SEGMENTATION AND RECOGNITION OF CONNECTED HANDWRITTEN DIGITS

by

Lu Zhongkang, BEng, MEng

A thesis submitted for the degree of Master of Philosophy

in the Department of Electronics and Information Engineering

of the Hong Kong Polytechnic University

Department of Electronics and Information Engineering

The Hong Kong Polytechnic University

May 1999

Abstract

Recognition of connected handwritten characters is a challenging task due mainly to two problems: poor character segmentation and unreliable isolated character recognition. Overlapping, touching, and ligatures between neighboring characters make character segmentation and then recognition very difficult. This thesis presents our research results on connected handwritten character recognition using a segmentation-based approach. The three main problems that have to be solved are the estimation of the number of character in a word, character segmentation, and reliable isolated character recognition. We discuss in this thesis a neural network based length estimation, a background-thinning-based segmentation algorithm, a new template representation and optimization technique for building a template based classifier, and dynamic programming techniques used in segmentation-based or segmentation-free approach for recognizing connected characters. We used digit strings as examples to evaluate our new algorithms.

Length estimation is very helpful for the successful segmentation and recognition of connected handwritten digits. The kernel of our algorithm is a neural network estimator with a set of statistical and structural features as the input. To extract features, several preprocessing steps including noise reduction, normalization and skeletonization have to be carried out. The output of the neural network is a set of fuzzy membership grades reflecting the degrees of an input digit string for having different number of digits. In experiments, we consider up to 4-digit strings (very few 5-digit or longer strings in real applications). NIST Special Database 3 was used for training and testing the neural network length

estimator. The database includes 20,852 isolated digit samples, 4,555 connected 2-digit samples, 355 connected 3-digit samples and 48 connected 4-digit samples. Because we do not have many 3-digit strings and 4-digit strings, we artificially generate some by merging the existing samples of digits and digit stings. Experimental results on NIST Special Database 3 and derived digit strings show that only 55 (0.6%) out of 9910 digit strings are poorly estimated.

Correct segmentation is vital for character recognition using segmentation-based approaches. In our approach, the segmentation is based on the analysis of the background (the regions excluding the characters) skeleton of a digit string image. In the exploration of connected digit samples, we found the shape of the background regions is much simpler than the foreground. Making use of the shape information of background can simplify the searching of the segmentation paths and decrease the number of segmentation candidates accordingly. In this algorithm, we extract segmentation paths by matching feature points on background skeletons. Feature points include fork points, terminal points, and curve points. The definition of fork points and terminal points are same as the length estimation algorithm. A curve point is a point on a segment where the direction of the line changes sharply. A three-step-matching scheme is developed to find the matched point pairs and a segmentation path is constructed by connecting these feature points with possible extension to the top and/or bottom of the skeleton. We applied our segmentation algorithm to the connected 2-digits. The membership degree to which a candidate is a good segmentation path is determined by fuzzy decision rules with the nine properties associated with a segmentation path. The separated digits are recognized by a nearest-neighbor classifier. Tested on NIST Special Database 3, our background-thinning-based approach for segmenting and recognizing handwritten digit strings show better performance than some existing techniques. Moreover, our approach can deal with both single- and multi-touching problems.

We present a multi-module classifier to recognize isolated digits. Among

four modules, a template-based classifier based on the rational B-spline surface representation of the Pixel-to-Boundary Distance Map (PBDM) is adopted to improve the performance of the classifier, in particular, in rejecting non-digit patterns. To extract optimized templates, we used a two-stage algorithm based on a neural network and an evolutionary algorithm. The classifier can reliably distinguish non-digit patterns from digits, which is a desirable feature for recognizing handwritten digit strings. The classifier can be applied together with a segmentation-based recognition algorithm or a segmentation-free recognition algorithm. Experimental results show that the designed multi-module classifier compares favorably with other classification techniques tested.

In this thesis, we also discuss a segmentation-based and a segmentation-free approaches for recognizing connect characters. Based on the designed multi-module classifier and the background-thinning-based segmentation algorithm, a segmentation-based recognition approach is presented. A dynamic programming algorithm is applied in this approach. Experimental results show that our approach can achieve more favorable classification performance. To deal with some hard-to-segment handwritten digit strings, a segmentation-free recognition approach with a dynamic programming algorithm is also discussed.

Acknowledgements

I would like to express my sincere gratitude to my supervisor Dr. Zheru Chi, for his help in all aspects of conducting research, and to my co-supervisor Professor Wan-Chi Siu for having created a very simulating and well-equipped lab environment in Department of Electronic and Information Engineering, the Hong Kong Polytechnic University, through many years of dedicated efforts.

I am grateful to all the people who helped me one way or another during those years in PolyU. Special thanks to Kong Jong, Li Xin, Wang Zhiyong for their unselfish help, and to K. C. Chu for many fruitful discussions.

Special thanks to my breaks pals, J. J. Wang, Ai Wu, Li Wei, Ringo for sharing such relaxing and fun moments. Thanks also to the other members of Institute of Image Processing and Pattern Recognition at Shanghai Jiaotong University and Department of Electronic and Information Engineering of the Hong Kong Polytechnic University for their friendship and help.

Statement of Originality

The work described in this thesis was carried out at the Department of Electronic and Information Engineering, the Hong Kong Polytechnic University, between September 1996 and September 1998, under the supervision of Dr. Zheru Chi and Professor Wan-Chi Siu.

The thesis consists of seven chapters. The work described in this thesis was originated by the author except where acknowledged and referenced, or where the results are widely known. The following is the statement of original contributions.

1. A combined morphology operations and fuzzy rules for pre-processing digit images was presented by the author. (Chapter 2, Section 2.1)

2. A neural network based technique to estimate the number of characters in a digit string was proposed jointly with Dr. Zheru Chi. Experimental results on NIST Special Database 3 and other derived digit strings show that 9855 (99.4%) of a total of 9,910 digit strings are well estimated. (Chapter 3)

3. A new digit template representation scheme, a neural network based template extraction algorithm, and an evolutionary algorithm for optimizing templates were presented by the author. (Chapter 4, Section 4.2)

4. A multi-module based classifier for recognizing isolated handwritten digits was developed by the author. Experimental results on NIST Special Database 3 show that the multi-module classifier can achieve a good recognition performance with higher reliability in rejecting non-digit patterns compared with other existing techniques. (Chapter 4)

5. A character segmentation technique based on background skeleton analysis was developed by the author. Experimental results show that the technique can achieve more favorable classification performance than several existing techniques. (Chapter 5, Sections 5.2, 5.3, 5.4)

6. A moving-window-based segmentation-free recognition algorithm was presented by the author. (Chapter 6)

Publications

Book Chapter

1. Z. Lu, Z. Chi, W. C. Siu and P. Shi, "A New Template Representation and Extraction Method for Handwritten Digit Recognition," Advances in Handwriting Recognition , S.-W. Lee (Editor), World Scientific Publishing, pp. 426-435, 1999.

International Journal Papers

1. Z. Chi, Z. Lu, W. C. Siu and P. Shi, "An Evolutionary Algorithm for Optimizing Handwritten Numeral Templates Represented by Rational B-Spline Surfaces," Journal of Advanced Computational Intelligence (to be published).

2. Z. Lu, Z. Chi, W. C. Siu and P. Shi, "A background-thinning-based approach for separating and recognizing connected handwritten digit strings," Pattern Recognition, Vol. 32, No. 6, pp. 921-933, June 1999.

3. Z. Lu, Z. Chi and W. C. Siu, "Length Estimation of Digit Strings Using a Neural Network with Structure Based Features," Journal of Electronic Imaging, Vol. 7, No. 1, pp. 79-85, January, 1998.

International Conference Papers

1. Z. Lu, Z. Chi and P. Shi, "A background-thinning based approach for separating and recognizing connected handwritten digit strings," Proceedings of

1998 International Conference on Acoustics, Speech and Signal Processing (ICASSP'98), Vol. II, pp. 1065-1068, May 12 -15, 1998, Seattle, U.S.A.

2. Z. Chi, Z. Lu and F. H. Chan, "Multi-Channel Handwritten Digit Recognition Using Neural Networks," 1997 IEEE International Symposium on Circuits and Systems, pp. 625-628, June 9-12, 1997, Hong Kong.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Automatic character recognition, a very active research area, has found many applications, such as form and cheque reading. Character recognition can either be on-line or off-line. On-line recognition refers to those systems where the data to be recognized is from a tablet digitizer that acquires the positions of the pen tip as the user writes in real-time. In contrast, off-line system obtains the data from a document through an acquisition device, such as a scanner or camera.

The off-line reading of characters by computer known as Optical Character Recognition (OCR) is a topic that has been investigated for many years [1, 2]. Now both machine printed and isolated handwritten characters can be recognized with good performance by machines. A variety of systems are now in commercial use for processing printed materials, forms, etc. However the OCR techniques have not yet been successfully applied to read cursive handwritings. The problem remains extremely challenging due to:

1. There exist many different writing styles. Each person has his/her own writing style that may change according to his/her physical and psychological conditions.

2. The perfect segmentation of connected characters into individual characters is very difficult to achieve if not impossible.

3. There exist ligatures that are formed at the end of strokes by dragging the pen to the next stroke, which make the segmentation and then recognition very challenging.

4. Poor writing and thinning/thresholding processing may cause broken, touching and merged characters.

Furthermore, compared with on-line recognition, off-line recognition is much more difficult because it has to be done without knowing pen movement. As a result, the off-line recognition has attracted much more attention than the on-line recognition [3, 4, 5].

1.1 Review of Previous Work

This section reviews some of the handwritten recognition algorithms that have been detailed in the literature. Here only a brief review of these algorithms is given. More detailed discussion are given in later chapters when particular issues are addressed.

1.1.1 Isolated Digit/Character Recognition

Suen *et al* [1] provided a good review of handwriting recognition up to 1980, concentrating on isolated character recognition, which had been a focus of research until then. They discussed a variety of feature based approaches which were categorized into global features (templates and transformations such as Fourier, Walsh and Hadamard transforms); point distributions (zoning, moments, n-tuples, characteristic loci, crossings, and distances), and geometrical or topological features. The third category were and have remained one of the most interesting topics. The approach involves separate detectors for each of several types of features such as loops, curves, straight lines, end points, angles and intersections. Impedovo *et al* [6] used cross-points, end-points and bend-points

as features, which are coded based on their locations in three horizontal and three vertical zones within each character. The encoded characters are identified using a decision-tree classifier. Chi *et al* [7] also used features such as curves and loops, with each of which is associated with a set of numerical quantities, such as type, length, and mass center location, before being decoded in a fuzzy rule classifier. This classifier is combined with Markov chain matching and a three-layer perceptron with pixel intensities as the input to deliver final classification. Senior *et al* [8, 9] proposed a morphological presentation of characters in which the normalized character image is partitioned into frames and the features in a frame (dots, junctions, end-points, turning points, line segments, and loops) are given a number and the whole frame is assigned with a vector that can be sent to a recurrent neural network for classification.

A host of other authors have tackled the problem of recognition isolated digits or characters in the past few years. The algorithms differs in the types of features used and classification methods employed. Two comprehensive reviews were given by Govindan [10] and Impedovo [6]. Trier *et al* [11] also presented an exhaustive survey of feature extraction methods for off-line recognition of isolated digits/characters. Lee *et al* [12] proposed a classifier based on Radial Basis Network and Spatio-Temporal Features. Lee *et al* [13] presented another classifier based on Multilayer Cluster Neural Network and Wavelet features. Cheng *et al* [14] proposed to use a classifier based on morphological operations.

There are a number of studies reported in the literature that have applied template-based techniques to isolated digit recognition. The Template matching based technique, which is one of the earliest pattern recognition techniques for digit recognition, has re-attracted much attention of many researchers in the past few years with advanced computational algorithms such as artificial neural networks and evolutionary algorithms. A number of studies have been reported in the literature, which have applied template-based techniques to digit recognition. Yan proposed an Optimized Nearest-Neighbor Classifier (ONNC) for the

recognition of handprint digits [15], the templates are represented by 8×8 gray-scale images, re-scaled from the original 160×160 normalized binary images. Wakahara used iterated Local Affine Transformation (LAT) operations to deform binary images to match templates [16]. The templates used are 32×32 binary images. Nishida presented a structure-based template matching algorithm [17]. The templates are composite with straight lines, arcs and corners, and the matching is based on the geometrical deformation of the templates. Cheung *et al* proposed a template representation based on splines [18]. They modeled a digit images using a spline and assumed the spline parameters have multivariate Gaussian distributions. Revow *et al* gave another digit recognition algorithm based on spline templates [19]. The templates are elastic spline curves with ink-generating Gaussian "beads" strung along their length. Jain *et al* presented a deformable templates matching algorithm based on object contours [20, 21]. In their approach, the recognition procedure is to minimize the objective function by iteratively updating the transformation parameters to alter the shapes of the templates so that the best match between the templates and unknown digits is determined. Jain also proposed a review of applying deformable templates on various matching problems [22].

Using a combination of multiple classifiers seems to be a more promising way of reducing error rate than finding a better classifier. Some of discussions in this topic can be found in [23, 24, 25, 26, 27, 28, 29]. The performance of recognition of isolated digit/character is believed to be good enough for practical applications. The classifier only fail to classify those digits that are entirely ambiguous and even human beings could not confidently classify [30]. Because most of classifiers for recognizing isolated characters are not knowledge driven, the classifiers are usually much poorer in rejecting non-characters than human readers. More work needs to be done in this direction.

1.1.2 Handwritten Digit String/Script Recognition

The recognition of handwritten digit strings/scripts has received a lot of attentions recently because it has many applications, such as mail sorting, automatic form reading, etc. The recognition techniques developed can be categorized into different classes by different criteria. By the objects to be recognized, they can be divided into *word recognition* and *handwritten digit string recognition*; by the techniques applied, they can be divided into *segmentation-based recognition* and *segmentation-free recognition.*

Word recognition and handwritten digit string recognition adopted the similar procedure in earlier studies. (1) some technique is utilized to separate scripts (handwritten digit string or word) into individual segments [31, 32, 33, 34, 35, 36]; (2) these segments are then sent to a classifier and a ranked list of possible words or digit strings is generated based on the recognition results. Such a technique is commonly referred to as *segmentation-based recognition* or *analytical approach.* With the adaptation of the hidden Markov model from speech recognition to handwritten script recognition [37, 38, 39, 40, 4, 41, 42] and applying lexical information (contextual information) to word recognition [43, 44], a variety of segmentation-free techniques have been developed for recognizing handwritten words and digit strings. The recognition of connected digits mainly use a segmentation-based approach, while the recognition of words mainly use segmentation-free approaches because that there is little lexical information in connected digits that can applied to improve the recognition accuracy as the word recognition.

1.1.2.1 Segmentation-based Approaches

Two comprehensive overviews of the segmentation techniques for machine printed characters and handwritten words can be found in [45, 46], respectively. Cheriet *et al* presented a region-based background analysis algorithm to find a married

pair of background valleys in order to separate a handwritten digit string [47]. On the other hand, a context-directed hierarchical algorithm was proposed by Shridhar and Badreldin to separate handwritten 2-digit strings [34]. In their approach, the segmentation was done based on the analysis of horizontal border-to-background transitions. Yu and Yan have recently proposed a recognition-based segmentation algorithm [48] in which a set of structure based models together with several criteria were adopted to select the most promising one-touching point or pair of two-touching points at which the string was separated. Another contour analysis based algorithm was proposed by Strathy *et al* [49] in which contour information was used to find pairs of cutting points. Nine credits on contour features were then assigned to each pair of cutting points and the weighted sum of nine credits was used to prioritize cut links with the weights trained using a genetic algorithm. Gader *et al* [50] also proposed a contour information based segmentation algorithm in which a distance transform is used to detect and split the initially separated components that may contain more than one characters. In another study, an integrated segmentation algorithm was proposed by Pervez *et al* [51].

Two main concerns in the segmentation-based approaches are:

1. the segmentation algorithm must be able to identify all possible segmentation paths. However, the number of the segmentation candidates to be tested should be as small as possible for real-time processing;

2. Reliable classification of isolated character and rejection of non-characters is a key factor for achieving a better system performance.

1.1.2.2 Segmentation-free Approaches

Since character segmentation algorithms are prune to error and designing a robust classifier is very difficult, many segmentation-free approaches for the recognition of handwritten scripts have been proposed [37, 38, 52, 40, 41, 5, 4, 39, 42,

53, 9]. Many of these techniques are based on Hidden Markov Models (HMMs) that have been successfully applied to speech recognition. Some of these methods also make use of lexical information to improve the recognition performance.

The statistical methods of hidden Markov modeling were initially introduced and studied in the late 1960s. It can be briefly described as a stochastic process generated by two interrelated mechanisms, an underlying Markov chain having a finite number of states, and a set of random functions, one of which is associated with each state. At discrete instant of time, the process is assumed to be in some state and an observation is generated by the random function corresponding to the current state. The underlying Markov chain then changes states according to its probability matrix. The observer sees only the output of the random functions associated with each state and cannot directly observe the states of the underlying Markov chain, hence it is termed as hidden Markov model [41].

HMMs have been found extremely useful for a wide spectrum of applications. This technique was first applied to speech recognition problems. Since there are many similarities between handwritten word recognition and speech recognition, we have also seen many applications of HMMs in handwritten word recognition.

Another segmentation-free algorithm, a subgraph matching algorithm, was presented by Rocha *et al* [54, 55, 56]. The algorithm extracts meaningful subgraphs (characters or digits) from the feature graph (word) by template matching.

1.2 Psychology of Writing and Reading

Before attempting the machine recognition of handwritings, it is worthwhile considering the way that people read and write. Considering human reading may lead to an better understanding of the transfer of information through the medium of handwriting, so that it can be seen which processes play a useful role, and which are merely epiphenomena. If it can be understood what information people use to recognize handwritten digit strings, then a clue is found as to

what features might be useful for a machine recognition system. Understanding handwriting production may give insights as to which features of handwriting are representations of the information and which are mere artifacts of generation process.

A large body of psychological data has been gathered on the processes involved in reading type, some of which is applicable to handwritten digit strings. Most research so far has concentrated on reading individual letters or words out of context. It could be argued that this gives little indication of the processes occurring in normal reading where many words are visible and it is the text as whole, not individual words, that is important. However, results are hard to prove in such a natural environment with many variables, and it is only under restricted experimental conditions that hypotheses can be rigorously tested.

Research into reading, as in much of psychology, relies heavily on observing what errors are made under difficult conditions. One technique is the use of tachistoscopes to flash a word in front of a subject for a very short time followed by a patterned mask to inhibit iconic memory, which otherwise allows the subject to preserve an image of the word mentally for an uncontrolled period of time.

As well be seen later, many approaches to handwriting recognition rely on detecting features in the writing, such as the strokes which go to make up individual letters. It is discovered that the complex cells code the presence of bars and edges and provide a compact representation of lines which is particular appropriate to the representation of writing and print. A number of authors have sought to determine what higher-level representation might be used specifically for letters. Senior [30] and Guillevic [57] gave us some overviews of the previous researches and some conclusion can be concluded:

1. In conjunction with a lexicon of permitted words, a few simple features can identify most words;

2. the character recognition is carried out by finding word features that fill roles in internal models of characters;

3. recognition of words as single entities and not as the conjunction of their component characters is the same important in human recognition system.

The research results formed the foundation of the lexicon-driven and word-driven recognition technique. However, these researches can not provide help to the recognition of handwritten digit strings, because there is not any lexical information in digit strings.

Many studies have also been made into the processes involved in writing. If an accurate model of these processes can be found, then it could be used for representation of handwriting in a compact form, and for recognition. Several researches that can help to removal slant, slope and other variations have been applied to on-line script where the pen trajectory is accurately known. The static nature of off-line writing does not lend itself to these approaches. The reason is that the extraction of pen-moving information is easy to be failure.

1.3 The Database of Handwritten Characters Used in Our Experiments

We used NIST (National Institute of Standards and Technology) Special Database 3 to evaluate the algorithms that we developed. The database contains more than 800,000 images with hand-checked classification from 3600 writers. Figure 1.1 and figure 1.2 are some digit samples (isolated and connected digits) from the database. We can see that they are different in size, slant degree and writing style. The connection, overlap between neighboring digits, as well as tailors and noise increase the difficulty of recognition.

