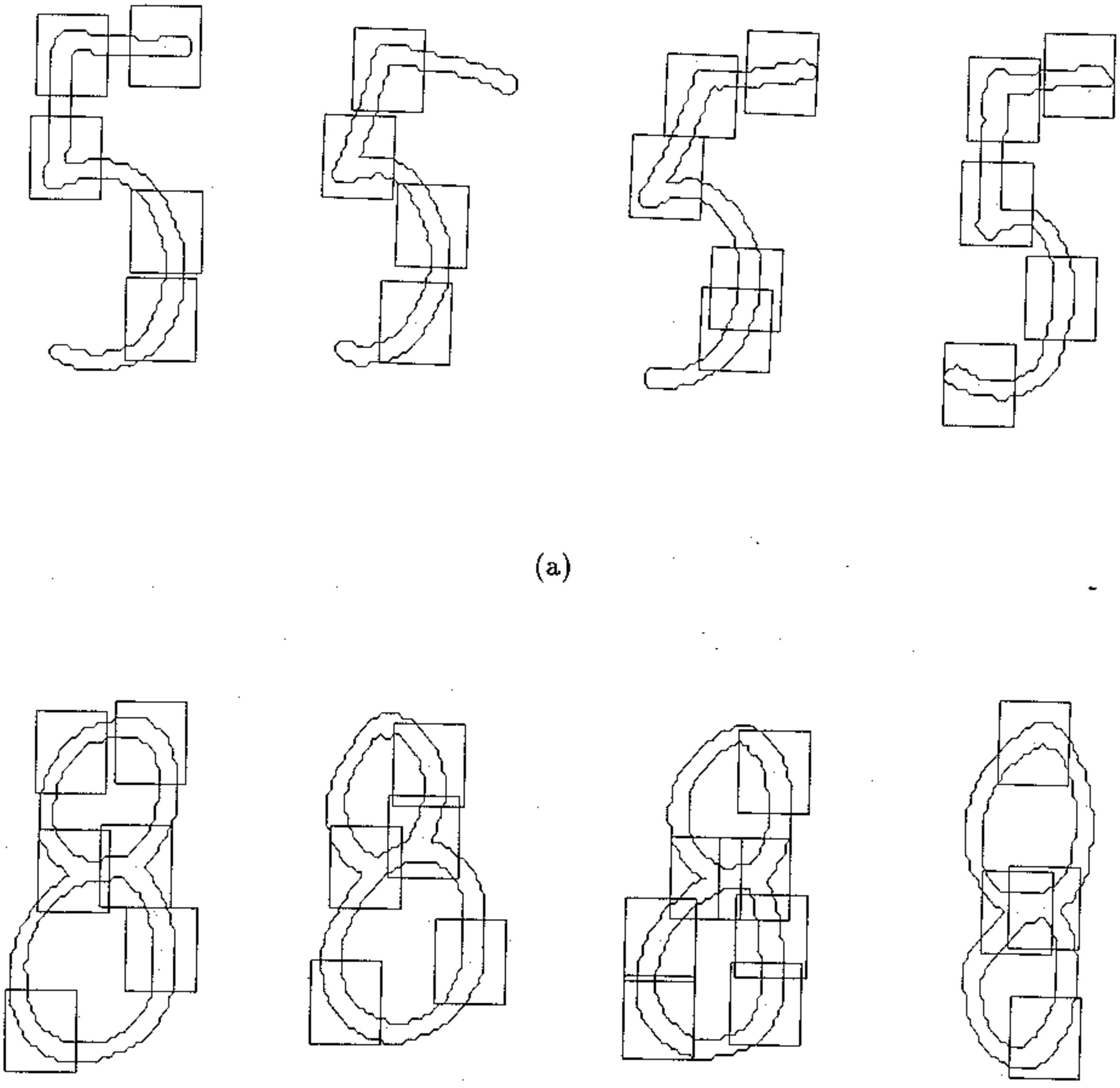
$D8 = \{abeagf,\ aeagf,\ aeagfg,\ abg,\}$



(a)

(b)

Figure 4.7: (a),(b) The images and the corresponding primitive fragments obtained by using defocusing mask of size 9×9, search window of size 5×5 and cutting window of size 17×17 for training sets $D2$ and $D3$ respectively.



(a)

(b)

Figure 4.8: (a),(b) The images and the corresponding primitive fragments obtained by using defocusing mask of size 9×9, search window of size 5×5 and cutting window of size 17×17 for training sets $D5$ and $D8$ respectively.

CLUSTER	FRAGMENTS
a	58,61,62,67,71,75,77
b	1,6,9,17,20,27,29,34,43,46,65
c	37,40,42,49,51,52
d	4,8,14,15,19,24,33
e	2,5,7,13,16,22,23,26,28,30,32,35,38,41,44,45, 47,56,57,60,63,64,68,69,70,72,73,76
f	3,12,21,50,53,59,66,74
g	10,11,18,25
h	31,36,39,48,54,55

Table 4.1: Clusters of fragments for the training sets of figure 4.7 and 4.8.