In total, we extracted from the database 20,852 isolated digits, 4,555 handwritten two-digit strings, 355 handwritten three-digit strings and 48 handwritten

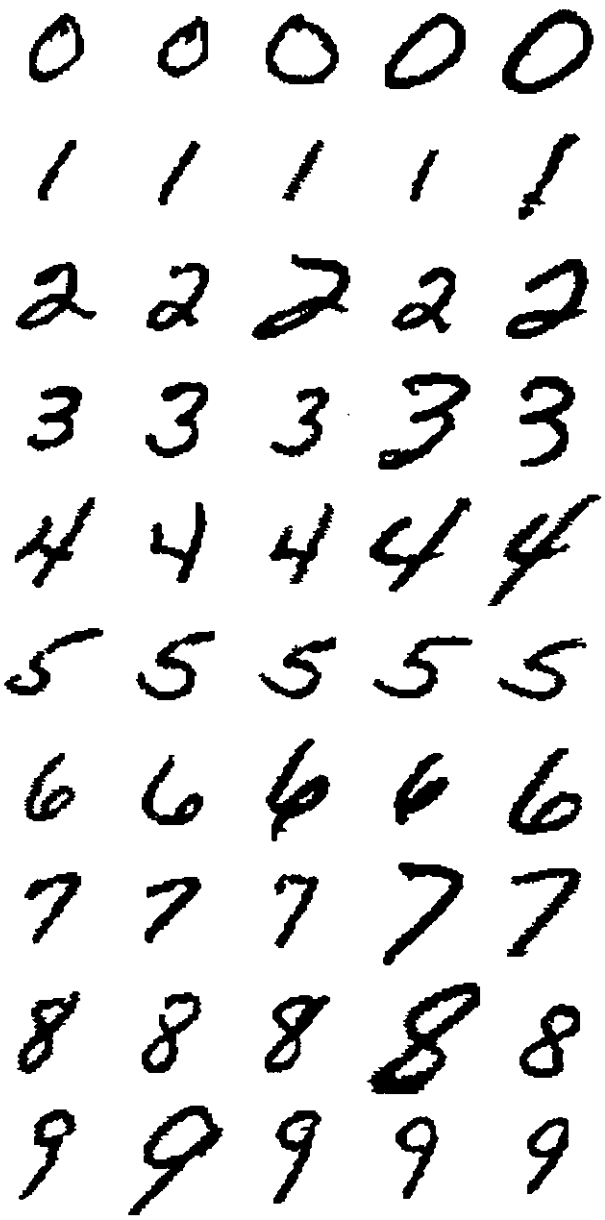


Figure 1.1: Some isolated digit samples extracted from NIST Special Database 3

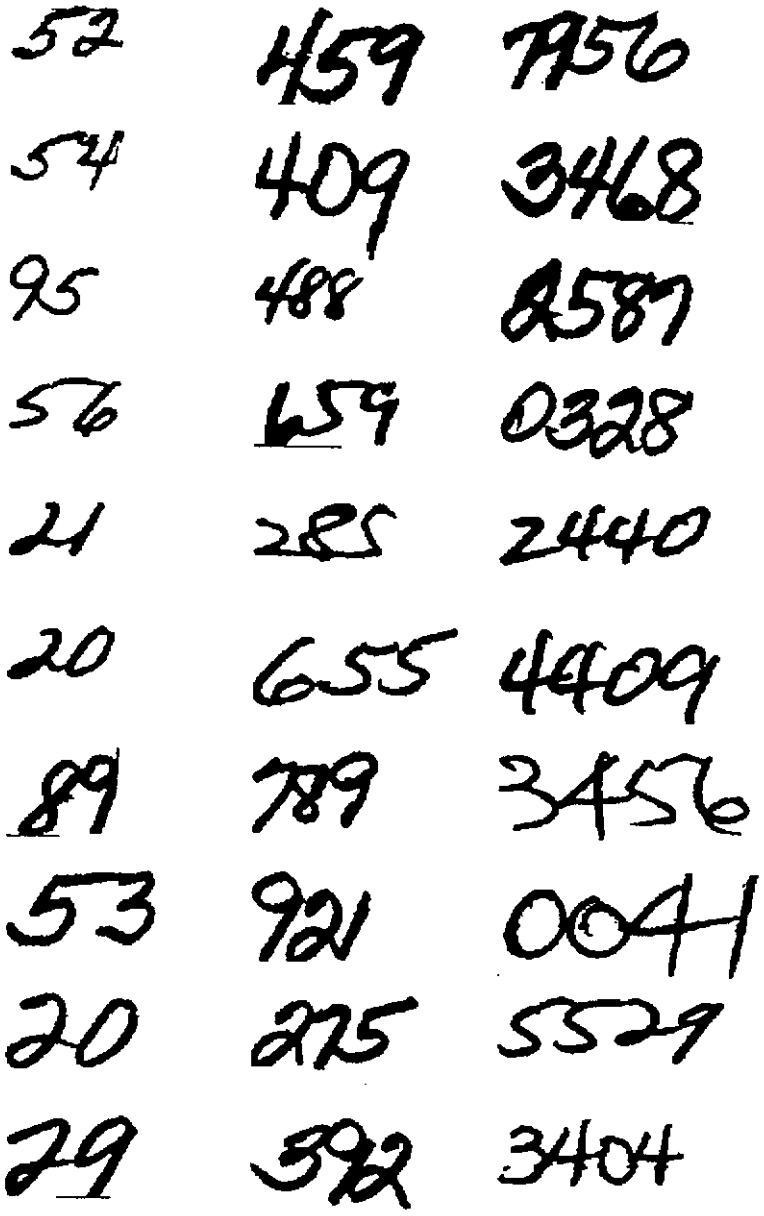


Figure 1.2: Some handwritten digit strings extracted from NIST Special Database 3 (left: handwritten two-digit strings; center: handwritten three-digit strings; right: handwritten four-digit strings).

four-digit strings, and some artificially generated handwritten digit strings for evaluating the algorithms that we developed.

1.4 Organization of This Thesis

This thesis is organized as follows. Following this "Introduction", we describe in Chapter 2 the pre-processing of digit string images including smoothing, normalization, and length (the number of digits in a digit string) estimation. Chapter 4 discuss our design of a multi-channel based digit classifier. A background-thinning based segmentation and recognition algorithm is presented in Chapter 5. In Chapter 6 a moving-window approach is proposed to recognize handwritten digit strings without segmentation. Concluding remarks are drawn in Chapter 7.

Chapter 2

Pre-processing of Character Image

Pre-processing is a crucial step for handwritten character recognition since it can greatly affect the features to be extracted for the recognition [58]. There are two main tasks in the pre-processing: smoothing and normalization.

Smoothing of an image consists of filling and thinning. Filling is a process whereby gaps and breaks are eliminated, and thinning is to remove noise, bumps and isolated bits. Heavy thinning is also referred to as skeletonization. Normalization is another aspect of pre-processing process. Elements are normalized include the character's size, position and skew, as well as line widths.

Essentially, the pre-processing using various techniques including filling of holes, skeletonization, centering, rotating, skewing, thickening, shearing and size normalization.

The pre-processing for the length estimation of connected digit strings include three steps: noise filtering, size normalization, and skeletonization, as shown in Fig. 2.1. Two processed images are output from each input image, the normalized noise-free image and the skeleton image, for later processing.

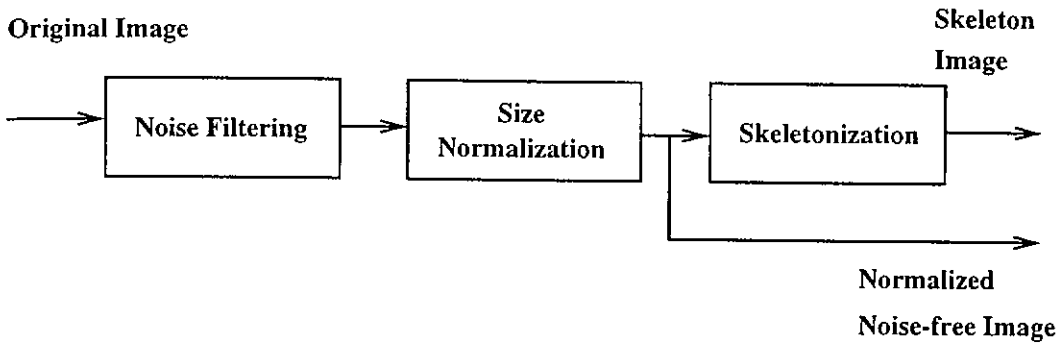


Figure 2.1: Flowchart of the pre-processing steps for the length estimation of connected digit strings.

2.1 Noise Filtering

Mathematical morphology, which can be used to handle geometrical features in an image, has been studied for about three decades. Using *a prior* geometrical information, the features in an image can be extracted, suppressed, or preserved by applying morphological operators [59, 60]. Mathematical morphology consists of set transformations that transform one set into another set. These transformations are carried out via the use of a structure element which contains the desired geometrical structure. Various interactions of the original set with the structuring element form the basis of all morphological operations [61].

There are four basic morphological operators: erosion, dilation, opening and closing. In our study, the opening and closing operations are used to extract noise from an original image.

Let I be an original image and B is a structural element. The noise N can be defined as

$$N = I - (I \circ B) \bullet B \tag{2.1}$$

where '$\circ$' denotes the opening operation, and '$\bullet$' denotes the closing operation.

The choosing of the structure element B is the most difficult problem in mathematical morphology analysis. The reason is that there exist a huge offering of potential structure elements for the vastness of the universe of all possible

object shapes, and experience shows that it is impossible to handle a structure element very well in the processing. The choosing is very much depend on what we want to get from the object and the property of the original data. In this algorithm, a 3×3 lattice was chosen as B considering the size and shape of the noise in the original image.

Figure 2.2(a) shows a few examples of digit string images. Possible noise (dark regions) extracted by morphological operations are shown in Fig. 2.2(b). The possible noise regions include:

- burrs on the border of foreground due to image scanning and binarization,
- small holes from the morphological operations,
- large artifacts from writing boxes, and
- narrow strokes due to poor writing, such as the narrow part at the top of '2' in the forth digit string in Fig. 2.2(b).

Burrs and large artifacts should be erased, small holes should be ignored, and narrow strokes should be kept to avoid broken strokes. Production rules are used to distinguish a noise region from other foreground regions according to its size, shape, and the number of connected foreground regions.

The size of a potential noise region, f_n, is given by its total pixel number. The shape of the region, f_s, is defined as the ratio of the number of boundary pixels to the number of inner pixels. Let the number of connected foreground regions be denoted by f_p. A potential noise region with $f_n < 5$ is removed immediately without any further processing. However, three rules are used to distinguish larger noise regions from the other foreground ones. Assume that T_n and T_s are thresholds. The three rules are:

- Rule 1: If $f_n < T_n$, then it is a noise region;
- Rule 2: If $f_s > T_s$, then it is a noise region;

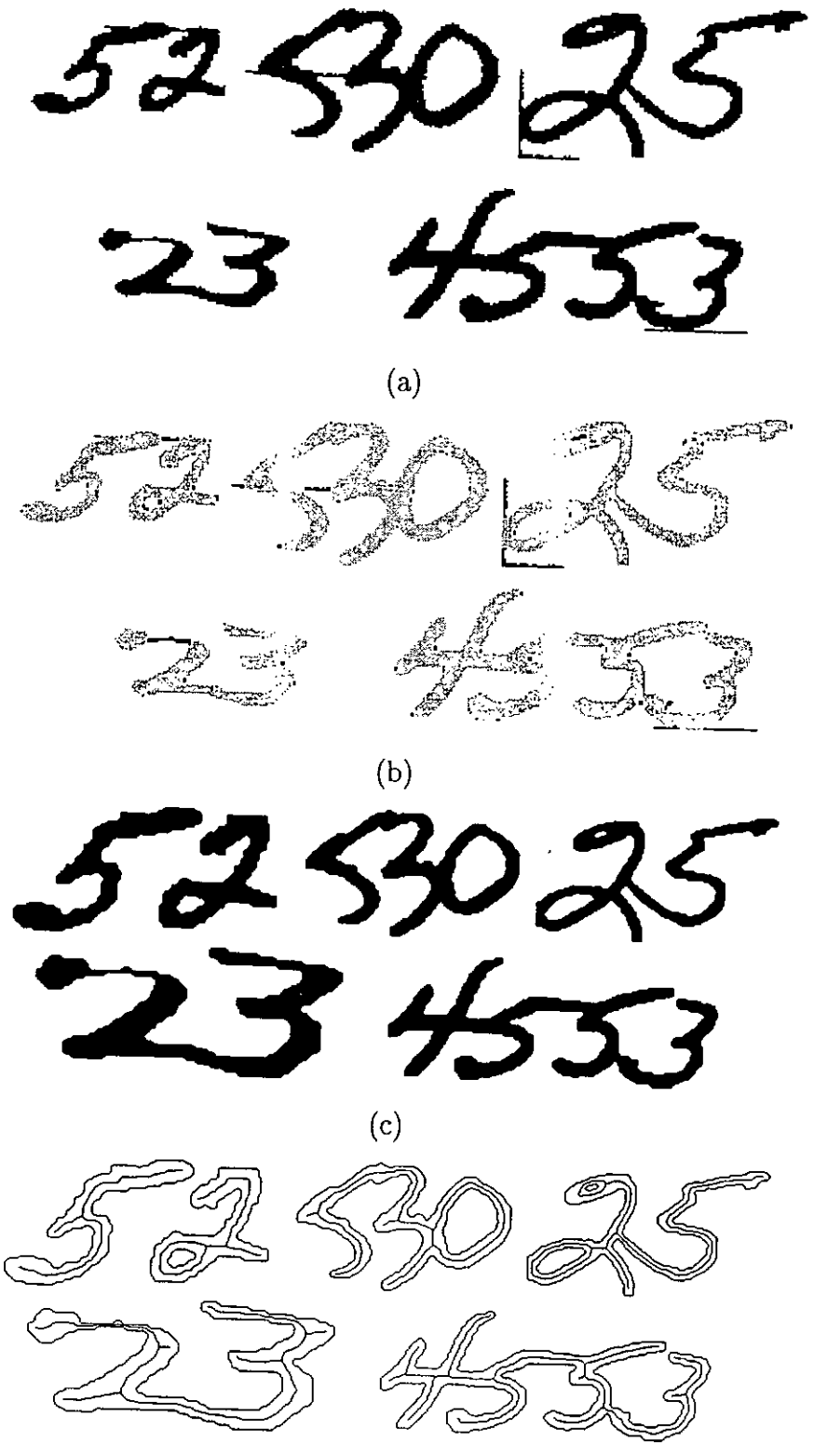


Figure 2.2: Preprocessing of binary digit string images: (a) original binary images; (b) the possible noise (dark regions) extracted from the original binary images; (c) the normalized digit strings after removing noise; (d) the skeleton images.

- Rule 3: If $f_p \leq 1$, it is a noise region; otherwise, it is a foreground region.

We associate a fuzzy membership function with each of the first two rules. For Rule 1, we adopt a membership function as shown in Fig. 2.1(a) where T_n is set to 10.

$$m(f_n) = \begin{cases} 1.0 & : \quad f_n \leq T_n - \alpha_n/2 \\ 0.0 & : \quad f_n \geq T_n + \alpha_n/2 \\ \frac{(T_n+\alpha_n/2)-f_n}{\alpha_n} & : \quad T_n - \alpha_n/2 < f_n < T_n + \alpha_n/2 \end{cases} \tag{2.2}$$

where α_n is an extension factor. For Rule 2, we use a membership function as shown in Fig. 2.1(b) where T_s is set to 1.3.

$$m(f_s) = \begin{cases} 1.0 & : \quad f_s \geq T_s + \alpha_s/2 \\ 0.0 & : \quad f_s \leq T_s - \alpha_s/2 \\ \frac{f_s-(T_s-\alpha_s/2)}{\alpha_s} & : \quad T_s - \alpha_s/2 < f_s < T_s + \alpha_s/2 \end{cases} \tag{2.3}$$

Again, α_s is an extension factor. For Rule 3, $m(f_p) = 1$ if $f_p \leq 1$; otherwise $m(f_p) = 0$.

Let P_n represent the degree that a region belongs to noise. P_n is given by:

$$P_n = \alpha m(f_n) + \beta m(f_s) + \gamma m(f_p) \tag{2.4}$$

where weights $\alpha = \beta = 0.1$ and $\gamma = 0.8$. If $P_n \geq 0.5$, the region is assigned as noise and can be removed.

2.2 Normalization

The second step of the preprocessing is size normalization that is very necessary because the original characters are usually in different sizes, as shown in Fig. 2.2(a). Generally, there are two size normalization schemes. One is to scale a foreground image to touch all four sides of the pre-defined box. The other scheme is to touch the top-bottom sides or the left-right sides only. The first

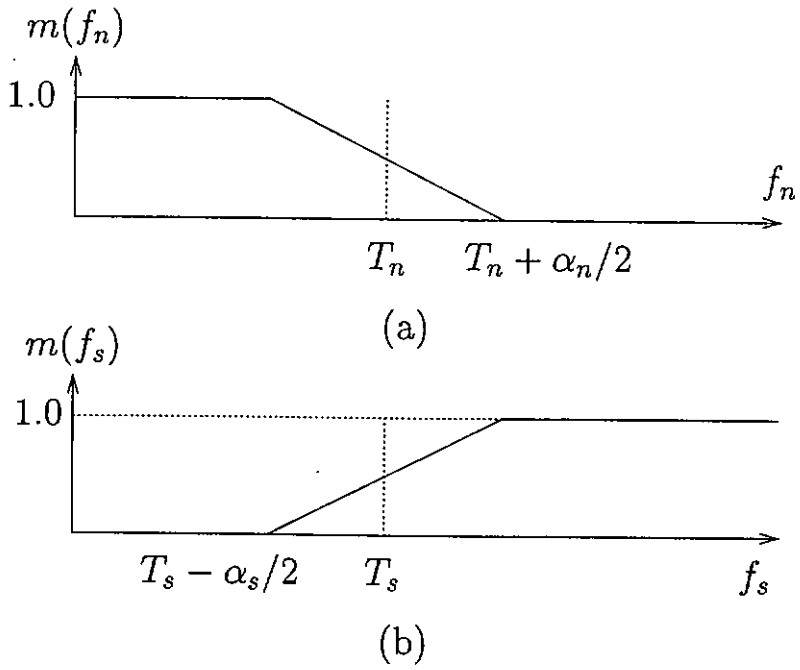
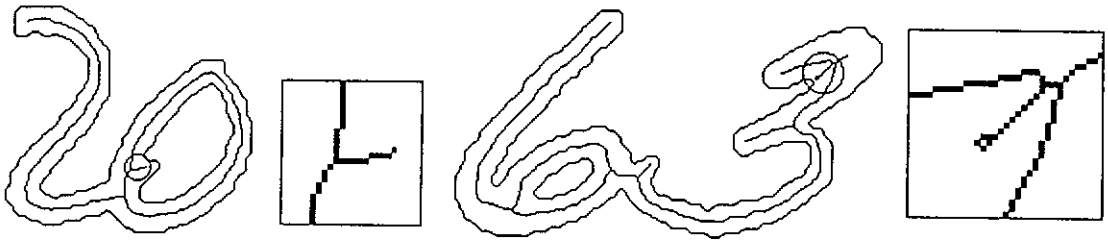


Figure 2.3: Two types of membership functions

normalization scheme is rarely used because it may produce deformation. In our study, the length of a digit string is unknown, so the second normalization scheme with a pre-defined height will do a better job. Fig. 2.2(c) shows the normalized noise-free images where all the foreground images are scaled to the same height.

2.3 Skeletonization

The last part in the preprocessing is skeletonization. The task here is to find a thinned representation of a foreground image. The skeleton image together with the normalized noise-free image is used to extract features for estimating the length of a digit string. The skeletonization algorithm is adapted from Hilditch's thinning algorithm [62], which depends on the "hit-or-miss" transform [62, 63]. The shortcoming of this algorithm is that it produces artifacts (buds) or extra branches during the processing. The extra branches are usually short in comparison with the other strokes. There are two kinds of artifacts, one from the poor writing in the form of a short line as shown in Fig. 2.4(a) and the other



(a) (b)

Figure 2.4: Artifacts generated during the Hilditch's thinning processing: (a) an artifact due to poor writing; (b) an artifact due to the hit-or-miss transform.

generated by the "hit-or-miss" transform in the form of a straight line with a small circle at the end and acute angles to the neighboring strokes as shown in Fig. 2.4(b).

Figure 2.2(d) shows the final skeleton images. We can see that not all artifacts should be removed. Some long artifacts are kept because they may indicate the sharp corner points of the strokes, which is useful for the later processing. The decision of removing an artifact or not is made to be based on its length and shape, and its angles with the neighboring branches.

2.4 Concluding Remarks

In this chapter, a three-step pre-processing scheme is presented for smoothing, normalizing and skeletonizing binary images of connected digit strings, which is required for later processing including length estimation and character segmentation. Mathematical morphology operations combined with fuzzy rules are used for removing noise. The noise-free image is then normalized to a pre-defined height with no change in the aspect ratio and skeletonized by a thinning algorithm.

Chapter 3

Length Estimation of Connected Digit Strings

Accurate length estimation is very helpful for the successful segmentation and recognition of connected digit strings, in particular, for an off-line recognition system. However, little work has been done in this area due to the difficulties involved. Only one paper proposed by Fenrich [64] give a length estimation method. It is an iterative segmentation-based algorithm to recognize strings of different length [64]. In this algorithm, vertical projection, upper and lower contours are used to find splitting positions. In each iteration, a high confidence character is recognized and removed from the image, till no characters are left. Note that there is a length estimation method mentioned in this algorithm, which is different from the problem we will tackle. The length estimation method, which is based on Constrained Linear Regression (CLR), use the image density as an independent variable and the number of characters recognized as a dependent variable. Since characters are removed as they are recognized, their effective contribution to the image density can be recorded. Recorded values are used to establish a least square linear model, which can be used to estimate the number of characters left in the image.

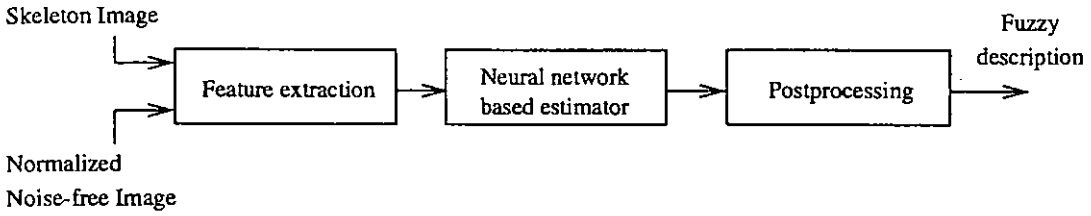


Figure 3.1: Block diagram of the length estimation system.

Obviously, the segmentation of character strings of unknown length introduced in section 1.1 can be improved significantly if the number of the characters can be estimated beforehand.

In this thesis, a new length estimation approach is presented. The kernel of our approach is a neural network estimator with a set of structure based features as the inputs. The outputs of the algorithm are a set of fuzzy membership grades reflecting the degrees of an input digit string for having different lengths.

As shown in Fig 3.1, our length estimation approach mainly includes four steps, preprocessing, feature extraction, neural network estimation, and postprocessing. A set of structure based features from the normalized noise-free image and its skeleton image are extracted in the step of feature extraction, which will be discussed in section 3.1. A multi-layer feedforward neural network is then trained to perform the length estimation based on the extracted features. Section 3.2 discusses data collection and the training of the neural network estimator. In Section 3.3, we explain how to derive fuzzy membership grades from the outputs of the neural network. Experimental results on NIST Special Database 3 and other artificially generated digit strings about this length estimation algorithm are reported in Section 3.4.

3.1 Feature Extraction

Feature extraction is an important step in achieving good performance of any pattern recognition system. The extracted features must be invariant to the possible distortions and variances of digit strings. Moreover, with a limited training

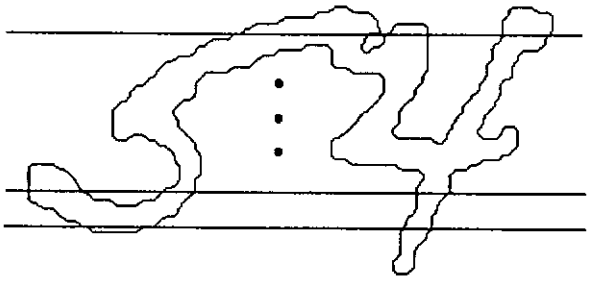


Figure 3.2: Horizontal transitions

set, the number of features must be kept reasonably small if the statistical classifier is to be used [11].

In this algorithm, seventeen structure-based features were extracted for training a multi-layer feedforward neural network to perform the length estimation. Among them, ten features are horizontal transitions extracted from the normalized noise-free image, six features come from feature points extracted from the skeleton image, and the aspect ratio of the image.

3.1.1 Horizontal Transitions

Horizontal transitions have been used for the segmentation of connected digit strings [34]. As shown in Fig. 3.2, the approach is to scan the image horizontally and to count the number of foreground-background and background-foreground transitions. The whole image can be divided into ten bands from the top to bottom, and the average number of horizontal transitions in each band is used as a feature (10 features in total).

3.1.2 Feature Points

Fork points and end points in the skeleton image are important feature points. As shown in Fig. 3.3, a fork point is a skeleton point that has more than two connected branches, and an end point is the skeleton point which has only one connected branch, that is, at the end of a stroke. We can see that some artifacts

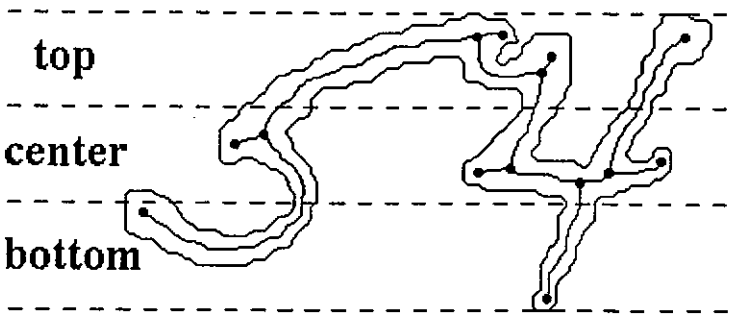


Figure 3.3: Feature points in a skeleton image.

are deliberately kept to indicate sharp turning corners. The whole image is partitioned into three bands as the top, center, and bottom ones. Two features, the number of the fork points and the number of end points, are extracted from each band. Therefore, in total six features are obtained.

3.2 Neural Network Based Length Estimation

As mentioned in section 1.3, we have 4,555 connected 2-digit strings, 355 connected 3-digit strings, 48 connected 4-digit strings, and 20,852 isolated digits. Obviously, the numbers of the connected 3-digit and 4-digit strings are too small as compared with those of the other two types. To solve the problem, we have generated some 3-digit and 4-digit strings by merging the existing samples together, as shown in Fig. 3.4 where a 2-digit string '53' is merged with an isolated digit '3' to make a connected 3-digit string '533'. The degree of overlapping is randomly set within a pre-defined range.

In our experiments, we chose 2,000 samples of isolated digits, 1,200 connected 2-digit strings, 1,200 connected 3-digit strings, and 1,200 connected 4-digit strings as the training set. For the testing, we have an independent set of 2,000 isolated digits, 3,355 connected 2-digit strings, 3,355 connected 3-digit strings, and 1,200 connected 4-digit strings. The strings of five or more connected digits is rare in real-world applications so they are not included in our research.

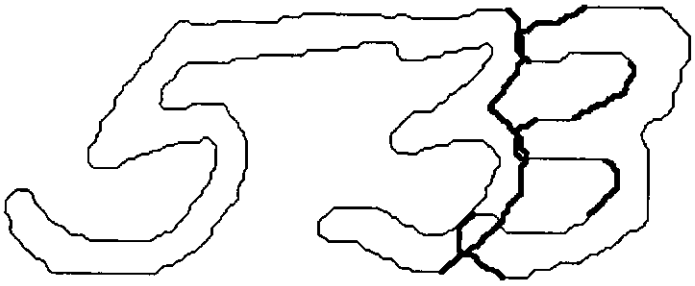


Figure 3.4: An artificial 3-digit string from merging a 2-digit string and an isolated digit.

An 18-60-4 three-layer feedforward neural network was trained to estimate the string length based on 17 features discussed in the previous section. Note that an extra input node with a fixed activity is used to produce bias terms to the hidden nodes through the corresponding weight connections. Each of the four outputs of the neural network is assigned to each of four classes, isolated digit, 2-digit string, 3-digit string, and 4-digit string. The neural network was trained using the back-propagation algorithm with the conjugate-gradient optimization technique.

3.3 Post-processing

The outputs of the neural network are a set of continuous values in $[0, 1]$. It will be more helpful for the later segmentation and recognition if the system can also indicate the confidence level at which the string belongs to each of four categories. Assume that the output vector of the neural network for the ith input digit string is $\mathbf{o}_i$. We compute the center of the output vectors of the digit strings that belong to the same class, that is,

$$\mathbf{c}_j = \frac{\sum_{i=1}^{N_j} \mathbf{o}_i}{N_j} \qquad j = 1, 2, ..., L \tag{3.1}$$

where L is the number of classes (four in our application) and N_j is the total

Table 3.1: The experimental results of the length estimation on the test set.

	isolated	2-digit	3-digit	4-digit	total
isolated	1973	27	0	0	2000
2-digit	19	3268	68	0	3355
3-digit	0	55	3085	215	3355
4-digit	0	0	70	1130	1200

number of digit strings belonging to class j. A set of fuzzy membership grades is derived for each digit string according to its distances to the class centers. Euclidean distance is applied here:

$$d_{ji} = \sqrt{(\mathbf{o}_i - \mathbf{c}_j)^T \times (\mathbf{o}_i - \mathbf{c}_j)} \tag{3.2}$$

where d_{ji} is the distance from output vector $\mathbf{o}_i$ to center $\mathbf{c}_j$. Finally, the membership grade of digit string i belonging to class j, m_{ji}, is given by:

$$m_{ji} = \frac{\frac{1}{d_{ji}}}{\sum_{l=1}^{L}(\frac{1}{d_{li}})} \tag{3.3}$$

3.4 Experimental Results and Discussion

Table 3.1 lists the classification results of the neural network on the test set. The overall correct classification rate is 95.3%. We can see from the table that the misclassification occurs among the neighboring classes only, for example, a connected 3-digit string might be misclassified into the 2-digit or 4-digit category.

Based on fuzzy membership grades, we introduce a parameter, λ, which is defined as the difference between the two maximum grades, that is,

$$\lambda_i = \max(m_{0i}, m_{1i}, m_{2i}, m_{3i}) - \text{second\_max}(m_{0i}, m_{1i}, m_{2i}, m_{3i}) \tag{3.4}$$

Table 3.2 summaries the classification of digit strings in terms of different λ values. Among the 9,456 correctly classified digit strings, 70% of them were well

classified ($\lambda \geq 0.5$). As to the 454 misclassified digit strings, about 88% of them were not totally misclassified, that is, the values of λ were smaller than 0.5 which means that an alternative choice may also be favorable. The length estimation is a part of our automatic digit recognition system and the results from this estimation only serves as a guideline for the later processing. If the value of λ is not large enough, say smaller than 0.5, both of the two highest outputs should be considered. As a result, among all these 9,910 testing samples, 6,838 (69%) samples were correctly classified with a high confidence level (one output with $\lambda \geq 0.5$), 3,017 (30.4%) digit strings had the correct estimation if the two best estimations were considered, and only 55 (0.6%) samples were poorly classified. The experimental results suggests that our approach of length estimation of digit strings is reliable and it will be very helpful for the recognition at the later stage.

Table 3.2: The analysis of the outputs with different λ values

λ	correct		wrong		all	
	number	%	number	%	number	%
$0 \sim 0.1$	234	2.47	167	36.8	401	4.05
$0.1 \sim 0.2$	352	3.72	100	22.0	452	4.56
$0.2 \sim 0.3$	411	4.35	70	15.4	481	4.85
$0.3 \sim 0.4$	648	6.85	37	8.15	685	6.91
$0.4 \sim 0.5$	973	10.3	25	5.51	998	10.1
$0.5 \sim 0.6$	1837	19.4	17	3.74	1854	18.7
$0.6 \sim 0.7$	1828	19.3	20	4.41	1848	18.6
$0.7 \sim 0.8$	2048	21.7	10	2.20	2058	20.8
$0.8 \sim 0.9$	862	9.12	5	1.10	867	8.75
$0.9 \sim 1.0$	263	2.78	3	0.66	266	2.68
total	9456	100.0	454	100.0	9910	100.0

3.5 Concluding Remarks

In this chapter, the length estimation of connected digit strings using a three-layer (one hidden) feedforward neural network with structure based features is presented. The approach consists of four steps, preprocessing, feature extraction,

neural network estimation, and post-processing. In the preprocessing, morphological operations are firstly applied to remove noise and artifacts in a digit string image, which is followed by the normalization and skeletonization. Two images, the normalized noise-free image and the skeleton image, are produced from the preprocessing. In feature extraction, seventeen structure based features are obtained from pre-processed images. A three-layer feedforward neural network is then trained to map the features to the target lengths of digit strings. In post-processing, the training results from the neural network are used to compute the centers of classes that are used to determine the fuzzy membership grades of a digit string having different numbers of digits. Experimental results on NIST Special Database 3 and other derived digit strings show that 9855 (99.4%) of a total of 9,910 digit strings are well estimated.

Chapter 4

Multi-Module Digit Classification

As mentioned in Chapter 1, the performance of the isolated character classifier is a key factor to decide the performance of the whole recognition system. In this chapter, we present a multi-module (MC) approach for handwritten digit recognition based on the human recognition experience in a hope of achieving human-like performance.

The structure of the classifier is shown in Fig. 4.1. The outputs of the classifier are 10 digit classes. The classifier has four modules: three neural network classifier with different sets of features and a template-based classifier. The outputs from the four modules are combined by a combinator, which is also a neural network. At the learning stage, each module is trained indiviually using its own feature set. At the recognition stage, all trained classifier work together.

Compared with a single large and complex classifier, the advantages of this multi-module structure are:

- Training is less complex because each module is designed to handle a specific sub-problem;

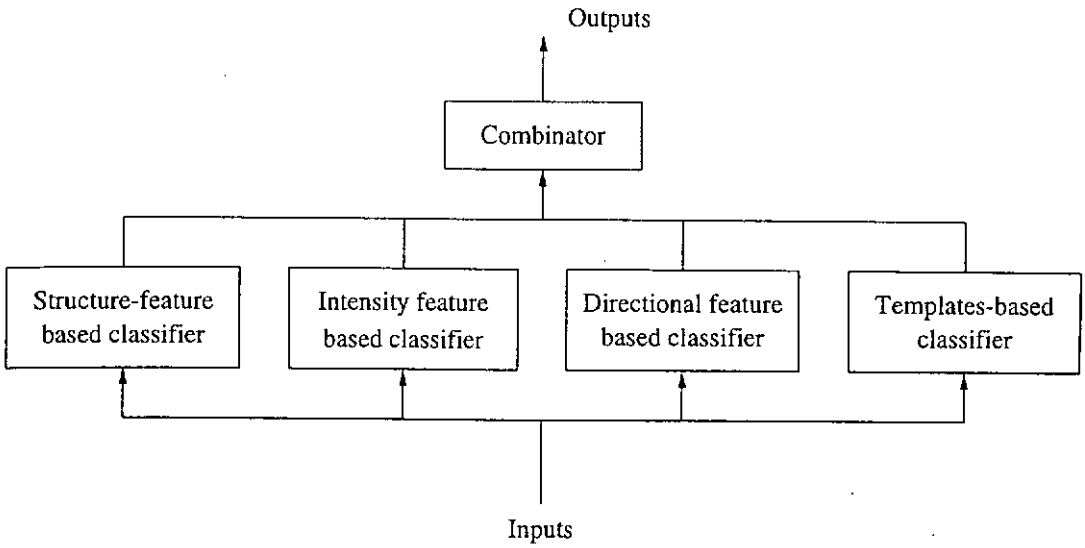


Figure 4.1: Block Diagram of the multi-module digit classification.

- it is expected that each module can tackle the specific problem more efficiently and accurately;
- because each module is trained independently, it is easy to add and delete modules, and only the new module(s) should be trained and the combinator retrained, which will save the cost in the training;
- this multi-module structure also makes the hardware implementation easier.

4.1 Neural Network-based Modules

As we can see from figure 4.1, we make use of three classifiers that use structural features, intensity features, and directional features as inputs, respectively. Each neural network classifier, which is individually trained by using its own training set, will mainly tackle one type of variation in handwritten digit patterns.

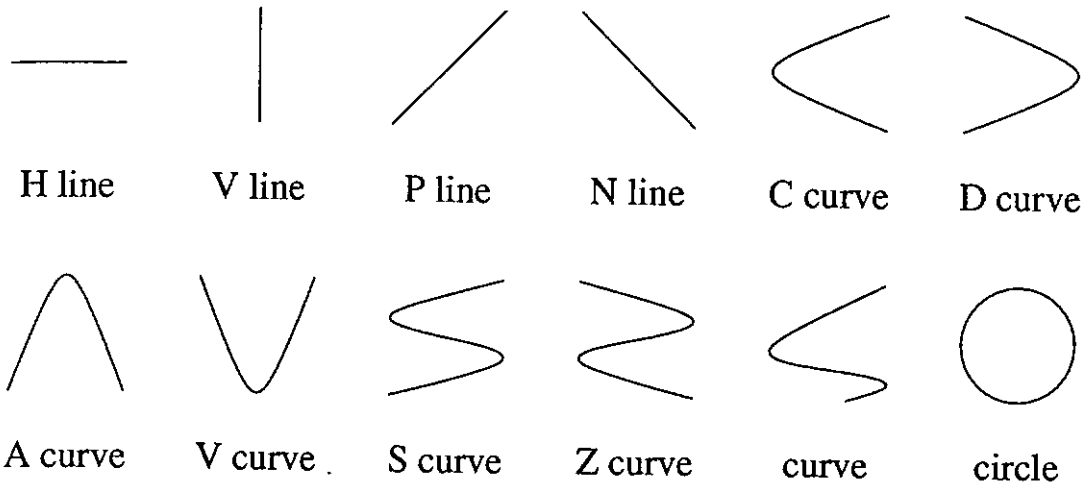


Figure 4.2: Twelve types of line segments.

4.1.1 Feature Extraction

4.1.1.1 Structural Features

Structural features are extracted from skeleton image of a binary image. In a skeleton image, a set of fork points (with more than two neighbors) and end points (with only one neighbor) are located, and segments between these points are traced. A set of symbolic, discrete-value, and continuous-valued features of segments are extracted to perform the classification.

A *branch* is a segment connecting a pair of adjacent nodes. A *circle* is a special branch connecting to a single node. We categorized each segments as one of 12 types shown in Fig. 4.2, *H line, V line, P line, N line, C curve, D curve, A curve, V curve, S curve, Z curve, curve, circle.* Type *curve* is reserved for a curve segment that cannot be fitted to any of the six curve types.

The measure of the straightness of a branch is determined by fitting a straight line using the least square error method. The straightness of a non-circular branch is defined as:

$$f_{SL} = \begin{cases} 1 - S/S_T & : \quad \text{if } S < S_T \\ 0 & : \quad \text{if } S \geq S_T \end{cases} \tag{4.1}$$

where S is the fitting error and S_T is a threshold. A branch is classified as a curve, if $0 \leq f_{SL} < 0.5$, or a straight line, if $0.5 \leq f_{SL} < 1$.

According to its angle with the horizontal direction, a straight line is classified into *H line, V line, P line, and N line.* A curve segment is categorized into one of six curve types based on its shape information. If a curve segment can not be assigned to any of these types, it is considered to be type *curve.*

Preliminary examination on the NIST Special Database 3 [65] showed that less than 0.5% of digit characters have more than six segments, so no more than six segments (in the order of decreasing lengths) are considered for each skeleton digit image. Four features were used to describe a segment [66]. They are:

1. The *type* of a segment.

2. The normalized length of a segment, l_i^n. Suppose that the width and height of the image (in pixels) are w and h, respectively, the number of segments is N_1 (could be greater than six), and the number of pixels in the i-th segment is N_{pi}. We have:

$$l_i = \sum_{p=1}^{N_{pi}-1} \text{step}(p, p+1) \tag{4.2}$$

where $\text{step}(p, q)$ is defined as:

$$\text{step}(p, q) = \begin{cases} 1 & \text{if } q \in N_4(p) \\ \sqrt{2} & \text{if } q \in N_8(p) \wedge \neg(q \in N_4(p)) \end{cases} \tag{4.3}$$

where p is a character pixel on the skeleton, and $N_4(p)$ and $N_8(p)$ are the 4-neighbors and 8-neighbors of p, respectively. The normalized l_i denoted by l_i^n, is defined as:

$$l_i^n = \frac{l_i}{2(w+h)} \tag{4.4}$$

3. x_{ci}^n and y_{ci}^n, the normalized horizontal and vertical coordinated of the relative center of the i-th segment to the center of the skeleton image, (x_g, y_g). First, the center of the i-th segment, (x_{ci}, y_{ci}), is calculated by:

$$x_{ci} = \frac{\sum_{p=1}^{N_{pi}-1} \text{step}(p, p+1)\frac{x_p + x_{p+1}}{2}}{l_i} \tag{4.5}$$

$$y_{ci} = \frac{\sum_{p=1}^{N_{pi}-1} \text{step}(p, p+1)\frac{y_p + y_{p+1}}{2}}{l_i} \tag{4.6}$$

Second, the center of the skeleton image, (x_g, y_g), is given by:

$$x_g = \frac{\sum_{i=1}^{N_l} l_i x_{ci}}{\sum_{i=1}^{N_l} l_i} \tag{4.7}$$

$$y_g = \frac{\sum_{i=1}^{N_l} l_i y_{ci}}{\sum_{i=1}^{N_l} l_i} \tag{4.8}$$

Finally, x_{ci}^n and y_{ci}^n are defined as

$$x_{ci}^n = \frac{x_{ci} - x_g}{w} \tag{4.9}$$

$$y_{ci}^n = \frac{y_{ci} - y_g}{h} \tag{4.10}$$

Besides the four features for each of six segments, which make 24 features, the number of segments that a digit has (set the maximum value to six) is used as a feature because some digits tend to have more segments than other. As a result, there are totally 25 structural features extracted for the module. Figure 4.1.1.1 shows an example of the segments of a skeleton image.

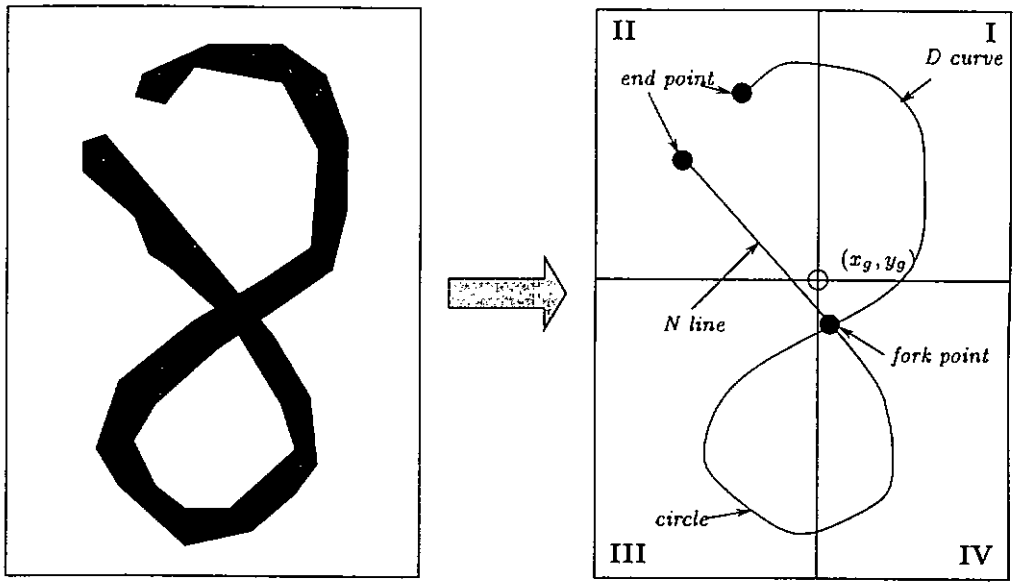


Figure 4.3: An example of thinning and segment representation.

In experiments, the input binary image are first size-normalized to 160×160 for feature extraction. The segments shorter than 10 pixels are ignored although they may be useful in the classification to a certain degree.

4.1.1.2 Directional Features

Kirsch masks, as shown in Fig. 4.4, have been adopted for extracting directional features. Kirsch defined a nonlinear edge enhancement algorithm as follow [63]:

$$G(i,j) = \max\{1, \max_{k=0}^{7}[|5S_k - 3T_k|]\} \tag{4.11}$$

where

$$S_k = A_k + A_{k+1} + A_{k+2} \tag{4.12}$$

$$T_k = A_{k+3} + A_{k+4} + A_{k+5} + A_{k+6} + A_{k+7} \tag{4.13}$$

$G(i,j)$ is the gradient of pixel (i,j), the subscripts of A are evaluated modulo 8, and A_k, $(k = 0, 1, \cdots, 7)$ are neighbors of pixel (i,j), as shown in Fig. 4.1.1.2.

In this module, an input pattern is size-normalized by 160×160 and the directional feature vectors for horizontal (H), vertical (V), right-diagonal (R), and left-diagonal (L) directions are computed as fellows:

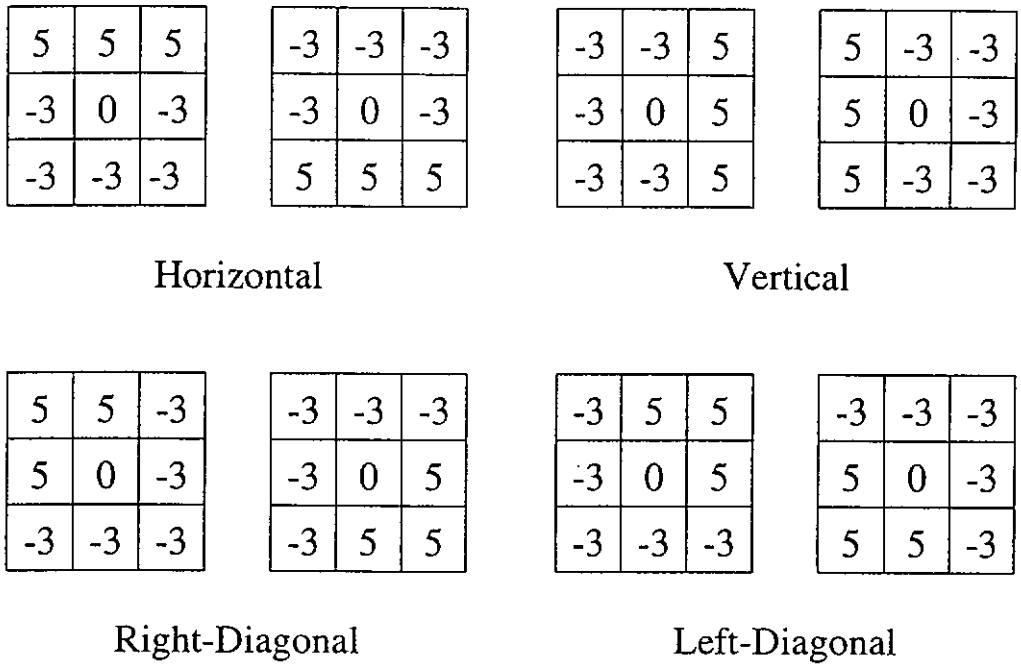


Figure 4.4: Four directional Kirsch masks.