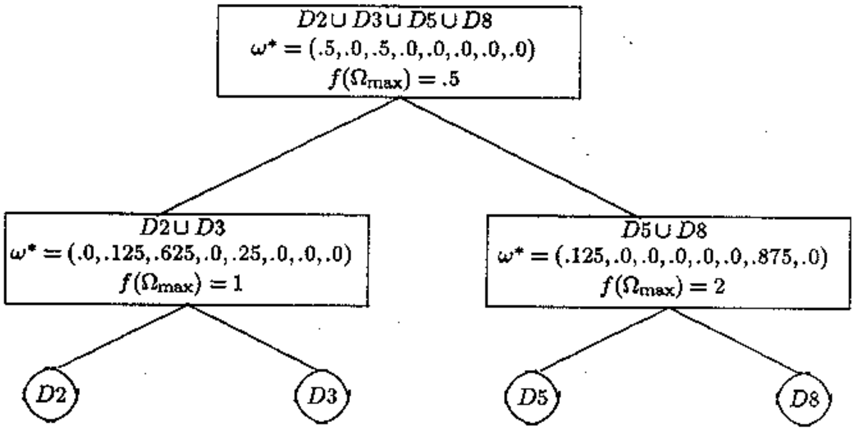
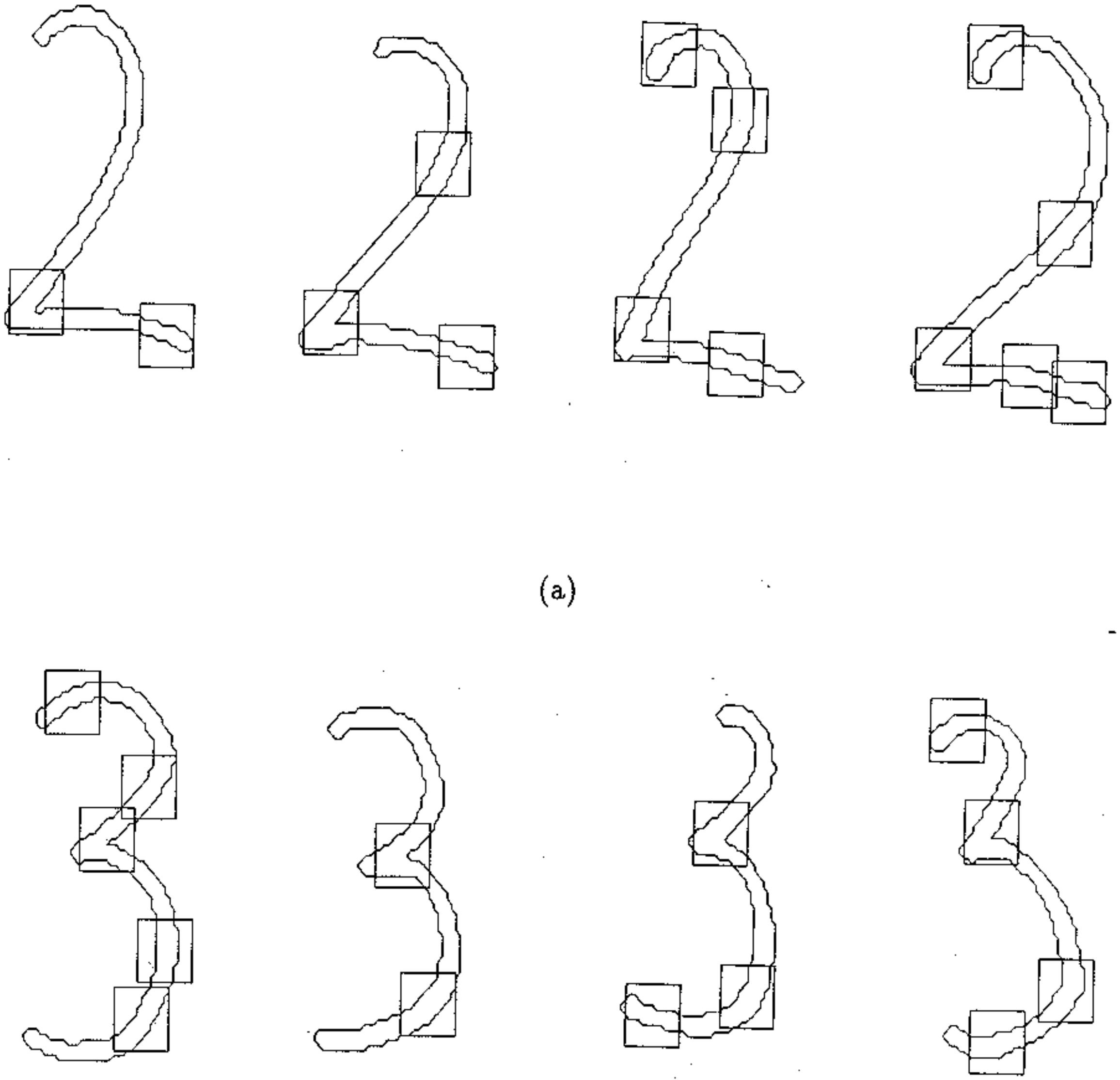

$D2 \cup D3 \cup D5 \cup D8$
$\omega^* = (.5,.0,.5,.0,.0,.0,.0,.0)$
$f(\Omega_{\max}) = .5$
$D2 \cup D3$
$\omega^* = (.0,.125,.625,.0,.25,.0,.0,.0)$
$f(\Omega_{\max}) = 1$
$D5 \cup D8$
$\omega^* = (.125,.0,.0,.0,.0,.0,.875,.0)$
$f(\Omega_{\max}) = 2$
$D2$
$D3$
$D5$
$D8$

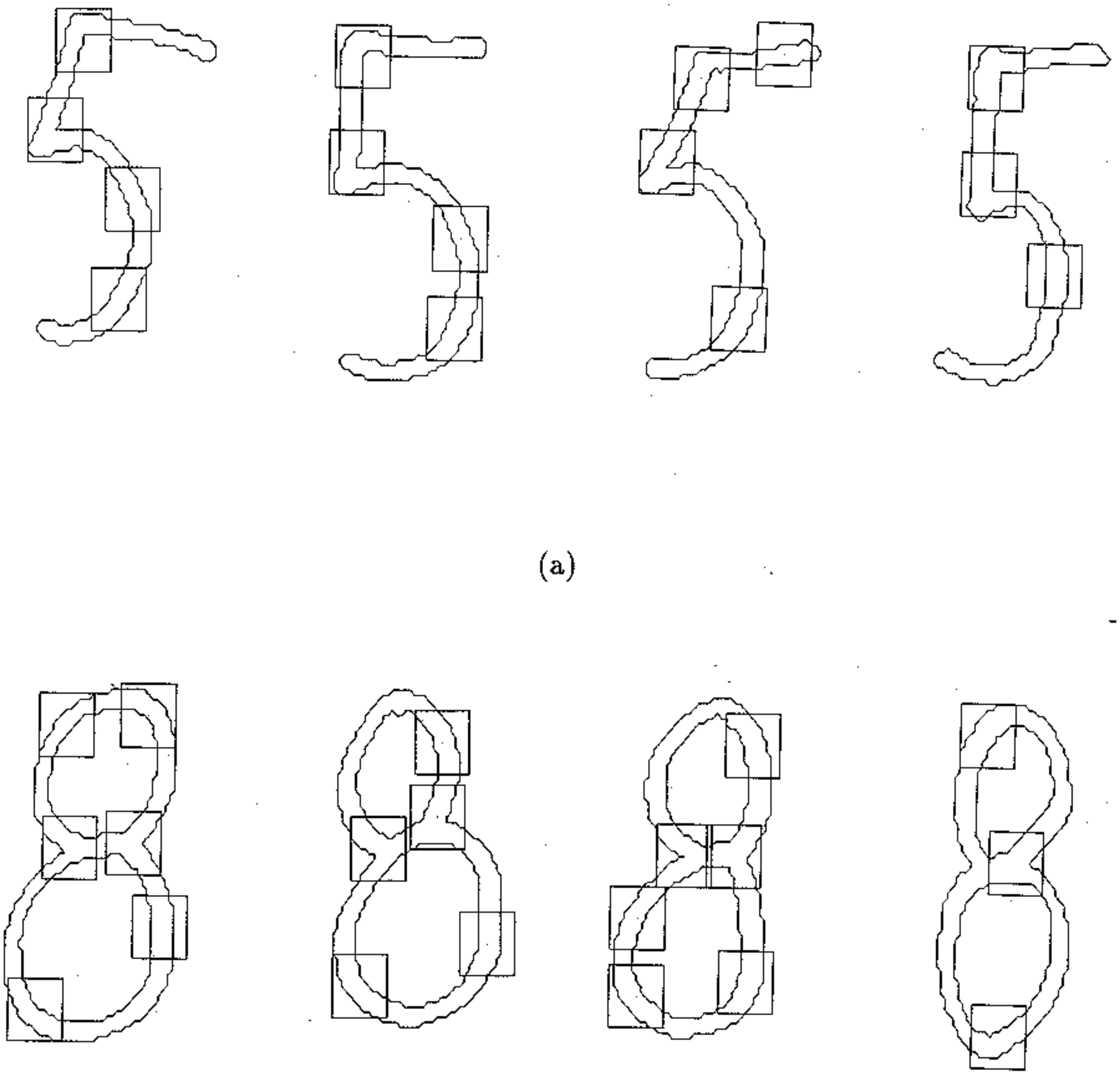
Figure 4.9: The final binary decision tree for the training sets of figure 4.8.



(a)

(b)

Figure 4.10: (a),(b) The images and the corresponding primitive fragments obtained by using defocusing mask of size 13 × 13, search window of size 13 × 13 and cutting window of size 13 × 13 for training sets $D2$ and $D3$ respectively.



(a)

(b)

Figure 4.11: (a),(b) The images and the corresponding primitive fragments obtained by using defocusing mask of size 13 × 13, search window of size 13 × 13 and cutting window of size 13 × 13 for training sets $D5$ and $D8$ respectively.