A_0	A_1	A_2
A_7	(i,j)	A_3
A_6	A_5	A_4

Figure 4.5: Definition of eight neighbors $A_k, (k = 0, 1, \cdots, 7)$ of pixel (i, j).

$$
\begin{aligned}
G(i,j)_H &= \max(|5S_0 - 3T_0|, |5S_4 - 3T_4|) \\
G(i,j)_V &= \max(|5S_2 - 3T_2|, |5S_6 - 3T_6|) \\
G(i,j)_R &= \max(|5S_1 - 3T_1|, |5S_5 - 3T_5|) \\
G(i,j)_L &= \max(|5S_3 - 3T_3|, |5S_7 - 3T_7|)
\end{aligned} \tag{4.14}
$$

Each 160 × 160 directional feature vectors is compressed into 4 × 4 feature vector. A fixed normalization factor of (1/200) is used to normalize all the features to be within a suitable range for a neural network. As a result, 64 directional features are extracted.

4.1.1.3 Intensity-based Features

The intensity-based feature extraction is to partition a digit image into 8 × 8 regions from 160 × 160 normalized image. The numbers of digit pixels in each region is counted and used as a feature. A total of 64 intensity-based features are extracted. There are 20 × 20 pixels in each region so the range of feature values is within $[0, 400]$. A fixed normalization factor of (1/200) is used to normalize all the features to be within [0, 2].

Figure 4.6 illustrates different types of features extracted.

4.1.2 Training of Neural Networks

Three neural network classifiers and the combinator are trained using the back-propagation algorithm with the conjugate-gradient optimization technique. 10,426 isolated digits from NIST Special Database 3 are used as the training set, and another independent 10,426 isolated digits are used as test set. The structural feature based module uses a 25-36-10 Multi-Layer Perceptron (MLP) and 25 structural features as inputs. Note that a bias input is used for each neural network. Directional feature based module and intensity feature based module

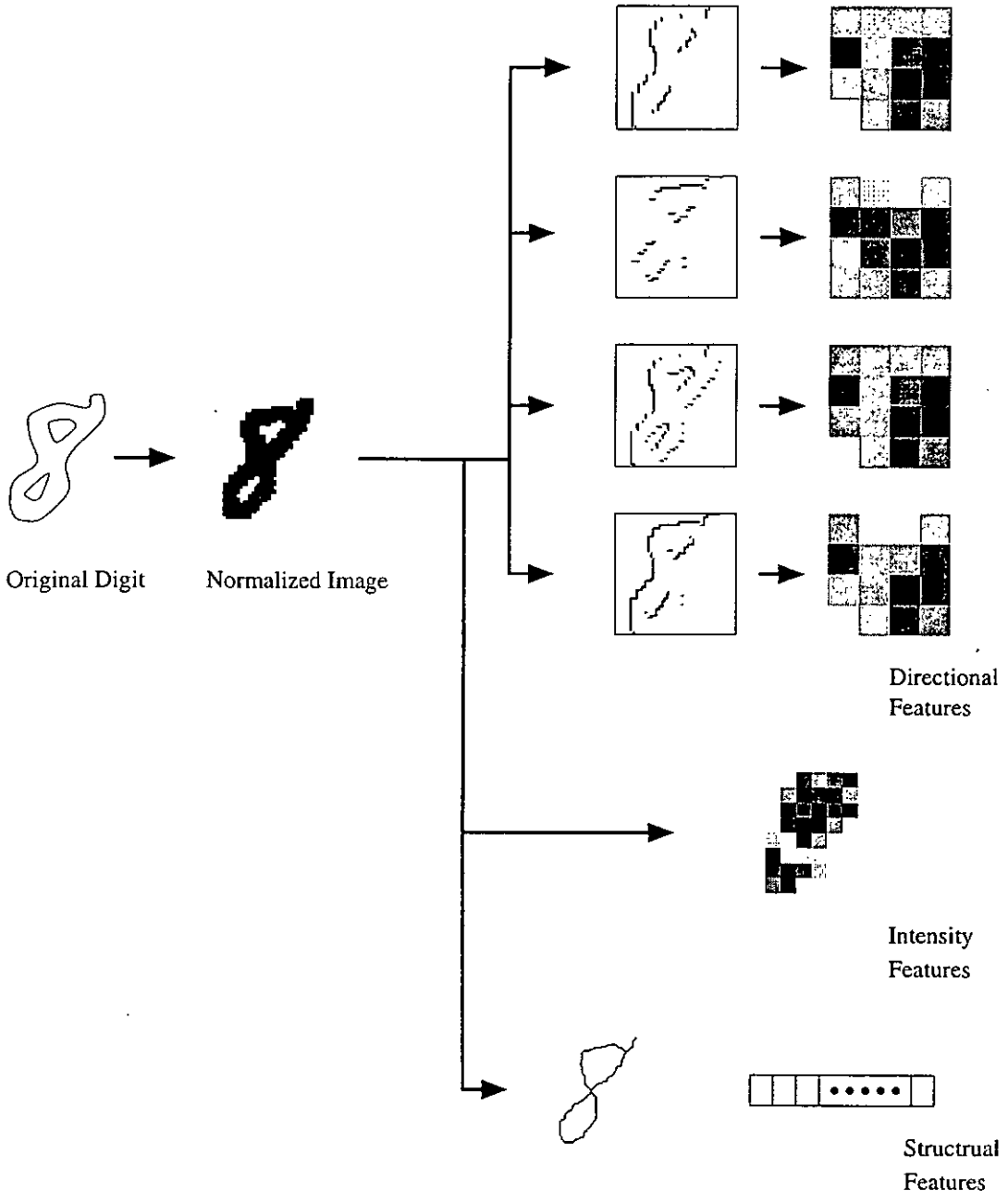


Figure 4.6: Different types of features extracted.

use 64-45-10 and 64-36-10 MLPs, respectively. The 30 outputs from three modules and the 10 outputs from the template-matching based classifier are used as the inputs of the combinator. The combinator, a 40-31-10 MLP, was trained independently using the same training set. If new modules are added, only the added modules need to be trained and the combinator needs to be re-trained.

4.2 Template-Based Classifier

In this section, we present a new template presentation scheme, a neural network approach for extracting templates, and an evolutionary algorithm for optimizing templates. Each template presented by a Pixel-to-Boundary Distance Map (PBDM) is approximated by a rational B-spline surface with a set of control knots. In template extraction, a cost function of the amplitude and gradient of the PBDM is minimized by using a neural network. The templates are then optimized by an evolutionary algorithm. While in the matching step, a similarity measure that takes into account of both the amplitude and gradient of the PBDM is adopted to match an input pattern to templates.

4.2.1 Representation of Templates

In our approach, the templates are represented by a rational B-spline surface with a set of control knots. Curve and surface representation and manipulation using the B-spline form (non-rational or rational) are commonly used in geometric design. In 1975, Versprille proposed a rational B-splines for geometric design of curves and surfaces [66]. The work outlined the important properties of rational B-splines, such as continuity, local control property, etc. Recently, B-spline curve (snake) have been used in the description of objects [67, 68], as well as digit images [18, 19]. With the B-spline estimation, information concerning the shape of desired objects can be incorporated into the control parameters of the curve or surface based templates.

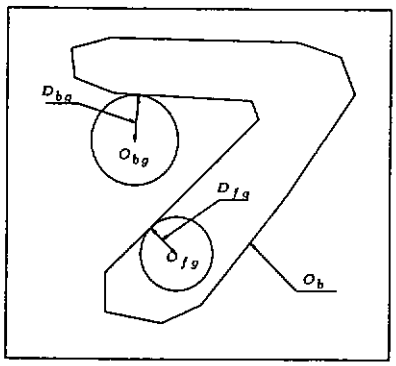


Figure 4.7: Pixel-to-boundary distance.

However, the estimation of B-spline curve involves a lot of uncertainties and is prune to failure because the corresponding parameters of templates and the input data should reflect the information in the same aspect of the image. In our approach, we use B-spline surface to approximate the PBDM of the digit image. The number of the control points of surface and the locations of them are prescribed, so for each control point, the control area is certain. Therefore, the problem can be avoided.

A binary digit image can be converted to a Pixel-Boundary Distance Map (PBDM). The distance of pixel (x, y) (we here use (x, y) instead of normally discrete (i, j) for its consistency with the definition of the rational B-splice surface) to the nearest foreground/background boundary is measured, as shown in Figure 4.2.1. For a foreground pixel, we have $d(x, y) \geq 0$. But for a background pixel, we assume it has a negative value, that is $d(x, y) < 0$.

Let $g(x, y)$ denote the value of pixel (x, y) in the PBDM. It is defined as

$$g(x, y) = \frac{e}{2} e^{-\frac{[(d(x,y)-d_{max})]^2}{d_{max}^2}} \tag{4.15}$$

where d_{max} is the maximum distance and $(e/2)$ is a constant to make a point on the boundary, O_b, have $g(x_{O_b}, y_{O_b}) = 0.5$.

Assume the rational B-spline surfaces for the templates are $\{S(P_j), 0 \leq j \leq N_h\}$, where N_h is the number of templates and P_j is a matrix of the control

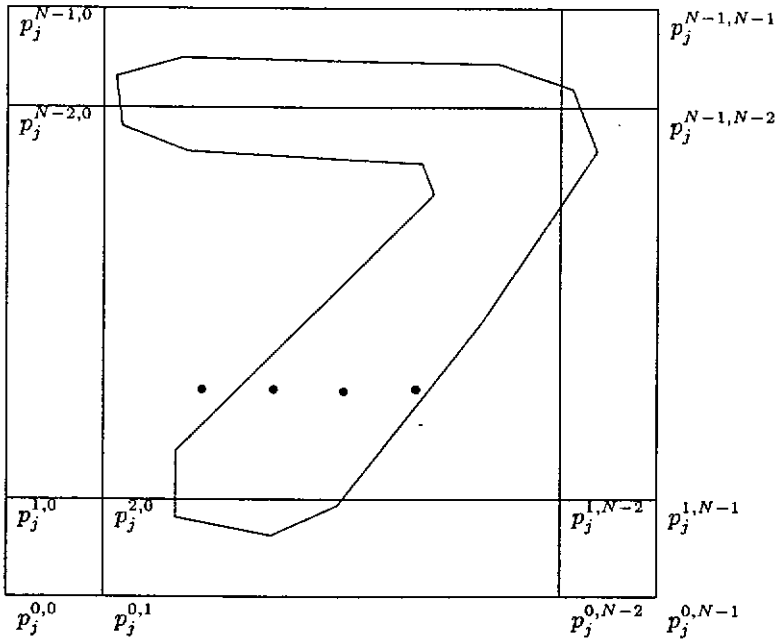


Figure 4.8: Control points of a template.

points of template j, as shown in Figure 4.2.1,

$$P_j = \begin{pmatrix} p_j^{0,0} & p_j^{0,1} & \cdots & p_j^{0,N-1} \\ p_j^{1,0} & p_j^{1,1} & \cdots & p_j^{1,N-1} \\ \cdots & \cdots & \cdots & \cdots \\ p_j^{N-1,0} & p_j^{N-1,1} & \cdots & p_j^{N-1,N-1} \end{pmatrix}_{N \times N} \tag{4.16}$$

A rational B-spline surface can be defined as

$$S^j(x,y) = B_{r_1}^t(x) P_j B_{r_2}(y) \tag{4.17}$$

where r_1 and r_2 are the orders of the B-spline bases. $B_r(x)$ is a base function vector and $B_r^t(x)$ is its transpose. Normally, we have $r_1 = r_2 = r$. $B_r(x)$ is a vector of base functions given by

$$B_r(x) = \begin{pmatrix} B_{0,r}(x) \\ B_{2,r}(x) \\ \cdots \\ B_{N-1,r}(x) \end{pmatrix}_{N \times 1} \tag{4.18}$$

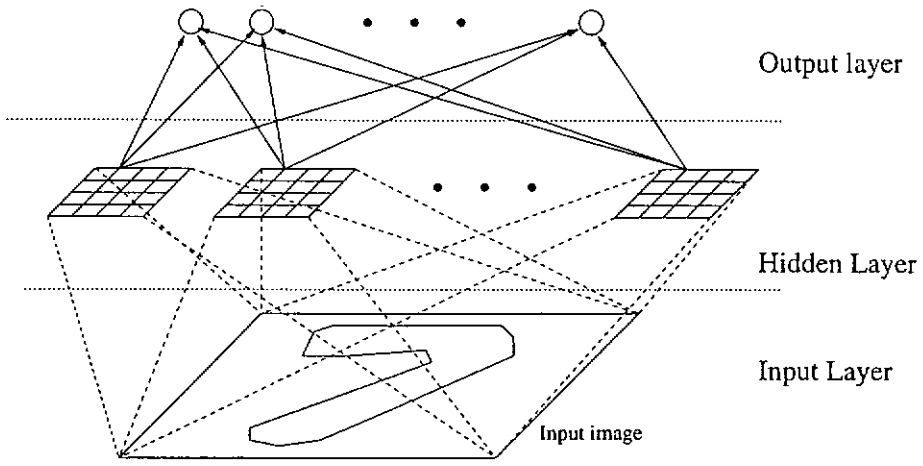


Figure 4.9: The template extraction neural network.

where $B_{i,1}(u)$ is given by

$$B_{i,1}(u) = \begin{cases} 1 & u_i \leq u < u_{i+1} \\ 0 & \text{otherwise} \end{cases} \tag{4.19}$$

and

$$B_{i,r}(u) = \frac{u - u_i}{u_{i+r-1} - u_i} B_{i,r-1}(u) + \frac{u_{i+r} - u}{u_{i+r} - u_{i+1}} B_{i+1,r-1}(u) \tag{4.20}$$

where $U = \{u_i | 0 \leq i \leq N_u\}$ is a knot vector of length N_u. It is assumed that $0/0 = 0$. The value of u_i determines the locations of control knots and should be pre-defined. To simplify to the B-spline surface, we define a knot vector as $U = \{\overbrace{0, \cdots, 0}^{r}, \frac{1}{(N-3)}, \frac{2}{(N-3)}, \cdots, \overbrace{1, \cdots, 1}^{r}\}$, so $N_u = N + 2 \times (r - 2)$. In most applications, the number of input pattern N_i is much greater than the number of control points in a templates, $N \times N$.

4.2.2 Template Extraction

Similar to an approach for optimizing digit prototypes proposed by Yan [15], initial templates are extracted using a multi-layer neural network as shown in Figure 4.9. The input to the neural network is a PBDM G. The hidden layer contains the templates to be extracted. Each hidden node corresponds to a template. $Z = [z_0, z_1, \cdots, z_{N_o}]$ is the output of the network, corresponding to N_o

output classes. The output of the hidden node is given by

$$\phi_j = f(G, P_j) = \gamma_1 \phi_{1,j} + \gamma_2 \phi_{2,j} \tag{4.21}$$

where $\gamma_1 + \gamma_2 = 1$ and ϕ_j is a function that measure the similarity between the input PBDM G and template P_j, considering not only the magnitude of but also the gradient at a point. $\phi_{1,j}$ is defined as $\phi_{1,j} = \frac{2}{1+e^{C_s v_j}}$, where C_s is a smoothing factor and

$$\begin{aligned} v_j &= \frac{\int_0^1 \int_0^1 [S_j(x,y) - g(x,y)]^2 \, dx \, dy}{\int_0^1 \int_0^1 g^2(x,y) \, dx \, dy} \\ &= \frac{\int_0^1 \int_0^1 [B_r'(x) P_j B_r(y) - g(x,y)]^2 \, dx \, dy}{\int_0^1 \int_0^1 g^2(x,y) \, dx \, dy} \end{aligned} \tag{4.22}$$

We defined $\phi_{2,j}$ as

$$\phi_{2,j} = \int_0^1 \int_0^1 \cos^2 \left(\beta_j(x,y)\right) \, dx \, dy \tag{4.23}$$

where $\beta_j(x,y)$ is the angle between the gradient of template $S^j(x,y)$ at (x,y) and the gradient of the input PBDM G at the same point. It is easy to verify that $0 \leq \phi_{1,j} \leq 1$ and $0 \leq \phi_{2,j} \leq 1$. If $\phi_{1,j} = 1$, then $S^k(x,y) \equiv g(x,y)$.

The connection weight from hidden node j to output node m is denoted w_{jm}. The output of the mth output node is given by

$$z_m = \sum_{j=1}^{N_h} w_{jm} \phi_j \tag{4.24}$$

where N_h is the number of templates. Weights $\{w_{jm}\}$ must be pre-defined.

Assume that the desired and actual activities of the output node m are z_m^t and z_m^o respectively for input G. For each j, we need to find p_j^{kl} so that the following energy function is minimized:

$$E = \frac{1}{2} \sum_{m=1}^{N_o} |z_{m0} - z_m|^2 \tag{4.25}$$

The generalized delta rule is adopted to train the feedforward neural network for extracting a set of templates. According to the generalized delta rule, p_j^{jk} can be learned by successively applying an increment given by

$$\Delta p_j^{kl} = -\alpha \frac{\partial E}{\partial p_{kl}}$$

$$
\begin{aligned}
&= -\alpha \frac{\partial E}{\partial z_m} \frac{\partial z_m}{\partial \phi_j} \frac{\partial \phi_j}{\partial p_{kl}} \\
&= \alpha \sum_{m=1}^{N_o} (z_{m0} - z_m) w_{jm} \frac{\partial \phi_j}{\partial p_{kl}}
\end{aligned} \tag{4.26}
$$

where α is a learning factor.

When an input image is given, its PNDM G can be obtained, so are $\frac{\partial g}{\partial x}$, $\frac{\partial g}{\partial x}$ and $g_{sum}^2 = \int_0^1 \int_0^1 g^2(x,y)\,dx\,dy$, therefore we have

$$
v_j = \frac{1}{g_{sum}^2} \int_0^1 \int_0^1 [B_r'(x) P_j B_r(y) - g(x,y)]^2 \,dx\,dy \tag{4.27}
$$

$$
\phi_{2,j} = \int_0^1 \int_0^1 \frac{\left(\frac{\partial S_j}{\partial x} g_y' + \frac{\partial S_j}{\partial y} g_x'\right)^2}{\left(\left(\frac{\partial S_j}{\partial x}\right)^2 + \left(\frac{\partial S_j}{\partial y}\right)^2\right)\left(g_x'^2 + g_y'^2\right)} \,dx\,dy \tag{4.28}
$$

Considering the differentiation property of the B-spline [70],

$$
\begin{aligned}
\frac{\partial S_j(x,y)}{\partial x} &= \frac{\partial B_r^t(x)}{\partial x} P_j B_r(y) \\
&= r B_{r-1}^t(x) P_j^x B_r(y) \\
&= r S_{j\,(r-1,r)}^x(x,y)
\end{aligned} \tag{4.29}
$$

similarly,

$$
\begin{aligned}
\frac{\partial S_j(x,y)}{\partial y} &= B_r^t(x) P_j \frac{\partial B_r(y)}{\partial y} \\
&= r B_r^t(x) P_j^y B_{r-1}(y) \\
&= r S_{j\,(r,r-1)}^y(x,y)
\end{aligned} \tag{4.30}
$$

where

$$
B_{r-1}(x) = \begin{pmatrix} B_{0,r-1}(x) \\ B_{2,r-1}(x) \\ \cdots \\ B_{N,r-1}(x) \end{pmatrix}_{(N+1)\times 1} \tag{4.31}
$$

$$P_x^j = \begin{pmatrix} \frac{p^k_{0,0}}{u_r-u_0} & \frac{p^k_{0,1}}{u_r-u_0} & \cdots & \frac{p^k_{0,N-1}}{u_r-u_0} \\ \frac{p^k_{1,0}-p^k_{0,0}}{u_{r+1}-u_1} & \frac{p^k_{1,1}-p^k_{0,1}}{u_{r+1}-u_1} & \cdots & \frac{p^k_{1,N-1}-p^k_{0,N-1}}{u_{r+1}-u_1} \\ \cdots & \cdots & \cdots & \cdots \\ -\frac{p^k_{N-1,0}}{u_{r+N}-u_N} & -\frac{p^k_{N-1,1}}{u_{r+N}-u_N} & \cdots & -\frac{p^k_{N-1,N-1}}{u_{r+N}-u_N} \end{pmatrix}_{(N+1)\times N} \tag{4.32}$$

$$P_y^j = \begin{pmatrix} \frac{p^k_{0,0}}{u_r-u_0} & \frac{p^k_{0,1}-p^k_{0,0}}{u_{r+1}-u_1} & \cdots & -\frac{p^k_{0,N-1}}{u_{r+N}-u_N} \\ \frac{p^k_{1,0}}{u_r-u_0} & \frac{p^k_{1,1}-p^k_{1,0}}{u_{r+1}-u_1} & \cdots & -\frac{p^k_{1,N-1}}{u_{r+N}-u_N} \\ \cdots & \cdots & \cdots & \cdots \\ \frac{p^k_{N-1,0}}{u_r-u_0} & \frac{p^k_{N-1,1}-p^k_{N-1,0}}{u_{r+1}-u_1} & \cdots & -\frac{p^k_{N-1,N-1}}{u_{r+N}-u_N} \end{pmatrix}_{N\times(N+1)} \tag{4.33}$$

The gradients of templates are also rational B-spline surfaces. Therefore, we have

$$\frac{\partial \phi_j}{\partial p^j_{kl}} = \gamma_1 \frac{\partial \phi^1_j}{\partial p^j_{kl}} + \gamma_1 \frac{\partial \phi^2_j}{\partial p^j_{kl}} \tag{4.34}$$

$$\begin{aligned} \frac{\partial \phi^1_j}{\partial p^j_{kl}} &= \frac{2C_s v_j(v_j-1)}{g^2_{sum}} \\ &\quad \oint_{e_{kl}} (S_j(x,y) - g(x,y)) \\ &\quad (B_{i,r}(x) B_{j,r}(y))\, dx\, dy \end{aligned} \tag{4.35}$$

$$\begin{aligned} \frac{\partial \phi^2_j}{\partial p^j_{kl}} &= \oint_{e_{kl}} \frac{1}{{g'_x}^2 + {g'_y}^2} \\ &\quad \frac{H(x,y)}{\left[\left(S^x_{j\,(r-1,r)}\right)^2 + \left(S^y_{j\,(r,r-1)}\right)^2\right]^2}\, dx\, dy \end{aligned} \tag{4.36}$$

where e_{kl} is the effective control region of p_{kl} and

$$\begin{aligned} H(x,y) &= 2\Big\{\Big[(g'_y)^2 - (g'_x)^2\Big] S^x_{j\,(r-1,r)} S^y_{j\,(r,r-1)} + g'_x g'_y \Big[\left(S^y_{j\,(r,r-1)}\right)^2 - \left(S^x_{j\,(r-1,r)}\right)^2\Big]\Big\} \\ &\quad \Big\{S^y_{j\,(r,r-1)} B_{l,r}(y) \Big[\frac{B_{k,r-1}(x)}{(u_{r+k}-u_k)} - \frac{B_{k+1,r-1}(x)}{(u_{r+k+1}-u_{k+1})}\Big] - \\ &\quad S^x_{j\,(r-1,r)} B_{k,r}(x) \Big[\frac{B_{l,r-1}(y)}{(u_{r+l}-u_l)} - \frac{B_{l+1,r-1}(y)}{(u_{r+l+1}-u_{l+1})}\Big]\Big\} \end{aligned} \tag{4.37}$$

In order to simplify the extraction algorithm, the clustering algorithm is utilized. At iteration t of the templates extraction procedure, as to an input PBDM, instead of calculate all templates, only the templates with maximum ϕ_j are updated. So, w_{jm} can be defined as:

$$w_{jm} = \begin{cases} 1 & : \quad j = \arg\max(\phi_i) \quad i \in S_m \\ 0 & : \quad \text{otherwise} \end{cases} \tag{4.38}$$

where S_m is the set in which each index corresponds to a hidden node that represents a prototype that belongs to the class represented by output node m, and m is the class that the input PBDM belongs to. So, Δp_{kl}^j can be given by

$$\begin{aligned} \Delta p_{kl}^j &= \alpha(z_{m0} - z_m)\frac{\partial \phi_j}{\partial p_{kl}^j} \\ &= \alpha(z_{m0} - z_m)(\gamma_1 \frac{\partial \phi_j^1}{\partial p_{kl}^j} + \gamma_2 \frac{\partial \phi_j^2}{\partial p_{kl}^j}) \end{aligned} \tag{4.39}$$

4.2.3 Template Optimization Using Evolutionary Algorithms

Evolutionary algorithms for optimization have been studied for more than three decades since Fogel *et al* published the first work on evolutionary simulation. Many years of research and application have demonstrated that evolutionary algorithms, which emulate the search process of natural evolution, are powerful for the global optimization. Current Evolutionary Algorithms (EAs) include Evolutionary Programming (GP), Evolutionary Strategies (ES), Genetic Algorithm (GA) and Genetic Programming (GP) [71]. Bäck *et al* has done a comparative study of the first three approaches [72]. An good introduction of evolutionary techniques on optimization is given by Fogel [73].