CLUSTER	FRAGMENTS
a	1,4,6,12,15,20,22,26,37,46,49,50,53,55, 59,62
b	9,10,19,25,28,39,48,61
c	29,31,33,34,40,41,42
d	2,5,7,13,14
e	32,35,45,52,58
f	24,44,51,57
g	3,8,11,16,17,18,21,23,27,30,36,38,43,47, 54,56,60,63

Table 4.2: Clusters of fragments for the training sets of figure 4.10 and 4.11.

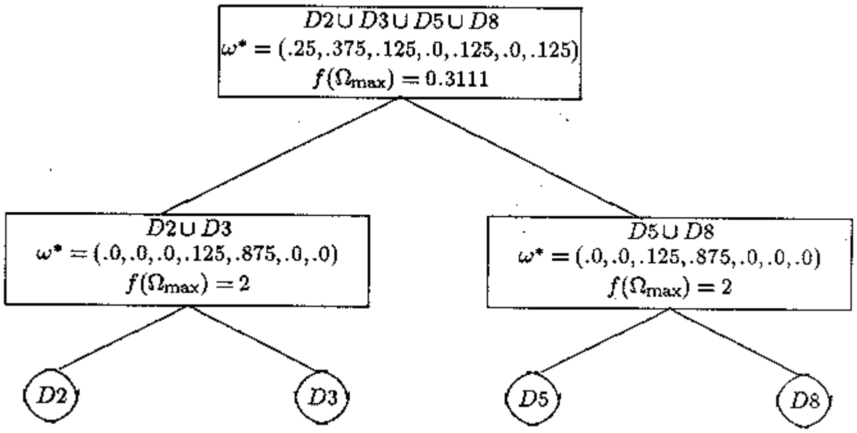
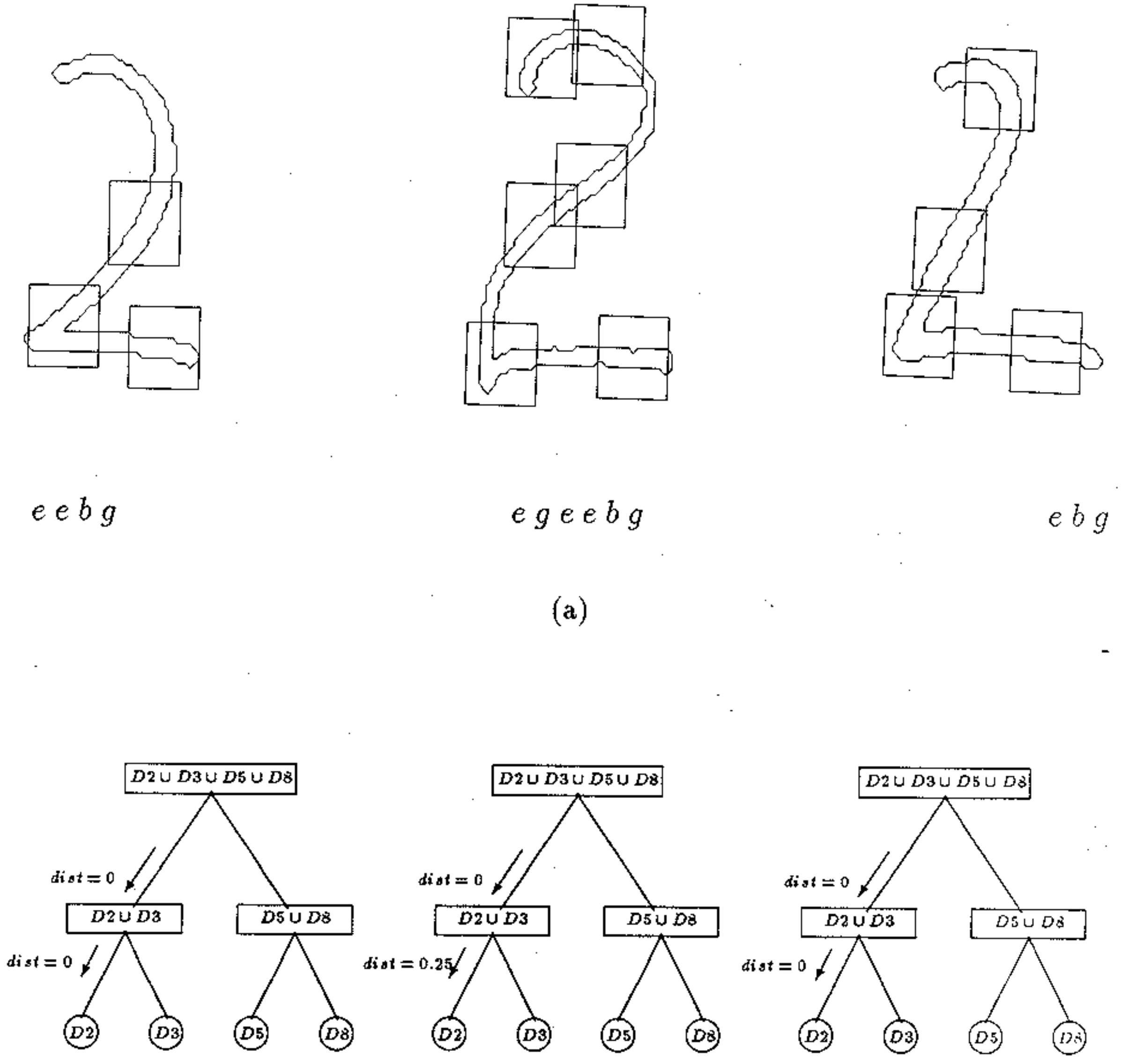

$D2 \cup D3 \cup D5 \cup D8$
$\omega^* = (.25,.375,.125,.0,.125,.0,.125)$
$f(\Omega_{\max}) = 0.3111$
$D2 \cup D3$
$\omega^* = (.0,.0,.0,.125,.875,.0,.0)$
$f(\Omega_{\max}) = 2$
$D5 \cup D8$
$\omega^* = (.0,.0,.125,.875,.0,.0,.0)$
$f(\Omega_{\max}) = 2$
$D2$
$D3$
$D5$
$D8$

Figure 4.12: An intermediate binary decision tree for the training sets of figure 4.10 and 4.11.

The values $f(\Omega_{\max})$ of each node of the final binary decision tree in figure 4.9 are satisfactory, whereas those in the figure 4.12 are not. Observe also the difference between the primitive fragments obtained by using the final and the intermediate parameters for primitive selection (figures 4.7,4.8 and figures 4.10,4.11). We can see that using the intermediate set of parameters (figure 4.10,4.11), some essential primitive fragments such as the intersection in digit 8 was not detected. This situation can only be corrected by reselecting the primitive fragments, i.e. changing the parameters of selection. On the other hand, several unessential primitive fragments are also added (figure 4.7,4.8). These fragments, however, are in fact ignored since the weight learning process assigns zero weights to the insertion/deletion operations corresponding to those primitive fragments.

The test patterns consist of digit 2, 3, 5 and 8 (three of each). Their images and the corresponding primitive fragments obtained by using the final parameters of selection, i.e. defocusing mask of size 9×9, search window of size 5×5 and cutting window of size 17×17, and the corresponding strings representing the images are shown in figures 4.13,4.14,4.15 and 4.16. The binary decision trees with the corresponding minimum distances for each node of the tree (computed using the corresponding weighting scheme) are given as well. Since we use only the final binary tree which are already presented in figure 4.10 all the time, the weighting schemes and the values $f(\Omega_{\max})$ are not shown in the figures. All digits are recognized to be in the proper classes, therefore we can see that the final binary decision tree generated by the dynamic system is the right decision tree for solving the recognition problem.



(a)

(b)

Figure 4.13: (a) Three digits 2 with the corresponding primitive fragments obtained by using the final parameters for selection and the corresponding string representations, (b) The binary tree with the corresponding minimum distances.