Several applications of evolutionary template optimization have been reported in [74, 75, 76, 77]. However, in these algorithms, only one best template is extracted from one set of training samples. Obviously, for most object recognition

problems, a single template for a class of object is not enough for a reliable recognition system. More templates have to be extracted to achieve a better performance. Sarkar [78] presented a *fitness* based clustering algorithm, which can be utilized for template optimization. In the algorithm, one opponent (parent or offsprings) contains variable number of clustering centers. In its selection procedure, the fitness values of parents and offsprings are compared and a number of opponents are selected arbitrarily as the reference opponents. The opponent with better performance to each reference opponent can receive a *win*. Based on the *wins*, some opponents are selected as the parents of next generation. The shortcoming of the algorithm is that the computation requirement is in an exponential relationship with the cluster number. If the number of cluster number is very large, the algorithm ceases to be feasible.

Unlike Sarkar [78] approach of using a set of cluster centers as a component in evolution and selecting only one set as the winner, we use templates as the components directly and the selected survivors are a group of templates. The evolutionary processing is to emulate the evolution of a nature social system, such as ants, bees and human society.

The study of social system, as a part of sociobiology, began in the middle of 19th century, after Charles Darwin published his most famous book, *On the Origin of Species by Means of Natural Selection* in 1859. Sociobiology seeks to extend the concept of natural selection to social systems and social behavior of animals, including humans.

In our algorithm, we define a small, simple and homogeneous social system. Each template is just like an individual in it, and the society can only accommodate a limited number of individuals, but the individuals can generate a larger number of offsprings in each generation. Therefore, only the offsprings of better ability will be kept and the others must be discarded, that is,"survival of the fittest". Moreover, the relationship between individuals can be not only competitive (only winners can survive) but also cooperative (the properties of individuals

are reflected by their performance collectively). After a number of generations of evolution, a group of templates are selected and they as the whole can achieve a good recognition performance. The algorithm can be divided into the following steps:

1. A population of N_P templates (called parents) is initially generated. We use the templates extracted by the multi-layer neural network presented in Section 4.2.2.

2. N_O offsprings are generated from the parents by evolutionary operations (discussed in Section 4.2.3.1).

3. A subset of N_P offsprings are selected as the parents of the next generation based on a fitness measure (discussed in Section 4.2.3.2).

4. If the number of generations is less than the pre-set number or the performance of the selected templates is not satisfactory, go to Step 2.

In our algorithm, the templates in each numeral class are optimized independently. The parallel processing nature of our proposed method allows a fast processing on parallel computers or multiple computers.

4.2.3.1 Generation of Offsprings

For better explanation of the offspring generation and selection procedure, we define:

S_{tr} the training set;

S_{tr}^i the ith training sample;

P the parent set;

P^k the kth component in set P;

O the offsprings set;

O^k the kth component in set O;

P' the selected subset of O as the parent set for the next generation;

$o_k^{x,y}$ the control parameter at location (x, y) of O^k;

$p_k^{x,y}$ the control parameter at location (x, y) of P^k.

Three ways to generate offsprings are replication, mutation and recombination. Replication is simply a copy operation, $O^i = P^i$, where O^i and P^i are an offspring and a parent template respectively. The number of replicated offsprings is $N_O^r = N_P$, where N_P is the number of parents.

Mutation is an operation by which perturbations are added to the parents. In our algorithm, the perturbation Q is Gaussian noise, that is, $O^i = P^j + Q$. The number of mutated offsprings is $N_O^m = nN_P$, where n is a positive integer.

The recombination mechanism used in evolutionary algorithms is either in their usual form, producing a new individual from two randomly selected parents, or in their global form, allowing of taking components for a new individual from potentially all individuals available in the parent population [72]. In our algorithm, the components of recombined offsprings are selected randomly from the two randomly selected parents, Gaussian noise is added as perturbations. Note that whether adding perturbation or not is controlled by a random binary variable. The recombination is given by

$$o_i^{x,y} = r \times p_{j_1}^{x,y} + (1 - r) \times p_{j_2}^{x,y} + r_p \times q^{x,y} \tag{4.40}$$

where $o_i^{x,y}$ is the element of o_i at (x, y), $p_j^{x,y}$ the element of P_j at (x, y), and $q^{x,y}$ the noise level at (x, y). Random variable r takes a random value within $[0, 1]$ and r_p is randomly set to 1 or 0 to control whether perturbation is added or

not. Assume the number of recombined templates is N_O^c. The total number of offsprings from three evolutionary operations is $N_O = N_O^r + N_O^m + N_O^c$.

4.2.3.2 Selection Procedure

The task of the selection procedure is to select a subset of N_P templates P' from N_O offsprings generated. Based on the similarity function ϕ_j^k defined in Eq. 4.21, we define the fitness of P' as

$$\text{fitness}(P') = \sum_{j=0}^{N_t} \max_{k \in P'}(\phi_j^k) \tag{4.41}$$

where N_t is the number of training samples. The fitness(P') of the selected subset should be greater than the fitness of any other subset of N_P templates from the offspring set O. The number of possible subsets of N_P templates from O is $C_{N_O}^{N_P}$, which is an extremely large number if we set $N_P = 100$ and $N_O = 1000$. In order to make this algorithm practical, a fast selection procedure based directly on the function ϕ_j^i is adopted. The computing time of our fast selection procedure is proportional to the number of templates, which is an improvement compared with the Sarkar's approach [78] in which the computational requirement is in an exponential relationship with the cluster number.

The fast selection procedure can be divided into following steps:

1. For each training sample S_{tr}^i, a number of N_{top} templates with top function ϕ outputs are recorded and sorted in the decreasing order. The others are discarded. We let $\{T^{i,j},\ \ i = 1, 2, \cdots, N_t,\ \ j = 1, 2, \cdots, N_{top}\}$ be the set of the recorded output values and $\{T_{index}^{i,j},\ \ i = 1, 2, \cdots, N_t,\ \ j = 1, 2, \cdots, N_{top}\}$ be the index from $T^{i,j}$ to the templates, so the relation between them can be presented as:

$$\begin{aligned} T^{i,j} &= \phi^i_{T^{i,j}_{index}} \\ &= f(S_{tr}^i, O^j) \end{aligned} \tag{4.42}$$

and for training sample S^i_{ir}

$$T^{i,0} > T^{i,1} > \cdots > T^{i,N_{top}-2} > T^{i,N_{top}-1} > \phi^i_l \tag{4.43}$$

for $\forall\ l \in O$ and $l \notin \{T^{i,j}_{index}\}$.

Moreover, a flag is also set to each training sample: $T^i_{flag} = -1$

2. For each template O^k, add the values of remaining outputs together:

$$Sum_k = \sum_{T^{i,j}_{index}=k} T^{i,j} \tag{4.44}$$

3. Find the highest Sum_k and move the template $O^{k'} = \arg_k(\max(Sum_k))$ to the parent set P' for next generation: $P \Leftarrow P + \{O^{k'}\}$, $O \Leftarrow O - \{O^{k'}\}$. The training samples, which index satisfy $i' = \arg_i(T^{i,j}_{index} = k')$, are given a flag with:

$$\begin{aligned} T^{i'}_{flag} &= j' \\ &= \arg_j(T^{i,j}_{index} = k'). \end{aligned} \tag{4.45}$$

4. If there still exist some flags of training samples that $T^i_{flag} = -1$, for each template O^k left in set O, get the new value of Sum_k with:

$$Sum_k = \sum_{T^{i,j}_{index}=k} \hat{T}^{i,j} \tag{4.46}$$

where,

$$\hat{T}^{i,j} = \begin{cases} T^{i,j} & : \text{ if } T^i_{flag} = -1 \\ T^{i,j} - T^{i,T^i_{flag}} & : T^i_{flag} \geq 0 \text{ and } j < T^i_{flag} \\ 0 & : T^i_{flag} \geq 0 \text{ and } j > T^i_{flag} \end{cases} \tag{4.47}$$

Figure 4.10: Examples of extracted templates in grey-scale.

if the flags of all training samples satisfy $T^{i}_{flag} \geq 0$, replace Equation 4.46 with Equation 4.44.

5. Similar to Step 3, find the template $O^{k'}$ with highest Sum_k in O and move it to P', re-set the flag of training samples which $i' = \arg_{i}(T^{i,j}_{index} = k')$, if $T^{i'}_{flag} \geq 0$, replace it with $T^{i'}_{flag} = \min(T^{i'}_{flag}, j'_r)$, $j'_r = \arg_{j}(T^{i,j}_{index} = k')$

6. Repeat step 4, 5 till the number of templates in set P' reach N_P.

Figure 4.10 shows some selected examples of the optimized templates, which are presented in grey-scale images.

4.3 Experimental Results and Discussion

In total, 10,426 digit samples from NIST Special Database 3 were extracted as the training set. Each digit image was binarized and scaled into a 64×64 binary image with 8-pixel-wide background borders. 1,000 templates were first extracted from the training set by the neural network approach discussed in Section 4.2.2. There are 100 templates for each digit. These 1,000 templates were then optimized by the evolutionary algorithm discussed in Section 4.2.3. Each template contains 11×11 control points. Because each digit has 100 templates, in each optimization computation, $N_P = 100$ and $N_O^r = 100$. We set $N_O^m = 2N_P = 200$, $N_O^c = 700$, and therefore $N_O = 1000$.

Independent 10,426 digit samples were extracted from the same database as the test set. For a comparison, we have also applied other existing algorithms to recognize the same digit samples, including decision trees and rules, fuzzified decision rules, three-layer MLP (Multi-Layer Perceptron) classifier, and the ONNC (Optimized Nearest-Neighbor Classifier) proposed by Yan [15]. We have also evaluated the classification performance using the initial templates extracted by training a feedforward neural network. Sixty-four intensities of the 8×8 image were used as the inputs of the MLP classifier and the ONNC.

Table 4.1 lists correct classification rates obtained by our classification approaches and some other classification techniques. A classifier using an decision tree of size 2163 rules achieves 98.8% correct classification on the training set and 91.4% on the test set. Using 828 pruned rules, the classifier has a slightly higher rate on the test set (91.9%) with a cost of training accuracy (down from 98.8% to 97.1%). Using the 151 simplified decision rules, we achieve 94.6% and 90.7% correct classification rates on the training set and the test set, respectively. As can be expected, performance degrades due to the simplification in converting the decision tree to the corresponding decision rules. However, after membership function were introduced into rules and optimized defuzzification applied [79], the

Table 4.1: Classification performance of using different techniques.

Techniques	Performance (%)	
	On training set	On test set
Decision tree (before pruning, size of the tree: 2163)	98.8	91.4
Decision tree (after pruning, size of the tree: 828)	97.1	91.9
Simplified decision rules (151 rules)	94.6	90.7
Fuzzified decision rules (151 rules)	97.7	95.0
Multi-Layer Perceptron (MLP) (64-100-10)	99.6	96.7
Yan's optimized prototypes (1,000 prototypes)	98.4	97.8
Chi's combined classifier	98.8	98.6
1,000 optimized templates from our approach		96.4
Our combined classifier	98.6	97.0

classifier using fuzzified decision rules achieves 97.7% and 95.0% correct classification on the training set and the test set, respectively. Another classifier based on 64-100-10 Multiple Layer Perceptron (MLP) achieves a 99.6% and 96.7% classification rate. Yan's optimized prototypes achieves a better performance with 98.4% and 97.8%. Chi's combined classifier, which combining fuzzified decision rules, Markov chain matching, and Yan's optimized prototypes [7], achieved the best classification rates among these classifiers with 98.8% and 98.6% on the training set and test set, respectively. As shown in table 4.1, the optimized templates extracted using our approach achieves 96.4% correct classification rate on the test set and our combined classifier achieves 97.0% classification rate which is at high end of recognition performance. However, the testing on true classes only are not sufficient to compare the performance of these classifiers for connected digit string classification where the ability of a classifier to reject non-digit

Figure 4.11: Examples of non-digit patterns.

Table 4.2: Experimental results on non-digit patterns with different classifiers.

Techniques	Rejection rate on illegible objects (%)	Rejection rate on training set (%)	Recognition rate on training set (%)
Our combined classifier	90.7	22.32	99.80
Yan's optimized prototypes	69.3	23.23	99.90
MLP (64-100-10)	44.5	25.4	99.90

patterns is very important. We have also conducted an experiment on non-digit patterns, like those shown in Fig. 4.11.

We extracted 10,426 non-digit patterns by merging two parts of isolated digits, the left part of the right digit and the right part of the left digit. The choosing of two digits, the width and relative position of each parts, and the overlap degree are set randomly. Table 4.2 shows our combined classifier can achieve the best performance on rejecting unsure patterns.

Further comparisons of these classifiers will be presented in chapter 5 and chapter 6 when the classifiers are applied to the segmentation-based and segmentation-free recognition of connected digit strings.

The main limitation of this multi-module classifier is its high computational cost in both storage space and computing time. Most of the time is spent in the template based classifier module. With 1,000 templates for the templates-based classifier, our experiments carried out on a 333 MHz Pentium II computer (64 MB RAM) with Linux operating system show that it takes tens of hours for the multi-module classifier to recognize all the 10,426 digits. Note that the time spent includes the overhead of reading the templates from the hard disk for

each test pattern since the internal memory is not large enough to store all the templates.

4.4 Concluding Remarks

A multi-module classifier is presented in this chapter to recognize isolated/separated handwritten digits and reject non-digit patterns. There are four recognition modules in this classifier: three neural network-based modules and a template-based module. Each of the three neural network-based module is specifically designed to deal with a type of digit image features, namely, one module for handling structural features, one for handling directional features, and another for taking care of intensity features. For the template-based module, each template presented by a Pixel-to-Boundary Distance Map (PBDM) is approximated by a rational B-spline surface with a set of control knots. Each module is trained independently. Neural network-based modules are trained by back-propagation algorithm with the conjugate-gradient optimization technique. In the template-based module, a cost function of the magnitude and gradient of PBDM is used for extracting initial templates and an evolutionary algorithm is used for optimizing the templates. The outputs from all of the four modules are then combined by a trained neural network module. Experimental results on NIST Special Database 3 show that the multi-module classifier can achieve a good recognition performance, in particular, more reliable in rejecting non-digit patterns compared with other existing techniques. Reliability in rejecting non-digit patterns is a very desirable feature for connected digit string recognition. The main limitation of this multi-module classifier is its high computational cost in both storage space and computing time.

Chapter 5

Segmentation-Based Recognition of Handwritten Digit Strings

The segmentation and recognition of connected characters is a key problem in the development of Optical Character Recognition (OCR) systems. Many algorithms have been proposed in the recent years. Lu given an overview on various techniques for the segmentation of machine printed characters in 1995 [45], while he and Shridhar have also written lately on various approaches for segmenting handwritten characters [46].

Cheriet *et al* proposed a region-based method by which a pair of background regions are identified by independent top-down and bottom-up matching and a segmentation path is constructed by connecting the matched regions [47]. If no pair of matched regions are found, a vertical segmentation path is constructed either upward from a lower region or downward from an upper region. Strathy *et al* introduced a segmentation algorithm based on contour structure features [49]. In their algorithm, the contour chains of a binary digit image are analyzed and high curvature points are identified. A segmentation path is assumed to pass a pair of high curvature points. Nine features are computed for each segmentation path and used to sort all the possible paths. Chi *et al* developed a contour curvature based algorithm for separating single- and double-touching handwritten

digit strings [81]. In their algorithm, all segmentation path candidates are sorted by a set of features with corresponding weight functions, and tested by a nearest neighbor classifier. Yu and Yan developed a morphological technique to analyze touching digit strings based on the distribution of a set of topological feature points on the contour of a digit image [48]. Westall and Narasimha presented a vertex directed segmentation algorithm [82]. The algorithm first identifies the vertices that are formed by converging vertically oriented edges of adjacent strokes. A segmentation path is formed by connecting the selected vertices with extension to the top and bottom of the image. The geometric features of the left and right parts of the image are used to validate the segmentation path. Although many approaches have been proposed for segmenting and recognizing connected handwritten characters, much improvement is still required in order to build a system of practical uses. In particular, further enhancement is much demand in reducing the number of candidate paths to be tested and achieving a smaller rejection rate while maintaining a reasonable high recognition rate.

In this chapter, a new segmentation and recognition approach based on the background (regions excluding the characters) skeleton of a digit string image is presented.

5.1 Types of the Connections in Handwritten Digit Strings

Normally, the connections of adjacent handwritten digits can be categorized into the following three types [82]:

- Overlapping of two vertically oriented strokes from two adjacent digits;
- the end of a stroke of the left/right digit touching the side of a stroke of the right/left digit; and

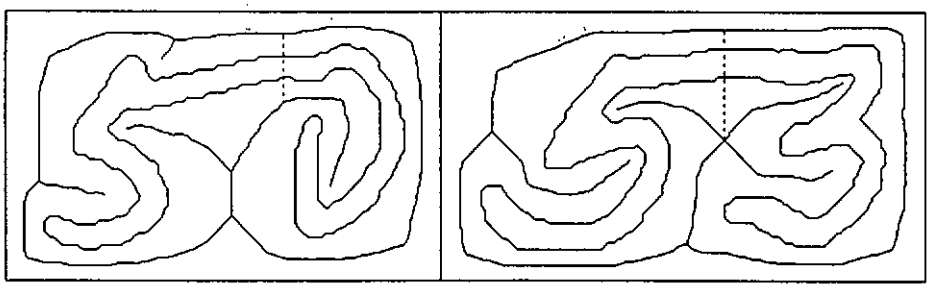


Figure 5.1: Connection type 3: two strokes touch end to end.

- two strokes from two adjacent digits touching end to end as shown in Fig. 5.1.

Among three types of connections, the digit strings of type 3 connection is the most difficult to handle when only the image contours are used, especially when two strokes have a nearly constant width as shown in Fig. 5.1. It is almost impossible to extract high curvature points or vertices in such a case. However, a segmentation path can be formed based on the background skeleton for the digit strings as shown in Fig. 5.1. The dotted line in each case represents a good segmentation path that extends a high curvature point or a fork point (to be defined in the next section) on the background skeleton to the top (could be the bottom) of the image. Another advantage of using the background skeleton is that after matched feature points are identified, a complete segmentation path can be constructed by extending the partial path to the top and/or bottom of an image by tracing the background skeleton.

5.2 Feature Point Extraction

In order to get a complete background skeleton, each digit image is placed in a rectangle block with a small space to each of the four frames. The background regions of a digit string image is thinned by using a skeletonization algorithm adapted from Hilditch's thinning algorithm [62]. For explaining our algorithm better, the definitions of different types of segments and feature points on the background skeleton are given first in the following:

- *Base segment:* a segment generated between the foreground and the top or bottom frame of the image, such as the thick lines shown in Fig. 5.2(a). The upper one is named as the upper-base segment and the lower one as the lower-base segment.

- *Side segment:* a segment generated between the foreground and the left or right image frame. Side segments are not considered in the subsequent processing.

- *Branch segment:* a segment connected to a base segment (excluding side segments), such as the thick lines shown in Fig. 5.2(b). Similar to a base segment, a branch segment connected to the upper-base segment is named as the upper-branch segment and a branch segment connected to the lower-base segment as the lower-branch segment.

- *Hole segment:* a segment generated from a hole region of the background, such as the thick lines shown in Fig. 5.2(c).

- *Upper segment:* a upper-base segment or a upper-branch segment.

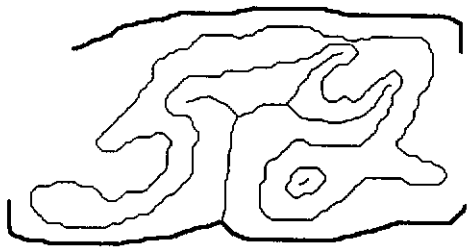
- *Lower segment:* a lower-base segment or a lower-branch segment.

- *Side point:* the end point on a base segment after removing a side segment, such as the points marked by $\diamond$ shown in Fig. 5.3.