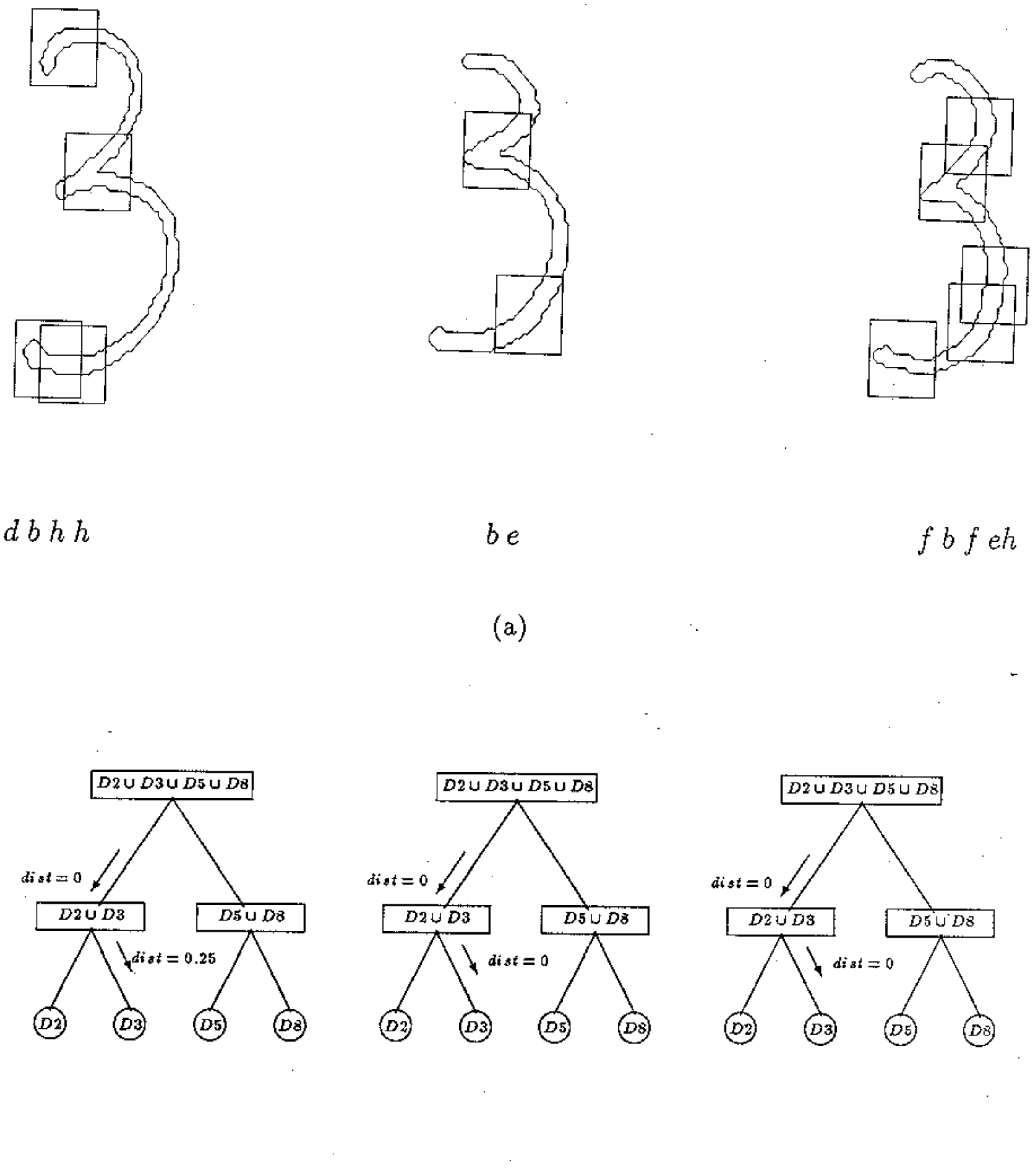


Figure 4.14: (a) Three digits 3 with the corresponding primitive fragments obtained by using the final parameters for selection and the corresponding string representations, (b) The binary tree with the corresponding minimum distances.

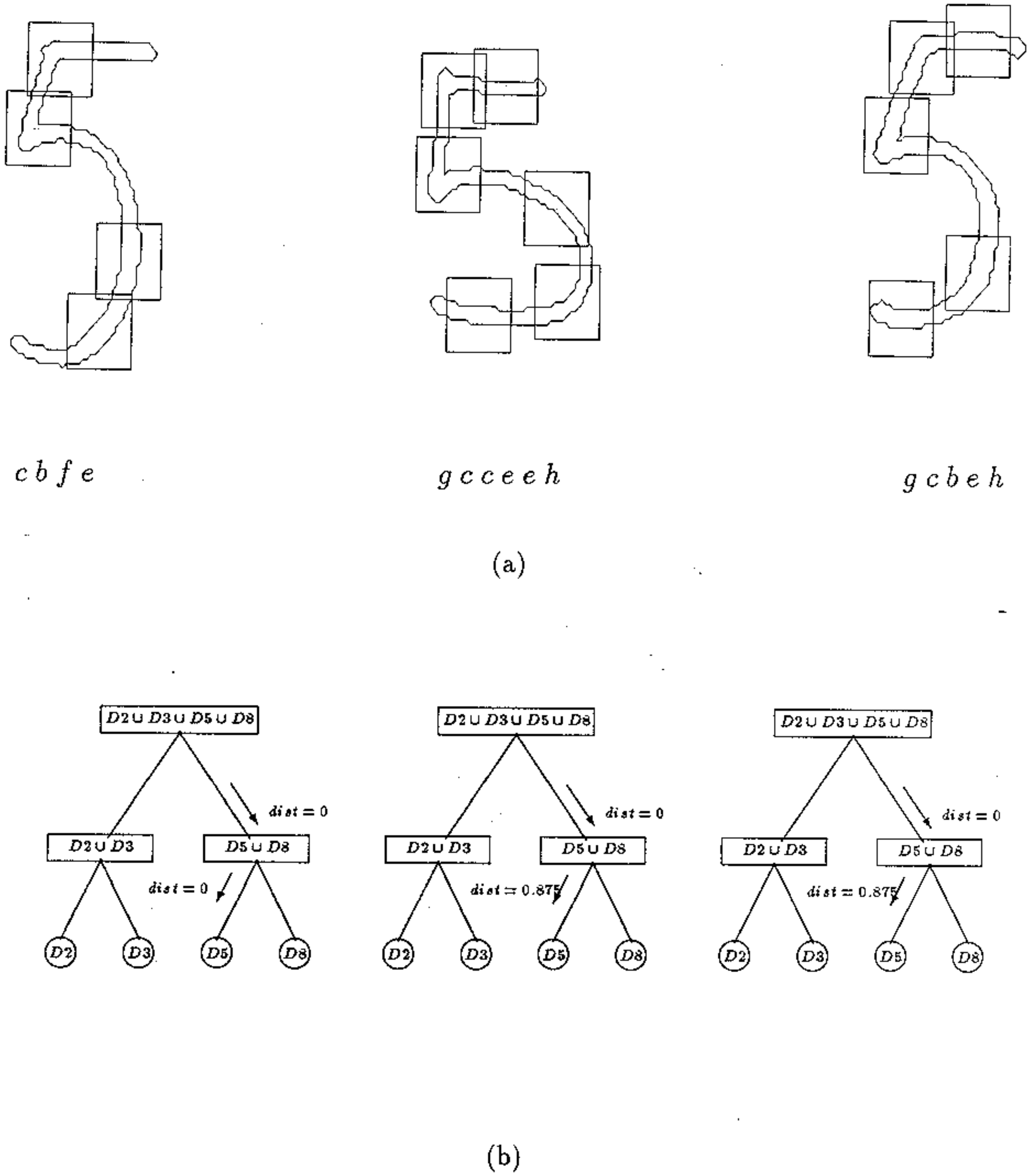


Figure 4.15: (a) Three digits 5 with the corresponding primitive fragments obtained by using the final parameters for selection and the corresponding string representations, (b) The binary tree with the corresponding minimum distances.

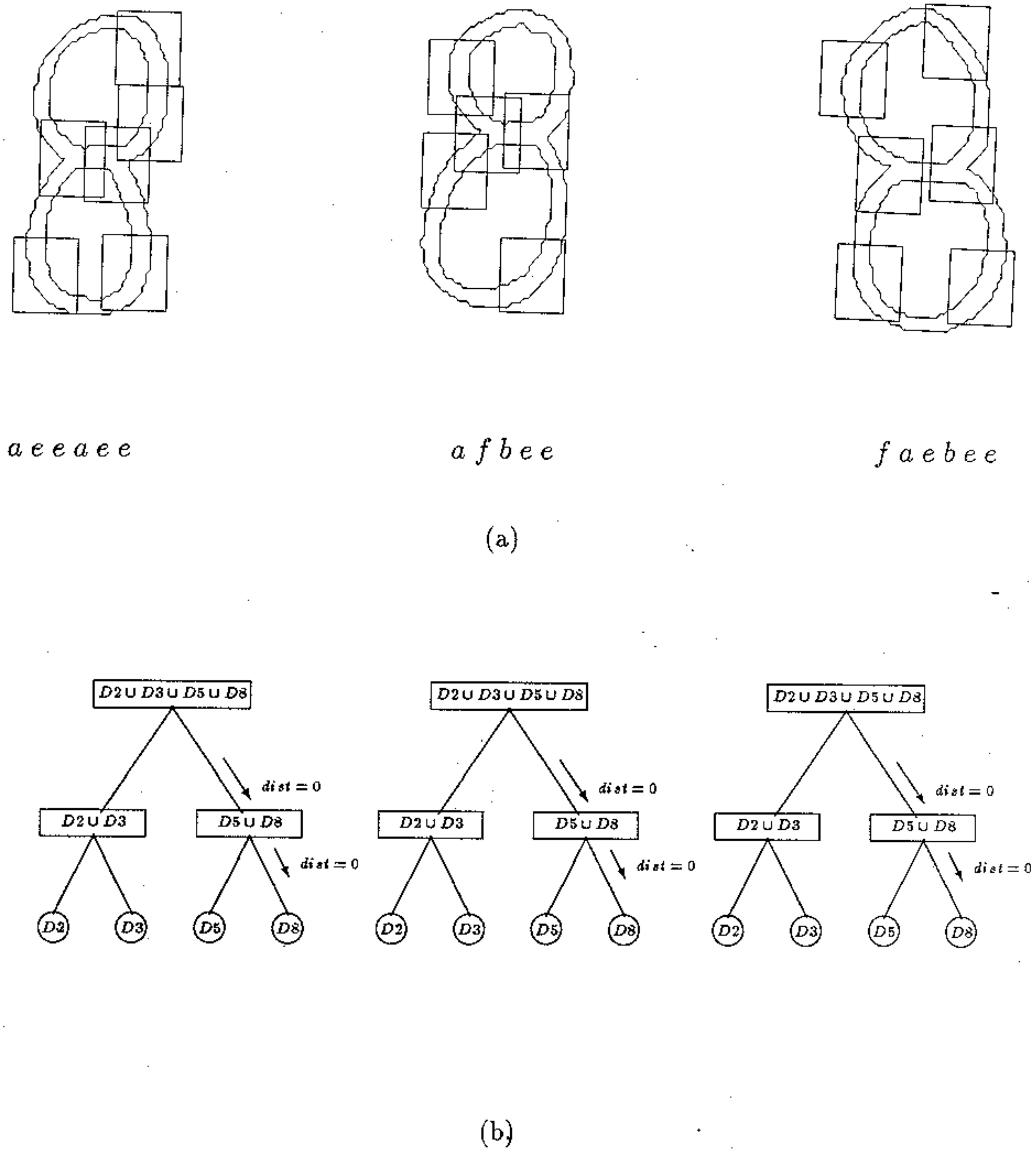


Figure 4.16: (a) Three digits 8 with the corresponding primitive fragments obtained by using the final parameters for selection and the corresponding string representations, (b) The binary tree with the corresponding minimum distances.

Chapter 5

Conclusion and Future Work

The dynamic selection of primitives in the metric model for pattern learning is very different conceptually from the existing models and, therefore, provides new freedom and flexibility. No system prior to this one makes use of the high-level analysis to drive the low-level analysis. The high-level analysis, i.e. the learning stage in the metric model, is able to correct, if necessary, some errors that occur in the low-level analysis, i.e. the primitive selection. If the errors can not be corrected, the high-level analysis commands the low-level analysis to repeat the primitive selection process.