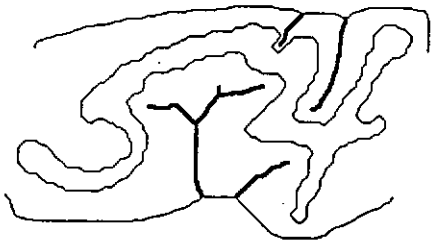
- *Fork point:* a point on a segment which has more than two connected branches, such as the points marked by $\bullet$ shown in Fig. 5.3.

- *End point:* a point on a segment which has only one neighbor (excluding the side points), such as the points marked by $\square$ shown in Fig. 5.3. An angle is assigned to each end point to indicate the orientation of the segment at the point.

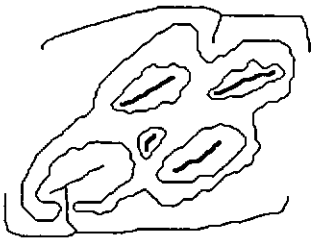
- *Isolated point:* a single point generated inside a circular hole.



(a)



(b)



(c)

Figure 5.2: Examples of background skeletons with think lines indicating (a) base segments; (b) branch segments; and (c) hole segments.

- *Corner point:* a point on a segment where the direction of the line changes sharply, such as the points marked by o shown in Fig. 5.3.

- *Feature point:* a fork point, an end point, or a corner point.

The extraction of feature points includes two steps. The first step is to find all fork points and end points on the background skeleton. The second step is to search for all corner points. To find corner points, we first estimate the direction of a segment at each point. Let the line directions at the points before and after point l be $\alpha_{pre,l}$ and $\alpha_{post,l}$, which should be between $-\pi$ and $+\pi$. We define

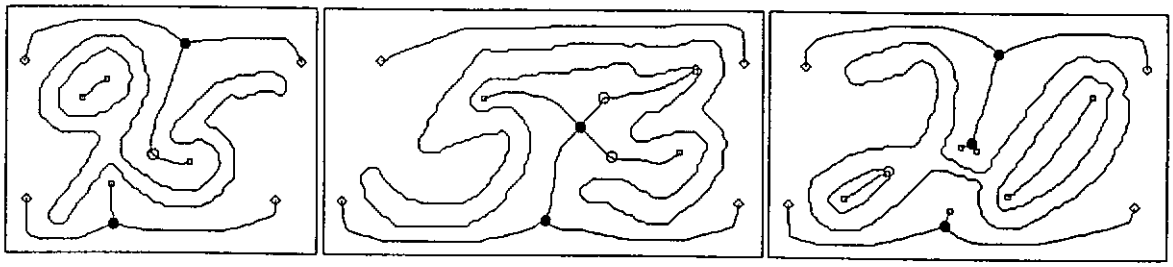


Figure 5.3: Examples of feature points on background skeletons.

$$\alpha_{pre,l} = \text{direction}\{p_{l-N}, p_{l-N-1}, \cdots, p_{l-1}, p_l\} \tag{5.1}$$

$$\alpha_{post,l} = \text{direction}\{p_l, p_{l+1}, \cdots, p_{l+N-1}, p_{p+N}\} \tag{5.2}$$

where N is the number of points used in the estimation. The LSEA (Least Square Error Approximating) method is used to estimate the line direction at a point. The curvature at point l, α_l, can then be obtained by:

$$\alpha_l = \begin{cases} D & : \text{ if } \quad D \leq \pi \\ 2\pi - D & : \text{ if } \quad D > \pi \end{cases} \tag{5.3}$$

where $D = |\alpha_{post,l} - \alpha_{pre,l}|$. The curvature value α_l is confined to the range of 0 to $+\pi$. Whether a point l is a corner point or not is dependent on its curvature value α_l. If α_l is a local maximum among neighboring points and it is larger than a threshold, then, point l is labeled as a corner point. Our method is immune to small noise and can also locate a corner point on a smooth curve, such as the corner point in the segment between '9' and '5' as shown in Fig. 5.3.

5.3 Construction of Segmentation Paths by Matching Feature Points

A segmentation path is constructed by connecting several feature points on the background skeleton with possible extension to the top and/or bottom of an

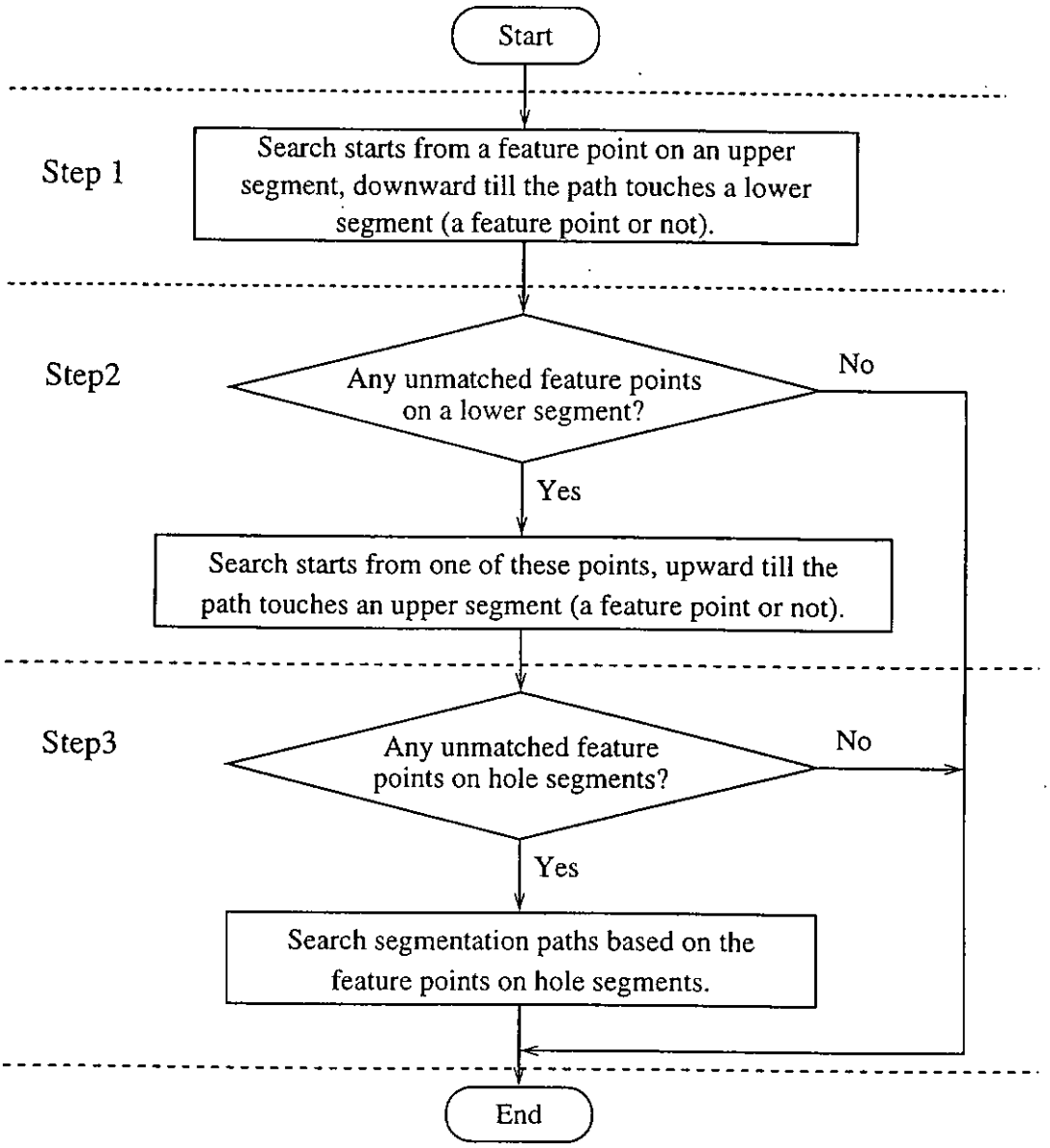


Figure 5.4: Flowchart of the construction of segmentation paths by a three-step searching process.

image. A three-step searching scheme shown in Fig. 5.4 is adopted to search the feature points on upper segments, lower segments and hole segments and to construct segmentation paths. The first step is to search the feature points from the top to bottom (top-down searching). If there exists unmatched feature points on lower segments, a bottom-up searching from these points is performed in the second step. If there exist unmatched feature points on hole segments, another search is conducted in the third step in which searching is first done upward and downward separately and the two traced segments are then integrated into a single segmentation path. As a result, all possible segmentation paths can be identified by this searching scheme.

5.3.1 Single-Direction Search

Figure 5.5 shows the flowchart of the top-down search in the first step. It starts from a feature point on an upper segment and end at a point (a feature point or not) on a lower segment. The important part of the searching is to find the matched feature points on hole segments and lower segments. In the top-down searching, the feature points to be matched must satisfy the following constraints:

- The coordinates of a feature point to be matched, (x, y), must satisfy $x_{curr} - x_{zone} \leq x \leq x_{curr} + x_{zone}$ and $y \geq y_{curr} + y_{zone}$, where x_{curr} and y_{curr} are the coordinates of the current feature point, and x_{zone} and y_{zone} are the parameters specifying the searching scope, as the dotted lines shown in Example 1 of Fig. 5.6(a).

- The two points are not connected through the background skeleton.

- The two points have to be face to face, that is, if the orientation of the segment at the current point is within $(0, \pi)$, the matching point should be within $(-\pi, 0)$, and vice versa.

- There is no background segment in a parallelogram region between two

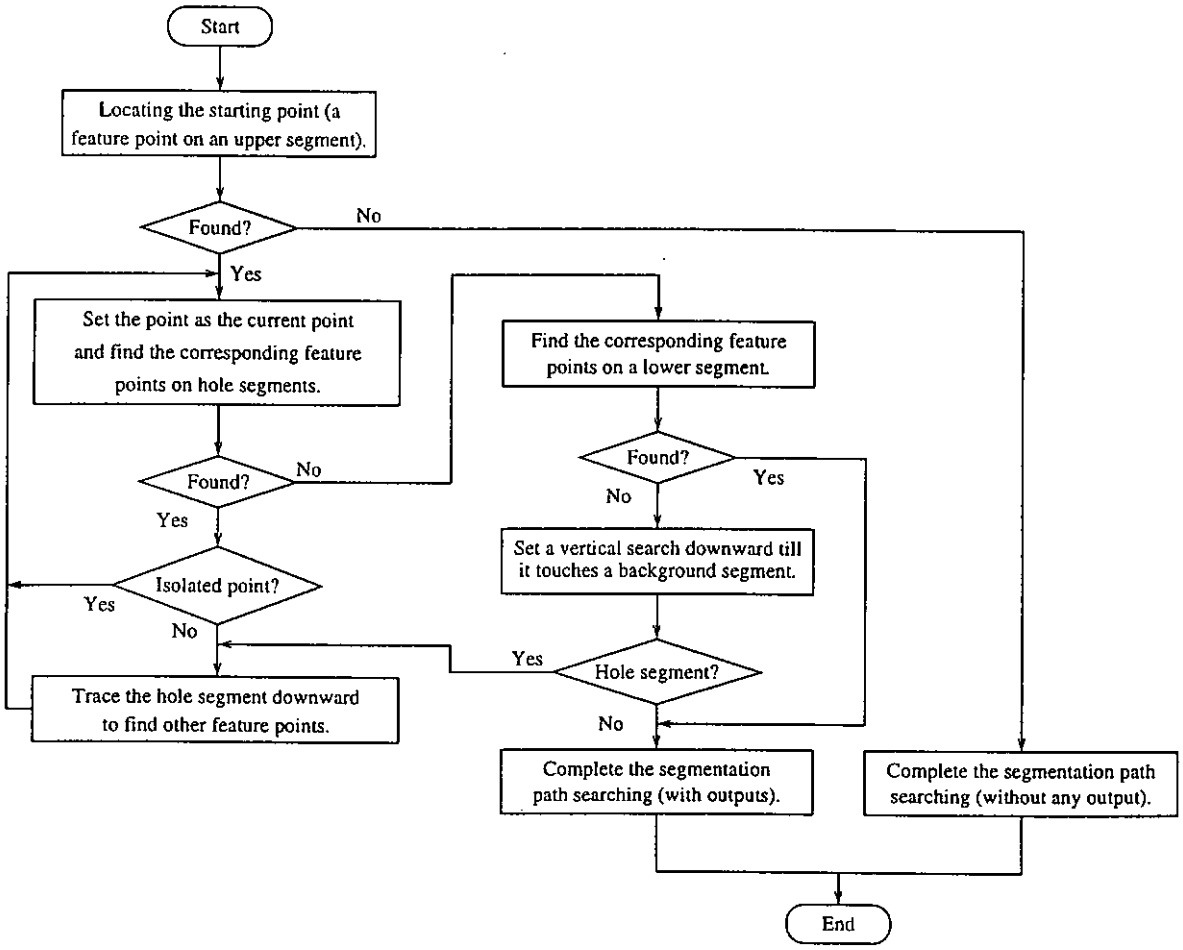


Figure 5.5: Flowchart of the top-down searching of segmentation paths.

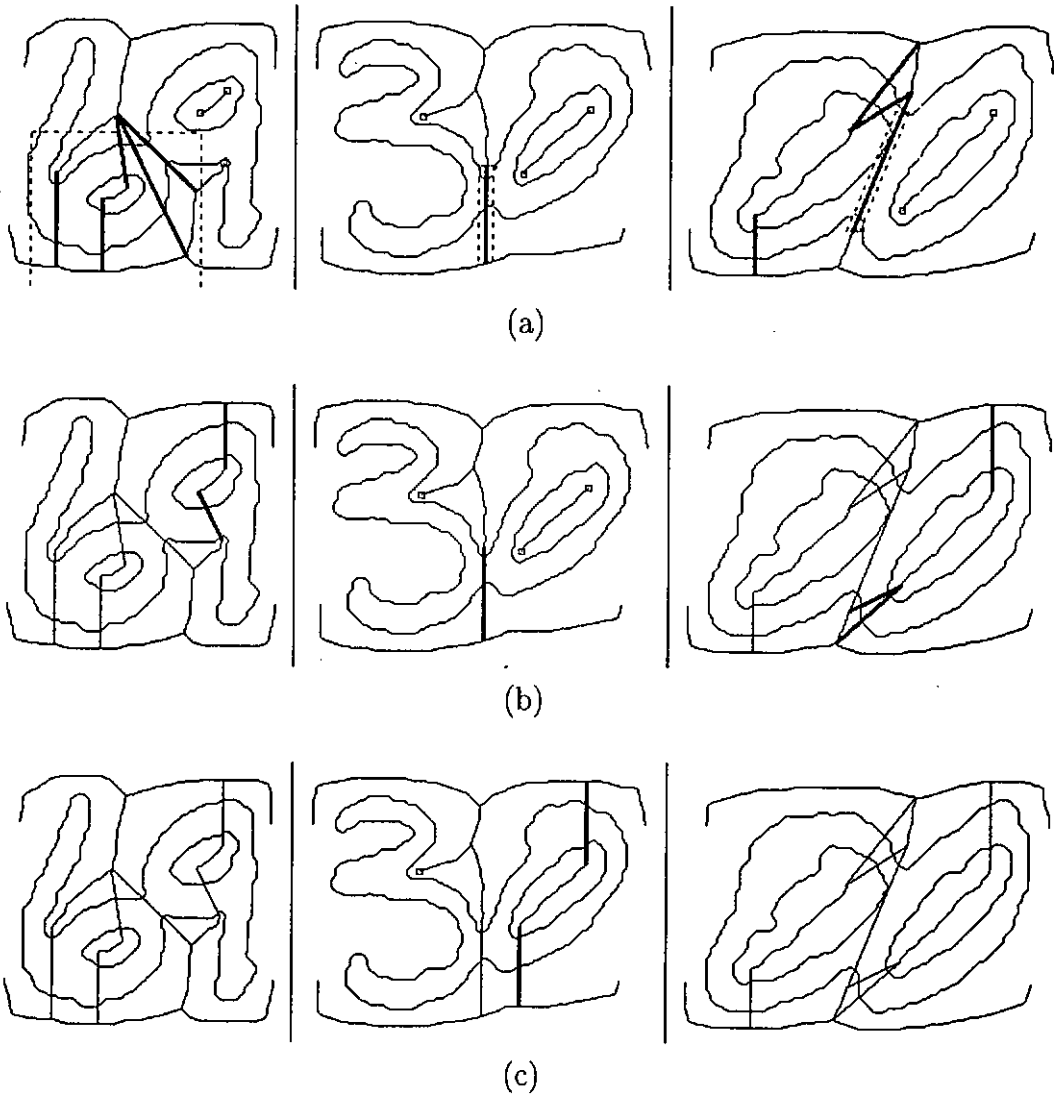


Figure 5.6: Examples of segmentation paths (feature points marked by '□' are unmatched points): (a) segmentation paths found (dark lines) by Step 1 (top-down searching); (b) segmentation paths found by Step 2 (bottom-up searching); and (c) segmentation paths found by Step 3 (searching from a hole segment).

points. The four vertices of the parallelogram are: $(x_{curr} - \frac{p_{width}}{2}, y_{curr} - p_{height})$, $(x_{curr} + \frac{p_{width}}{2}, y_{curr} - p_{height})$, $(x_{corresp} - \frac{p_{width}}{2}, y_{corresp} + p_{height})$, $(x_{corresp} + \frac{p_{width}}{2}, y_{corresp} + p_{height})$, where $x_{corresp}$ and $y_{corresp}$ is the coordinates of the matching point, p_{width} is the width of the parallelogram and p_{height} is the vertical distance from the upper side and the lower side of the parallelogram to the two points, as the dotted line shown in Example 2 of Fig. 5.6(a).

- There exists only one background-foreground transition in the straight line connecting the current point to the matched point.

Quite often, more than one feature points match the current feature point, such as Example 1 shown in Fig. 5.6(a). All the possible segmentation paths should be recorded for further processing. On the contrary, sometimes no feature point would match the current feature point, such as Example 2 shown in Fig. 5.6(a). In this case, a default vertical searching path is constructed downward till it touches a background segment (either a hole segment or a lower segment). Similar to matching feature point, several restrictions are posed for validating a touching point:

- There exists no connection between the current feature point and the touched point.

- No background segment is found in a rectangle region between the two points, The four corner points are: $(x_{curr} - \frac{p_{width}}{2}, y_{curr} - p_{height})$, $(x_{curr} + \frac{p_{width}}{2}, y_{curr} - p_{height})$, $(x_{reached} - \frac{p_{width}}{2}, y_{reached})$, $(x_{reached} + \frac{p_{width}}{2}, y_{reached})$, where $x_{reached}$ and $y_{reached}$ are the coordinates of the touching point, and $x_{curr} = x_{reached}$, as the dotted line shown in Example 2 of Fig. 5.6(a).

- There exists one background-foreground transition in the straight line connecting the two points.

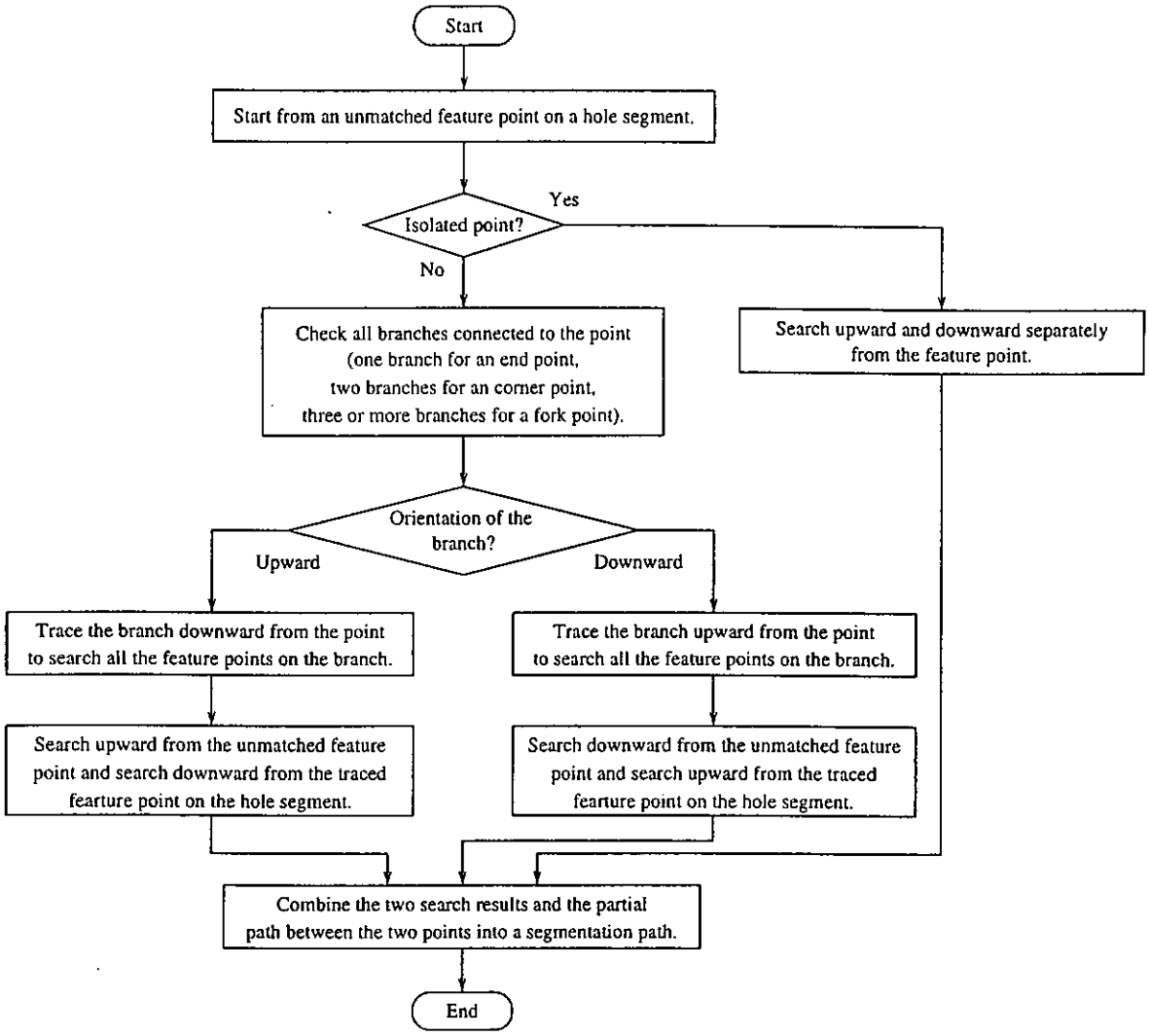


Figure 5.7: Flowchart for the searching beginning at a feature point on a hole segment.

The bottom-up searching in the second step is similar to the top-down searching in the first step. The difference is that the former is searching upward and the latter is searching downward, so the searching scope should be changed to: $x_{curr} - x_{zone} \leq x \leq x_{curr} + x_{zone}$ and $y \leq y_{curr} - y_{zone}$.

5.3.2 Search in Two Directions

Figure 5.7 is the flowchart for Step 3 of our searching scheme. The searching process begins from an unmatched feature point on a hole segment and search in two directions. The searching process is similar to those in the first two steps.

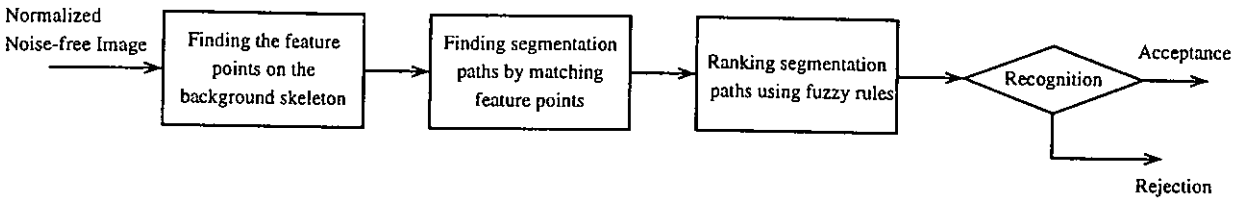


Figure 5.8: Flowchart of our background-thinning-based approach for segmenting connected handwritten 2-digit strings.

The key problem here is to determine the search direction upward or downward, for each feature point on a hole segment. In our approach, the orientation of the branch connected to the feature point is used to determine the search direction. After the searching, two searched segments and the segment between the unmatched feature points are integrated into a single segmentation path. As Example 2 shown in Fig. 5.6(c), there are two unmatched points on the hole segment after Steps 1 and 2, the searching starts from the two points, one upward and one downward, the two traced segments and the segment between the two points represents a possible segmentation path.

5.4 Segmentation and Recognition of 2-Digit Strings

To compare our approach with other segmentation techniques, we first apply our segmentation algorithm to handwritten 2-digit strings.

Figure 5.8 shows the flowchart of our background-thinning based digit segmentation approach of handwritten handwritten 2-digits. Among the four steps, the first two steps have been introduced in Sections 5.2 and 5.3, respectively. We will present the remaining two steps in the next two sections.

5.4.1 Ranking Segmentation Paths Using Fuzzified Decision Rules

Since Zadeh introduced the fuzzy set theory in 1965 [83], the fuzzy logic approach has been an increasing interest of many scientists and engineers for opening up a new area of research and problem solving. In the recent years, we have seen the booming of applications of fuzzy algorithms in pattern recognition. Increased popularity of fuzzy algorithms because (1) fuzzy rules are found to be naturally effective for any human-like cognition systems such as image understanding and pattern recognition and (2) fuzzy sets provides a good platform for dealing with noisy, and imprecise information which is often encountered in our daily life [84].

In separating and recognizing connected handwritten character recognition, a human being performs far better than a machine. We believe that human beings can perform better partially because they have sufficient knowledge about the problem and they adopt a very robust recognition scheme, in particular, in dealing with noisy and fuzzy patterns. As a fuzzy logic approach is a way to achieve this robustness, we apply fuzzified decision rules to prioritize the segmentation paths obtained using the method discussed in Section 5.3. In our approach, the membership grade to which a candidate is a good segmentation path is determined by fuzzified decision rules with the nine properties associated with a segmentation path and the separated parts segmented by the path, which will be discussed next. Ranking of the segmentation paths can decrease the computational complexity and improve the recognition accuracy.

5.4.1.1 Properties of a Segmentation Path

A digit string is split into the left and right parts by a segmentation path. Some parameters associated with this segmentation are given in the following:

- L = length of a segmentation path in the number of pixels;

- L_f = number of pixels on a segmentation path which are in the foreground of the image;
- N_1 = number of pixels in the left part;
- N_2 = number of pixels in the right part;
- $x_L^{(1)}, x_L^{(2)}$ = horizontal coordinates of the leftmost pixel of each part;
- $x_R^{(1)}, x_R^{(2)}$ = horizontal coordinates of the rightmost pixel of each part;
- $y_H^{(1)}, y_H^{(2)}$ = vertical coordinates of the highest pixel of each part; and
- $y_L^{(1)}, y_L^{(2)}$ = vertical coordinates of the lowest pixel of each part.

Segmentation paths are ranked based on the nine properties associated each path and the separated parts by the path. The nine properties are adapted from [81], which were successfully used to rank segmenting paths for recognizing handwritten single- and double-touching digit strings.

- F_0 = ratio between the sizes of left and right parts:

$$F_0 = \begin{cases} N_1/N_2 & : \text{ if } \quad N_1 \leq N2, \\ N_2/N_1 & : \text{ otherwise.} \end{cases} \tag{5.4}$$

  Values of F_0 are in the range of $[0.0, 1.0]$.

- F_1 = ratio between the value of L_f and the height of the image:

$$F_1 = \frac{L_f}{\max(y_L^{(1)}, y_L^{(2)}) - \min(y_H^{(1)}, y_H^{(2)})} \tag{5.5}$$

  F_1 is set to 1.0 when its value is greater than 1.0.

- F_2 = ratio between the horizontal length of any overlap of the two parts, and the smaller of the widths of the two parts:

$$F_2 = \frac{x_R^{(1)} - x_L^{(1)}}{\min\left[(x_R^{(1)} - x_L^{(2)}), (x_R^{(2)}, x_L^{(2)})\right]} \tag{5.6}$$

  F_2 is set to 1.0 when its value is greater than 1.0.

- F_3 = ratio between the heights of the two parts:

$$F_3 = \begin{cases} (y_L^{(1)} - y_H^{(1)})/(y_L^{(2)} - y_H^{(2)}) & : \text{ if } (y_L^{(1)} - y_H^{(1)}) \leq (y_L^{(2)} - y_H^{(2)}) \\ (y_L^{(2)} - y_H^{(2)})/(y_L^{(1)} - y_H^{(1)}) & : \text{ otherwise.} \end{cases} \tag{5.7}$$

  Values of F_3 are in the range of $[0.0, 1.0]$.

- F_4 = ratio between the widths of the two parts:

$$F_4 = \begin{cases} (x_R^{(1)} - x_L^{(1)})/(x_R^{(2)} - x_L^{(2)}) & : \text{ if } (x_R^{(1)} - x_L^{(1)}) \leq (x_R^{(2)} - x_L^{(2)}) \\ (x_R^{(2)} - x_L^{(2)})/(x_R^{(1)} - x_L^{(1)}) & : \text{ otherwise.} \end{cases} \tag{5.8}$$

  Values of F_4 are in the range of $[0.0, 1.0]$.

- F_5 = average angle, with the horizontal axis, of the segments of a path which are in the foreground of an image. Values of F_5 are in the range of $[-\pi/2, \pi/2]$.

- F_6 = normalized distance of the center of segmentation path (x_{lc}) to the center of the image on the horizontal axis (x_C):

$$F_6 = \frac{2|x_{lc} - x_C|}{x_R^{(2)} - x_L^{(1)}} \tag{5.9}$$

  where x_{lc} is obtained by

$$x_{lc} = \frac{\sum_{i=0}^{L} x_i^s}{L} \tag{5.10}$$

  where x_i^s $(i = 0, 1, ..., L)$ are the horizontal coordinates of pixels on the segmentation path.

- F_7, F_8 = ratios of the width to the height of the left part and right part, respectively.

5.4.1.2 Fuzzified Decision Rules

Interactive dichotomizing decision trees and rules were first proposed by Quinlan [85]. A rule extracted from a Quinlan's decision tree has the following form:

$$F_1(f_{1j}) \wedge F_2(f_{2j}) \wedge ...F_n(f_{nj}) \Longrightarrow C(L_j) \tag{5.11}$$

$F_i(f_{ij})$ is true when feature F_i takes a value in the region f_{ij}. For a discrete feature, f_{ij} is a single value. $C(L_j)$ denotes the class label associated with leaf L_j.

Decision rules work well when input data is noise-free. However, its performance degrades quite a bit when input data is uncertain or noisy. Chi *et al* proposed to use fuzzified decision rules to deal with the uncertain problems in handwritten digit recognition [7].

There are two definitions of value regions used for a feature (F_i) with continuous values in Quinlan's decision rules. They are $F_i > f_{ij}$ and $F_i \leq f_{ij}$.

For $F_i > f_{ij}$, we define a membership function as

$$m(F_i) = \begin{cases} 1.0 & : \text{ if } \quad F_i > f_{ij} \\ 0.0 & : \text{ if } \quad F_i \leq (1-\alpha) f_{ij} \\ \frac{F_i - (1-\alpha) f_{ij}}{\alpha f_{ij}} & : \text{ if } \quad (1-\alpha) f_{ij} < F_i \leq f_{ij} \end{cases} \tag{5.12}$$

where α is an extension factor to reflect the fuzziness of the data.

For $F_i \leq f_{ij}$, we define a membership function as

$$m(F_i) = \begin{cases} 1.0 & : \text{ if } \quad F_i \leq f_{ij} \\ 0.0 & : \text{ if } \quad F_i > (1+\alpha) f_{ij} \\ \frac{(1+\alpha) f_{ij} - F_i}{\alpha f_{ij}} & : \text{ if } \quad f_{ij} < F_i \leq (1+\alpha) f_{ij} \end{cases} \tag{5.13}$$

Using the membership grades instead of binary values 0 and 1 for the degree to which a rule is triggered, the decision rules becomes a set of fuzzy rules.

Nine features, $F_0, F_1, \cdots, F_8$, discussed in Section 5.4.1.1 are used for ranking segmentation paths. A rule outputs one of two classes: a good segmentation path and a bad segmentation path. An example of a rule we used for classifying a segmentation path is given below:

IF $F_0 \leq 0.70$ AND $F_1 > 0.10$ AND $F_5 > -0.73$ AND $F_5 \leq 0.84$ AND $F_8 > 0.89$, THEN the path is a bad segmentation path.

5.4.2 Defuzzification

The centroid defuzzification technique is utilized to defuzzify the classifier output:

$$O_p = \frac{\sum_{i=1}^{N} w_i D_p^i O^i}{\sum_{i=1}^{N} w_i D_p^i} \tag{5.14}$$

where N is the number of fuzzy rules and w_i weighs the confidence of rule i, that is, the degree that an input pattern fits rule i, and O_p represents the output of pattern p. In this application, we have two classes, good and bad segmentation paths. The value of O^i is assigned as 1 if the output of rule i is class 'good segmentation', otherwise it is assigned to -1. If the output O_p is larger than 0, the segmentation path is regarded as a 'good segmentation', otherwise a 'bad segmentation'. D_p^i measures how the pth pattern matches the antecedent conditions (IF-part) of the ith rule. D_p^i is given by the product of the matching degrees of the pattern in the fuzzy subsets which the ith rule holds, that is,

$$D_p^i = \prod_{k=1}^{M_i} m_{ki} \tag{5.15}$$

where M_i is the number of features used in that rule (called the size of a rule) and m_{ki} is the membership grade of the k-th feature in the fuzzy subset that the ith rule holds. We use O_p, which reflects the degree to which a candidate is a good segmentation path, to rank all possible segmentation paths.

5.4.3 Removing Redundant Segmentation Paths

Because all possible segmentation paths are extracted, some paths may be similar to others. Two criteria are used to remove the redundant segmentation paths. Firstly, we define:

- N_{ij} = number of foreground pixels enclosed by the ith and the jth segmentation paths.

- C_{ij} = ratio between N_{ij} and $(L_f^i + L_f^j)$, where L_f^i is the L_f (see Section 5.4.1.1) of the ith path and L_f^j is the L_f of the jth path.

$$C_{ij} = \frac{N_{ij}}{L_f^i + L_f^j} \tag{5.16}$$

Two thresholds, $T_{N_{ij}}$ and $T_{C_{ij}}$ are set to determine whether a pair of segmentation paths are similar. If either N_{ij} or C_{ij} is smaller than a pre-set threshold, we consider that two segmentation paths are similar and the one with smaller O_p will be removed.

5.4.4 Experimental Results and Discussion

All the parameters used in the construction of segmentation paths must be set according to the size of an image. In our experiments, the height of the normalized image is 160 pixels and we set $x_{zone} = 60$, $y_{zone} = 10$, $p_{width} = 9$, $p_{height} = 5$, $T_{N_{ij}} = 50$, and $T_{C_{ij}} = 3.5$.

Decision rules were generated and tested using 6,697 manually classified segmentation paths, which were extracted from 823 2-digit strings from NIST Special Database 3. With 4,000 as training samples, 28 decision rules were generated. Among them, 17 rules were fuzzified and the others were discarded because they have little impact on training samples. Experimental results show that the correct path classification rate is 81.1% on the 4,000 training paths and 79.9% on the remaining 2,697 test paths. Figure 5.9 shows some examples of segmentation paths and winning paths.

A comparison of the correct path classification rates among the fuzzified decision rules, straightforward decision trees (unpruned and pruned) [85], and multilayer perceptron (MLP) classifier is shown in Table 5.1. The inputs of these classifiers are the nine properties of a path given in Section 5.4.1.1 and the outputs are two classes, good segmentation path and bad segmentation path. The size of the MLP classifier is 9-46-2. Evaluated on the test set, the correct path

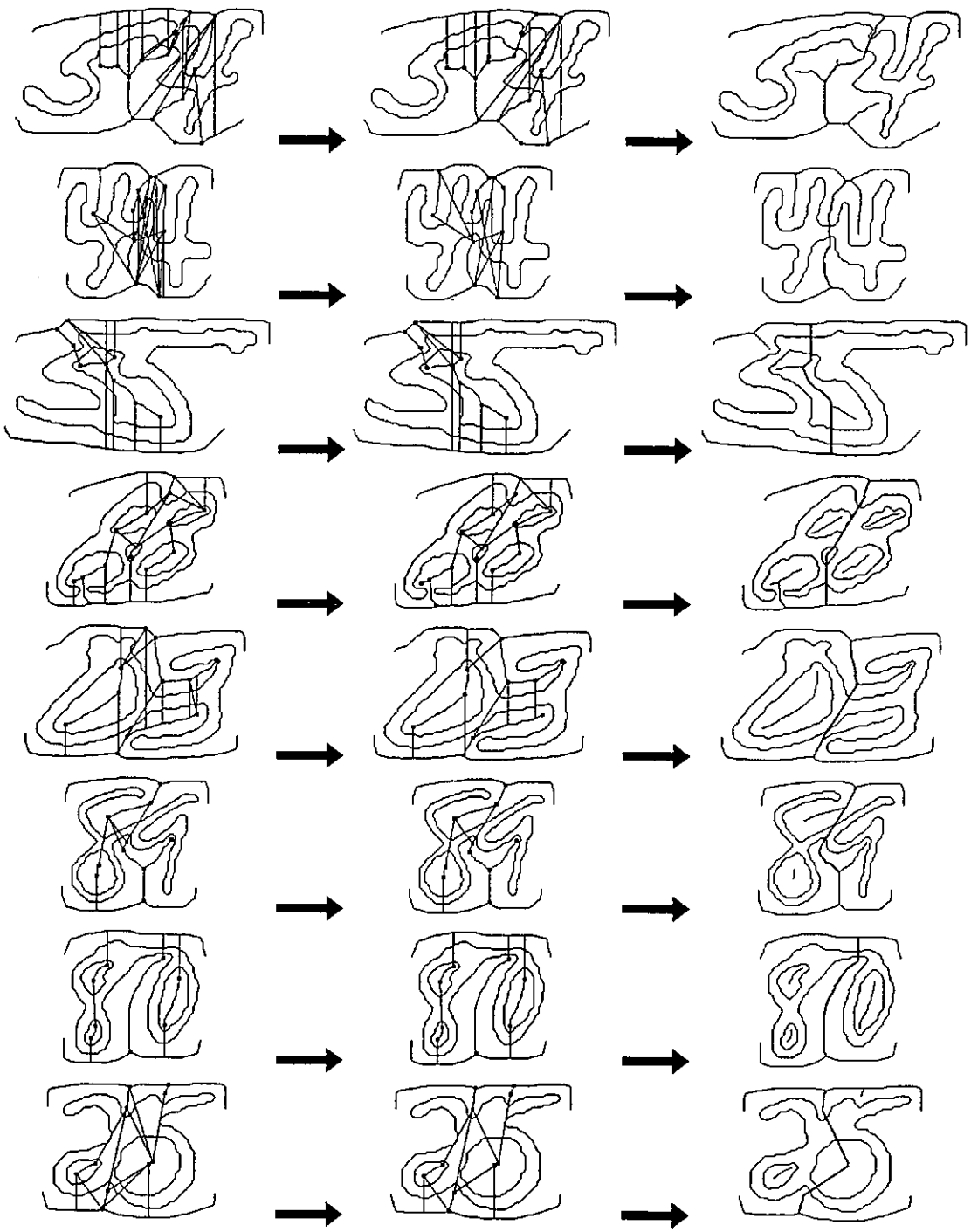


Figure 5.9: Examples of connected handwritten digits with (a) all possible segmentation paths; (b) segmentation paths after redundancy removing, and (c) winning segmentation paths (dark lines).

Table 5.1: A comparison of the correct path classification rates among fuzzified decision rules, straightforward decision trees (unpruned and pruned) and multi-layer perceptron classifier.

Techniques	Training set (%)	testing set (%)
Fuzzified decision rules	81.1	79.9
Decision tree (unpruned)	98.1	73.9
Decision tree (pruned)	93.5	76.2
MLP	85.0	81.6

classification rate of fuzzified decision rules is better than those of the straightforward decision trees (unpruned or pruned), but slightly lower than that of the MLP. However, our approach of path ranking based on the fuzzified decision rules has the following advantages:

1. The output of fuzzified decision rules is a membership degree to which a candidate is a good segmentation path, which is more suitable for the ranking problem.

2. Using only 17 rules is computationally efficient.

3. Rules extracted from human recognition experience can be easily incorporated into our system.

Since the path classification rate is not high enough, we tested a few top-ranked paths of a string based on the digit recognition results in order to determine a good enough segmentation path. The separated parts were recognized using an optimized nearest-neighbor classifier proposed by Yan [69]. Sixty-four intensities on the 8×8 image, rescaled from an original 160×160 normalized binary image, are used as features. The classifier returns both the assigned class and the Euclidean distance of the image from the closest class prototype. The distance is termed as the 'recognition measure' and used as an estimation of the reliability of a classification [81].

Table 5.2: Experimental results on 2-digit test strings with different recognition measure radius.

Recognition measure radius (a)	Rejected	Correct	Wrong
1.2	958 (28.6%)	2324 (97.0%)	73 (3.0%)
1.6	353 (10.5%)	2825 (94.1%)	177 (5.9%)
1.8	216 (6.4%)	2920 (93.0%)	219 (7.0%)
2.0	157 (4.7%)	2958 (92.5%)	240 (7.5%)
2.2	127 (3.8%)	2954 (91.5%)	274 (8.5%)

The classifier was trained using isolated digits extracted from NIST special database 3. In total, 53,449 isolated digits were extracted and classified for training the classifier, and other 53,185 samples for testing. The correct rate is 98.9% for training samples and 97.8% for test samples, without rejection. The mean values M_i $(i = 0, 1, \ldots, 9)$ of the recognition measures for the correctly classified digits in the training samples were obtained and used to decide whether a classification was accepted or not. On the other hand, 3,355 2-digit strings, which were also extracted from NIST special database 3, were used as the test data for the whole system. Note that the digits from these 4,178 2-digit strings, 3,355 for testing the whole system and the 823 for generating and testing fuzzy decision rules) were not included in training the optimized nearest-neighbor classifier. For the 3,355 2-digit strings, average 8.1 and 7.4 segmentation paths were constructed from each digit string before and after removing redundancy.

Suppose that a tolerant radius is a. If the recognition measure is smaller than aM_i, then the classification is accepted. Table 5.2 shows the experimental results with different recognition measure factors. The greater the rejection measure factor, the lower both the rejection rate and correct classification. Figure 5.9 shows some digit strings with possible and winning segmentation paths.

For a comparison, we also applied a few other algorithms to separate the same 2-digit strings used in our experiments. These algorithms include the MCM (Modified Curvature Method) of Chi *et al* [81], and those of Fenrich [64], Fujisawa

Table 5.3: A performance comparison of our approach with other digit separation algorithms on 2-digit strings.

Techniques	High rejection rate (%)		Low rejection rate (%)	
	Rejection	Wrong	Rejection	Wrong
Our algorithm	28.6	3.0	4.7	7.5
MCM of Chi [81]	32.7	4.9	3.7	10.8
Fenrich algorithm [64]	53.3	5.8	5.3	27.1
Fujisawa algorithm [86]	33.5	7.7	0.7	21.9
Zhao *et al* algorithm [87]	38.3	4.4	3.4	13.3

et al [86] and Zhao *et al* [87]. Table 2 summarizes the experimental results of these algorithms together with ours.

We can see from Table 5.3 that our background-thinning based approach for segmenting and recognizing connected 2-digit strings can produces results that compare favorably with those from the other techniques tested. Moreover, our approach can deal with both single- and multi-touching problems in digit string segmentation and achieve a correct rate of 92.5% with only 4.7% of digit strings rejected.

5.5 Segmentation and Recognition of Connected Digit Strings with Unknown Length

The main difference between the segmentation methods for recognizing connected 2-digits and connected digit strings with unknown length is that we do not have enough information to prioritize segmentation paths in the latter case. All the segmentation paths in a digit string with unknown length can only sorted by their x coordinates of the mass center x_{mass}. To achieve the best recognition performance, we adopt a dynamic programming technique that also makes use of the length estimation.

5.5.1 Dynamic Programming

A dynamic programming based approach is used to merge and split the segments from the segmentation step discussed in Section 5.3. Dynamic programming is the earliest technique applied in sequential pattern recognition, such as speech recognition. Fu *et al* gave a very useful and throughout description of dynamic programming solutions to pattern recognition [88]. Fu's paper postulates that in a decision problem such as pattern recognition, the optimal structure for the statistical recognition system is characterized by a classifier that computes the probability of feature measurements discussion patterns, and assigns the element to a pattern in which the joint probability is a maximum. The observation of measurements is sequential, and after each measurement the classifier must decide whether to *stop* and classify or seek new features. This linear structure is said to be non-optimal. The authors instead presented an optimal classification scheme based on a recursive optimization method that could find the truly optimal *stopping rules* in a number of sequential recognition problems. That is, if an approach is pursuing an optimal set of decisions that include both the choice of stopping or continuing the observation of feature measurements, then as each stage of observation the remaining decisions must themselves form an optimal sequence from that stage reached to the terminal stage of the process. In all, every choice made must always be the *best* one.