The result of this thesis shows how easy and naturally the model handles the handwritten digit recognition problem. This ease promises us that the model should also work for other types of problems.

For the future work on the dynamic selection of primitives using the Muchnik method in the metric model, the author suggests the following topics :

- Instead of having only one set of parameters for the primitive selection, one may want to develop a system with several sets of parameters.

- The simple insertion/deletion operations in the distance computation are, perhaps, not enough for more complex problems. The introduction of several other operations may be necessary.

- An extension of the Muchnik method to the gray level images for texture analysis may produce very interesting facts.
- Instead of encoding images as one-dimensional string, a more sophisticated encoding scheme may be used. Thus, the usage of the two- dimensional iconic indexing [4] can be considered. Also in these cases the distance algorithm will be replaced.

References

[1] Baker, M.A. and M. Loeb (**1973**). *Implications of measurement of eye fixations for a psychophysics of form perception.* Perception and Psychophysics, Vol. 13, no. 2, pp. 185-192.

[2] Bozkov, V., Z. Bohdanecky, T. Radel-Weiss, L. Mitrani and N. Yakimoff (**1982**). *Scanning open and closed polygons* . Vision Res., Vol. 22, pp. 721-725.

[3] Chan, T.Y.T. and L. Goldfarb (**1989**). *Primitive pattern learning.* 2nd UNB Application of Artificial Intelligence Workshop, pp. 100-115. School of Computer Science, UNB, Fredericton.

[4] Chang, Shi-Kuo, Qing-Yun Shi and Cheng-Wen Yan (**1987**). *Iconic indexing by 2-d strings.* IEEE Trans. on PAMI, Vol. PAMI-9, pp. 413-427.

[5] Fu, K.S. and J.K. Mui (**1980**). *A survey on image segmentation* . Pattern Recognition, Vol. 13, pp. 3-16.

[6] Fu, K.S. (**1982**). *Syntactic Pattern Recognition and Applications.* Prentice-Hall, Inc., N.J.

[7] Gaarder, K.R. (**1975**). *Eye Movements, Vision and Behaviour.* Hemisphere Publishing Co., Washington.

[8] Goldfarb, L. (**1990**). *On the foundations of intelligent processes - I. An evolving model for pattern learning.* Pattern Recognition, Vol. 23, no. 6, pp. 595-616

[9] Goldfarb, L. *What is distance and why do we need the metric model for pattern learning ?.* To appear in Pattern Recognition.

[10] Goldfarb, L. *A unified metric model for pattern learning* . IASTED International Symposium on Machine Learning and Neural Network, New York, 10-11 October 1990.

[11] Granovskaya, R.M. and I.J. Bereznaya (**1980**). *Experiments on human pattern recognition : A hierarchical sign-system approach.* Pattern Recognition, Vol. 12, pp. 17-26.

[12] Haralick, R.M. and L.G. Shapiro (**1985**). *Survey : Image segmentation techniques* . Comp. Vision, Graphics and Image Processing, Vol. 29, pp. 100-132.

[13] Kittler, J. and J. Illingworth (**1986**). *Minimum error thresholding.* Pattern Recognition, Vol. 19, no. 1, pp. 41-47.

[14] Mackworth, N.H. and A.J. Morandi (**1967**). *The gaze selects informative details within pictures* . Perception and Psychophysics, Vol. 2(11), pp. 547-552.

[15] Muchnik, I.B.(**1972**). *Simulation process of forming the language for description and analysis of the forms of images* . Pattern Recognition, Vol. 4, pp. 101-140.

[16] Noton, D. and L. Stark (**1971**). *Eye movements and visual perception* . Scientific American, June, pp.35-44.

[17] Pavlidis, T. (**1976**). *Structural Pattern Recognition* . Springer-Verlag, New York.

[18] Rosenfeld, A. (**1984**). *Image analysis : Problems, progress and prospects* . Pattern Recognition, Vol. 17, no. 1, pp. 3-12.

[19] Tou, J.T. and R.C. Gonzalez (**1974**). *Pattern Recognition Principles.* Addison-Wesley Publishing Co., Massachusetts.

[20] Wagner, R.A. and M.J. Fischer (**1974**). *The string-to-string correction problem.* Journal of ACM., Vol. 21, no. 1, pp. 168-173.

[21] Zavalishin, N.B. and I.B. Muchnik (**19**). *Models of Visual Perception and Image Analysis Algorithms.* (In Russian), Nauka, Moscow., CA.

VITA

Candidate's full name : Sonya Dewi
Place and date of birth : Blora, Indonesia,
April 5, 1963.
Permanent address : Jl. Pemuda no. 41,
Blora (Ja-Teng),
Indonesia.
Schools attended : Elementary School
Blora, Indonesia,
1968 - 1972.
: Blora Yunior High School I
Blora, Indonesia,
1973 - 1976.
Blora Senior High School I
Blora, Indonesia,
1976 - 1980.
Universities attended : Brawijaya University
Malang, Indonesia,
BSc.(Agriculture)
1980 - 1984.