For a better explanation of the dynamic programming, some definitions will be given first in the following:

Let $\{Q : q_0, q_1, \cdots, q_{N_s+1}\}$ be a set of observation sequence. In the character segmentation problem, q_i is the ith segmentation path extracted. N_s is the total number of the segmentation paths. All the segmentation paths are sorted by their x coordinates of the mass center $x_{mass}(q_i)$. Let $\{[x_j(q_i), y_j(q_i)], j = 0, 1, \cdots, L(q_i)\}$ are the coordinates of all the pixels on the segmentation path q_i.

We have

$$x_{mass}(q_i) = \frac{\sum_{j=0}^{L(q_i)} x_j(q_i)}{L(q_i)} \tag{5.17}$$

Among $q_i, i = 0, 1, \cdots, N_s+1$, q_0 is the left edge of the image and q_{N_s+1} the right edge of the image. $Dis(q_i, q_j)$ is a function to measure the distance between the segmentation paths q_i and q_j, which is defined as

$$Dis(q_i, q_j) = |x_{mass}(q_i) - x_{mass}(q_j)| \tag{5.18}$$

$\{\tilde{Q}_j : q_l\}$ is a set of segmentation paths on the right side of segmentation path q_j, that is, for $\forall s\ [x_s(q_j), y_s(q_j)]$ and $\forall t\ [x_t(q_l), y_t(q_l)]$, if $y_s(q_j) = y_t(q_l)$, then $x_s(q_j) < x_t(q_l)$.

Let $C^j_{n_1,n_2}$ denote the segments between segmentation paths q_{n_1} and q_{n_2}. We define a composition set $\{C : c^1, c^2, \cdots, c^{L^c}\}$, where L^c is the number of the segments. Obviously, C is what we want to identify. However, there might be more than one output for the algorithm. Therefore we define a set, $\{C_m, m = 1, 2, \cdots, M\}$, as the outputs of the algorithm, where M is the number of composition sets. Let c^i_m denote the ith segment in the mth composition set and L^c_m the number of segments in the mth composition set. The confidence of composition set C_m, $f_{conf}(C_m)$, can be determined by

$$f_{conf}(C_m) = \sqrt[L^c_m]{O(c^1_m) \times O(c^2_m) \cdots O(c^{L^c_m}_m)} \times m_{L^c_m} \tag{5.19}$$

where $O(c^i_m)$ is the classifier output of segment c^i_m. $m_{L^c_m}$ is the output of the length estimation algorithm discussed in Chapter 3, which represents the grade of the digit string having length L^c_m.

Figures 5.5.1 and 5.5.1 show the procedure of using a dynamic programming technique adapted from Fu's theory of dynamic programming [88] to segment a digit string of unknown length. A brief discussion is given as follows.

- T_{d1} in Fig. 5.5.1 is a threshold to control the search depth. In experiments, we set $T_{d1} = N_h$, where $(N - h)$ is the height of the normalized image.

Table 5.4: Experimental results of the dynamic programming based segmentation and recognition of handwritten digit strings with unknown length (ANCS: Average Number of Composition Sets).

Data set	total	ANCS	rejected	correct	recognition rate (%)
2-digit strings	4555	2.6	13	4186	91.9
3-digit strings	355	8.5	3	299	84.2
4-digit strings	48	27.6	0	36	75.0
all	4958	3.3	16	4521	91.2

- T_{d2} in Fig. 5.5.1 is another threshold that is used to control the search depth. We set $T_{d2} = 2N_h$.

- T_{d3} in Fig. 5.5.1 is a threshold used to check whether the searching can be terminated. We set $T_{d3} = N_h/6$.

- In Fig.5.5.1, The 'result is acceptable?' test is to determine whether the output of the classifier is acceptable. The 'satisfactory digit found?' test in Fig. 5.5.1 is to check whether the loop procedure shown in Fig. 5.5.1 has found any acceptable digits.

5.5.2 Experimental Results

As introduced in Section 1.3, we have 4,555 connected 2-digit strings, 355 connected 3-digit strings and 48 connected 4-digit strings. We used the multi-module classifier discussed in Chapter 4.

Table 5.4 shows the experimental results of the dynamic programming based segmentation and recognition of connected digits with unknown length. In the table, ANCS stands for the Average Number of Composition Sets. We can see from the table that with an increased string length, the ANCS increases dramatically and the recognition rate decreases quite a bit. Although the number of 3-digit and 4-digit strings is much smaller than that of 2-digit strings, the

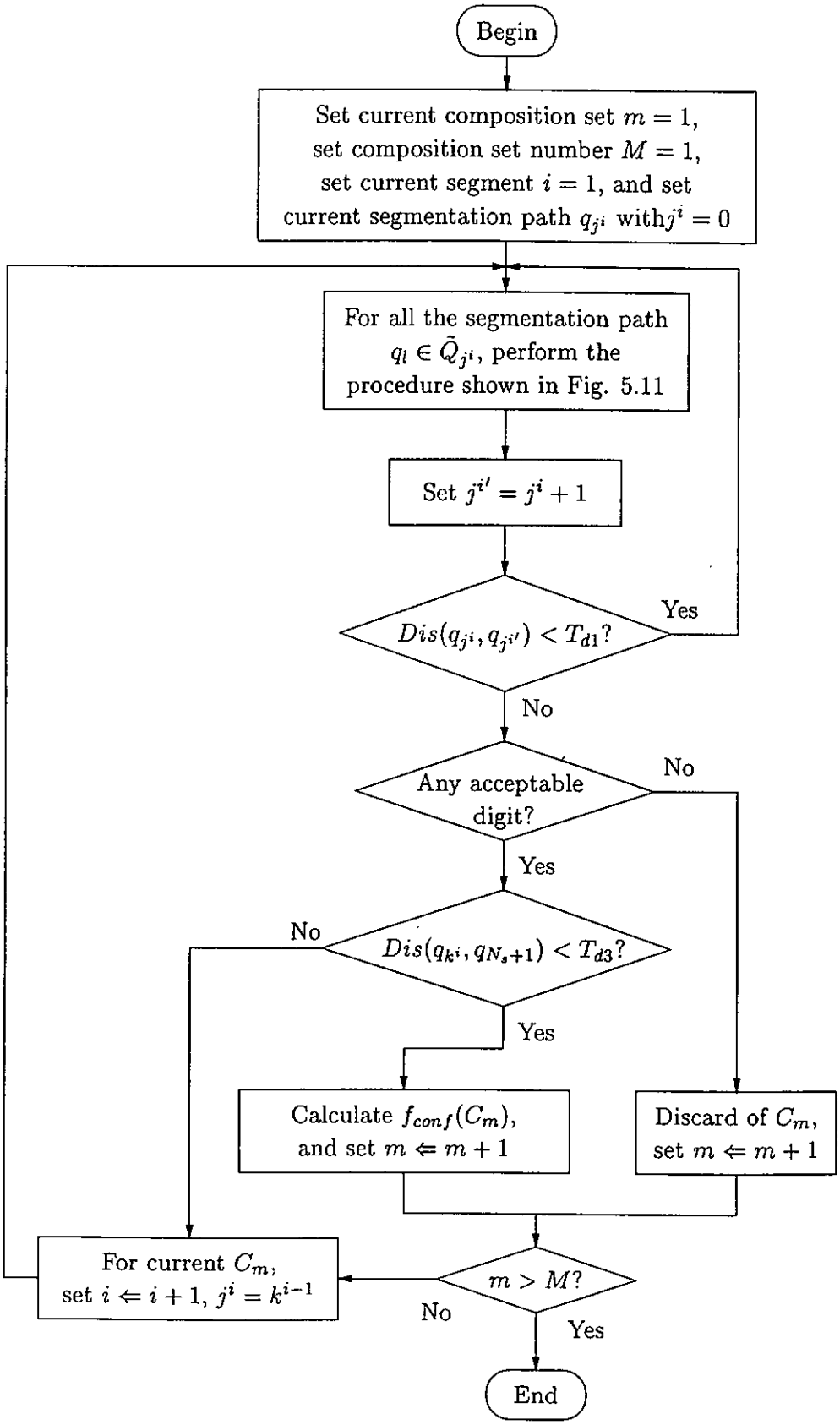


Figure 5.10: Flowchart of the dynamic programming algorithm for segmentation-based handwritten digit recognition.

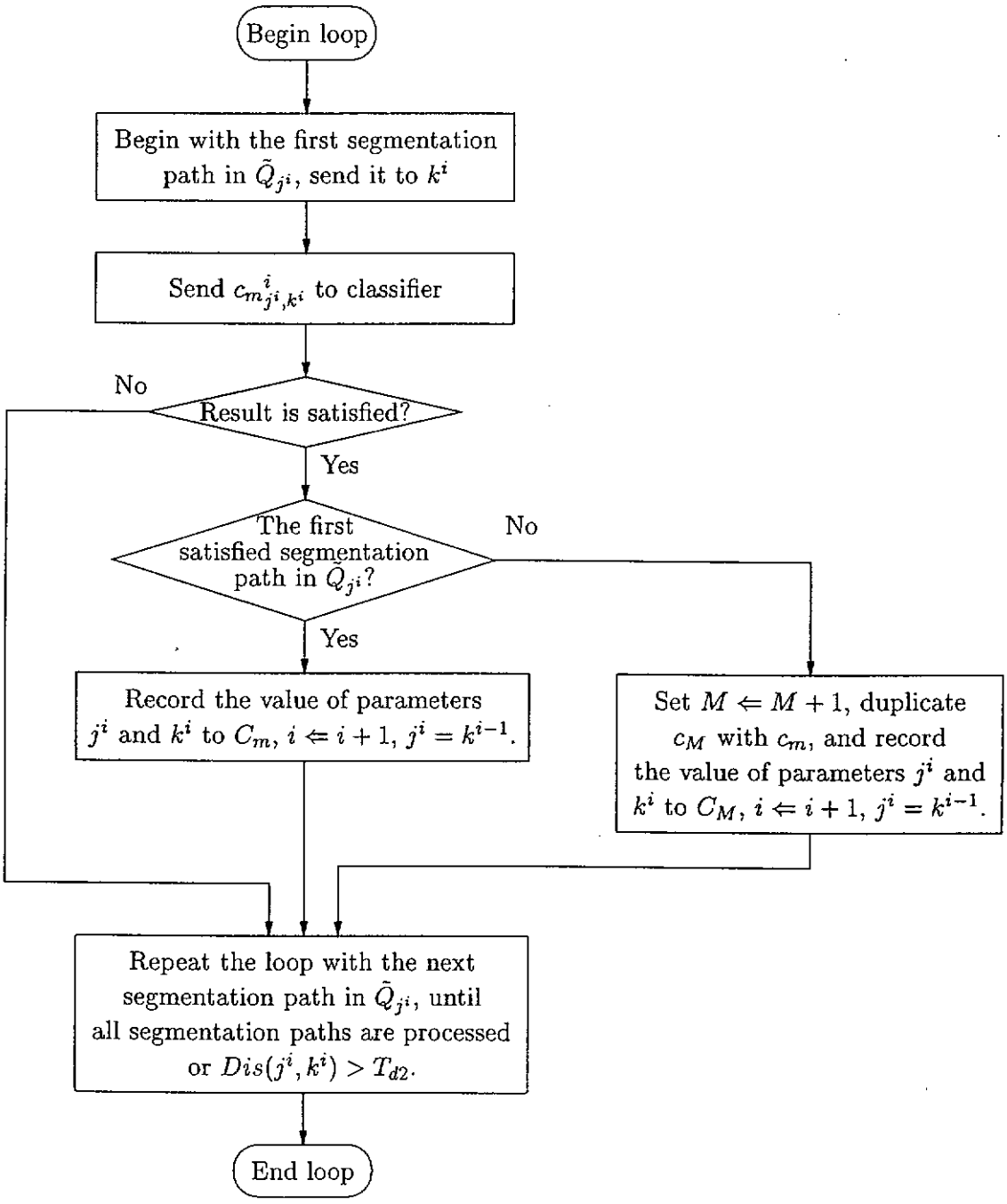


Figure 5.11: Flowchart of the loop procedure shown in Fig. 5.5.1.

Table 5.5: Error analysis of the dynamic programming based segmentation and recognition of handwritten digit strings of unknown length.

Data set	total	correct length		incorrect length	
		number	rate (%)	number	rate (%)
2-digit strings	369	232	62.87	137	37.13
3-digit strings	56	35	62.50	21	37.50
4-digit strings	12	6	50.0	6	50.0
all	437	273	62.47	164	37.53

Table 5.6: A comparison of different classifiers on the segmentation-based recognition of handwritten digit strings.

Classifier	rejection rate (%)	recognition rate (%)
Multi-Module classifier	0.32	91.2
Yan's optimized prototypes	1.36	84.3
MLP (64-100-10)	5.32	68.4

sample distribution does reflect possibilities of digit strings of different lengths in real world applications. The overall recognition rate of 91.2% is a fair indication of the system recognition performance. We also notice that a few samples were rejected because the algorithm could not recognize these digit strings with a high confidence level.

Table 5.5 shows the error analysis of our segmentation and recognition algorithm. All the mis-recognized samples are categorized into two sets: the samples with correct length estimation and the samples with incorrect length estimation. We notice that the former has more patterns misclassified (account for 62.47%). Including some misclassification on the latter set, most of the errors ($> 62.47\%$) are caused by poor classification. Another interesting finding is that with the increased length from two digits to four digits, the error rate in the set of incorrect length estimation increases from 37.13% to 50.0%. That suggests that the effect of length mis-estimation increases more than that of classifier misclassification as digit strings become longer.

Figure 5.12: An example of 4-digit string with a pruned background skeleton.

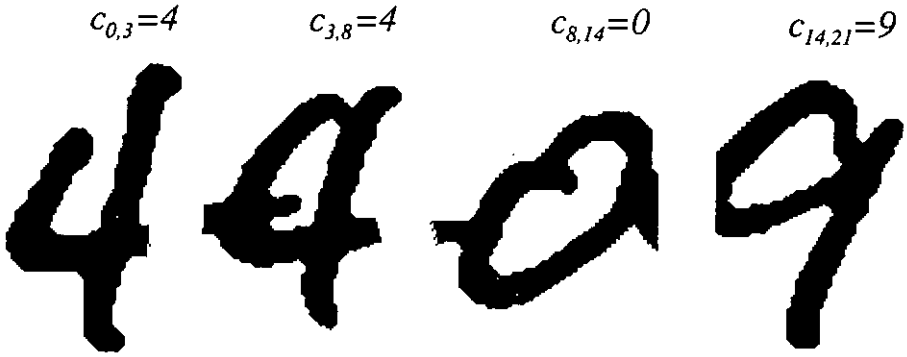


Figure 5.13: Winning recognition composition set.

A comparison of using different classifiers on the segmentation and recognition performance is presented in Table 5.6, which is an extension to the work discussed in Chapter 4. We have applied three classifiers, our multi-module classifier, Yan's optimized nearest neighbor classifier, and a 64-100-10 MLP. Experimental results show that our classifier can achieve the best performance on recognition of connected digit strings. We can also notice that the differences of the performance between these techniques are greater than those for isolated digit recognition reported in Section 4.3, which suggests that the differences of performance have been amplified. In other words, a small improvement on recognition of isolated digits may greatly improve the recognition performance of digit strings.

Figure 5.12 shows an example of 4-digit string with a pruned background skeletons. Figure 5.13 shows the winning composition set $\{C_w$:

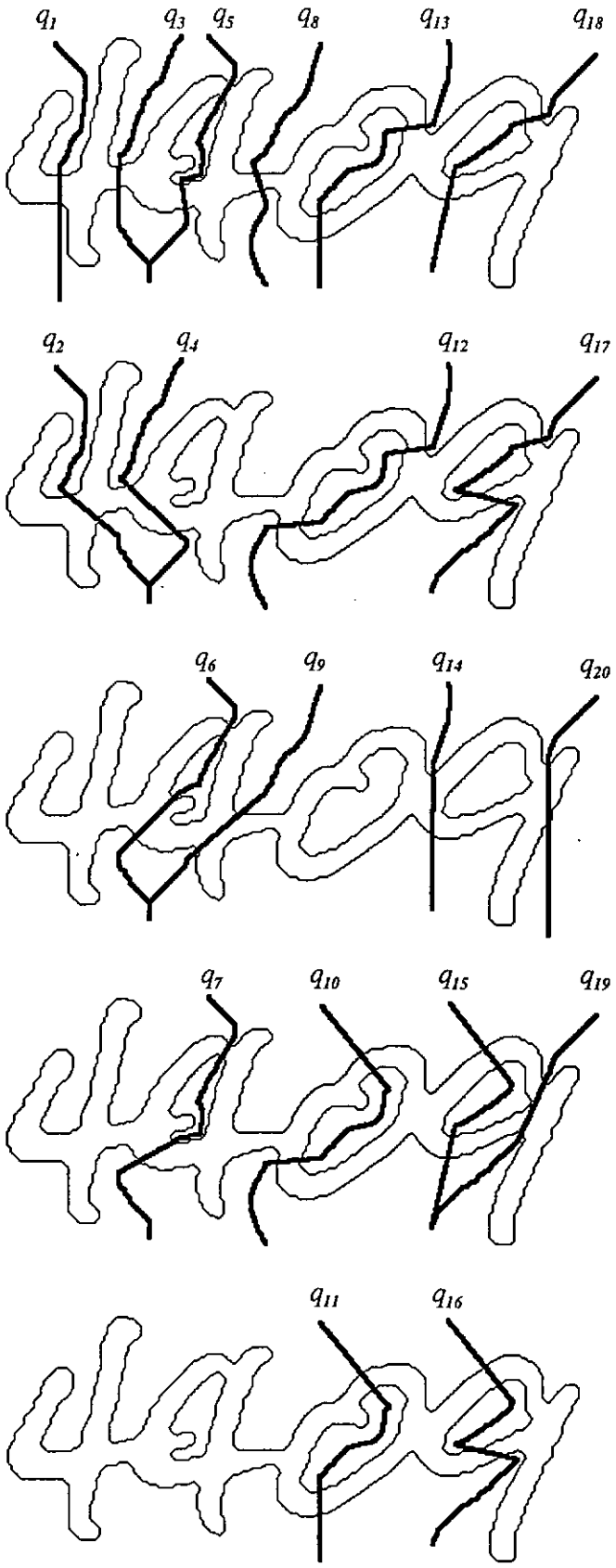


Figure 5.14: All the segmentation paths on a binary digit string image.

$c_{0,3}, c_{3,8}, c_{8,14}, c_{14,21}\}$, a correct segmentation. Figure 5.14 depicts all the segmentation paths $\{q_1, q_2, \cdots, q_{20}\}$ that identified by using the feature point matching procedure discussed in Section 5.3.

As mentioned in Chapter 4.3, the computational cost of the multi-module classifier is very high. In our experiments, recognizing a 4-digit string takes about half of an hour.

5.6 Concluding Remarks

A segmentation-based recognition approach based on the background (regions excluding the characters) skeleton of a digit string image is presented in this chapter. In the segmentation, a set of feature points on the background skeleton of a digit string image are used to trace all possible segmentation paths based on a three-step searching process. To compare our approach with other segmentation algorithms, we applied our segmentation algorithm to handwritten two-digit strings with fuzzified decision rules and Yan's optimized nearest neighbor classifier. Experimental results on NIST Special Database 3 show that our technique can successfully separate a large proportion of connected handwritten two-digit strings of single- or double-touching with a performance that is compared favorably with those of other techniques tested. We also tested our segmentation algorithm on digit strings of unknown length with a dynamic programming technique. Experimental results show that our approach can achieve a recognition rate of 91.2%. However, the computational cost is a main obstacle for applying our approach to real-world applications.

Chapter 6

Segmentation-Free Recognition of Digit Strings

Experimental results show that the segmentation algorithm presented in Chapter 5 cannot find all the possible segmentation paths. Some true segmentation paths might not be identified. Figure 6.1 is an example of 2-digit string that can not be correctly separated. We can see that the digit string are two 0s connected by a ligature at top of them. There should be two correct segmentation paths. However, our segmentation algorithm can only identify one of them, so the digit string is rejected by the segmentation-based recognition algorithm. Such a problem cannot be totally avoided in a segmentation-based recognition technique.

To overcome this shortcoming, we have developed a segmentation-free algorithm. A combination of a segmentation-based algorithm and a segmentation-free algorithm is expected to achieve a better recognition performance.

The segmentation-free recognition approach we present here is similar to the segmentation-based recognition algorithm introduced in Chapter 5. The main difference is that we do not apply a segmentation technique in this segmentation-free approach. The main idea of the algorithm is to apply a classifier on a rectangle window (variable widths) moving through a digit string image. Same

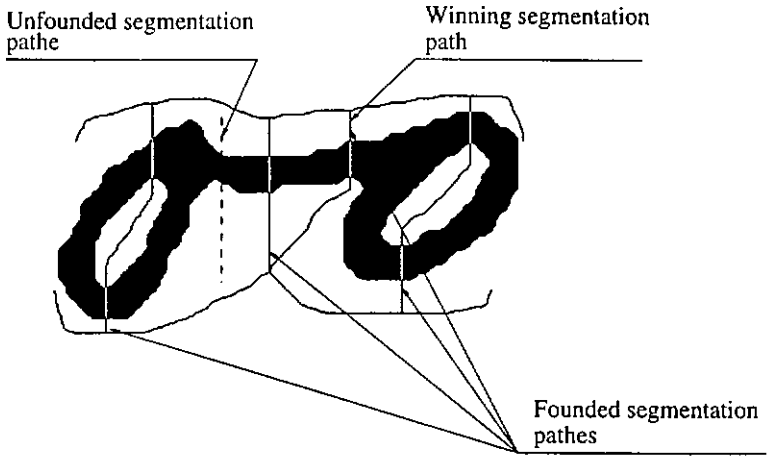


Figure 6.1: An example of 2-digit string that can not be separated by the segmentation algorithm presented in Chapter 5.

as the segmentation-based approach discussed in Chapter 5, we apply a dynamic-programming technique to find the best recognition path.

6.1 Moving Windows

As introduced in Chapter 2, input digit strings are first normalized. Let the height and width of the normalized images be N_h and N_w respectively. We choose the width and height of window W_w and W_h based on N_h and a moving step denoted by W_s. We set

$$W_h = N_h \tag{6.1}$$

The widths of nine windows $\{W_w^i, i = 0, 1, \cdots, 8\}$ are given by

$$W_w^i = (2 + i)W_s \qquad i = 0, 1, \cdots, 8 \tag{6.2}$$

with

$$W_s = \frac{W_h}{8} \tag{6.3}$$

Figure 6.2 shows moving windows applied to a 4-digit string.

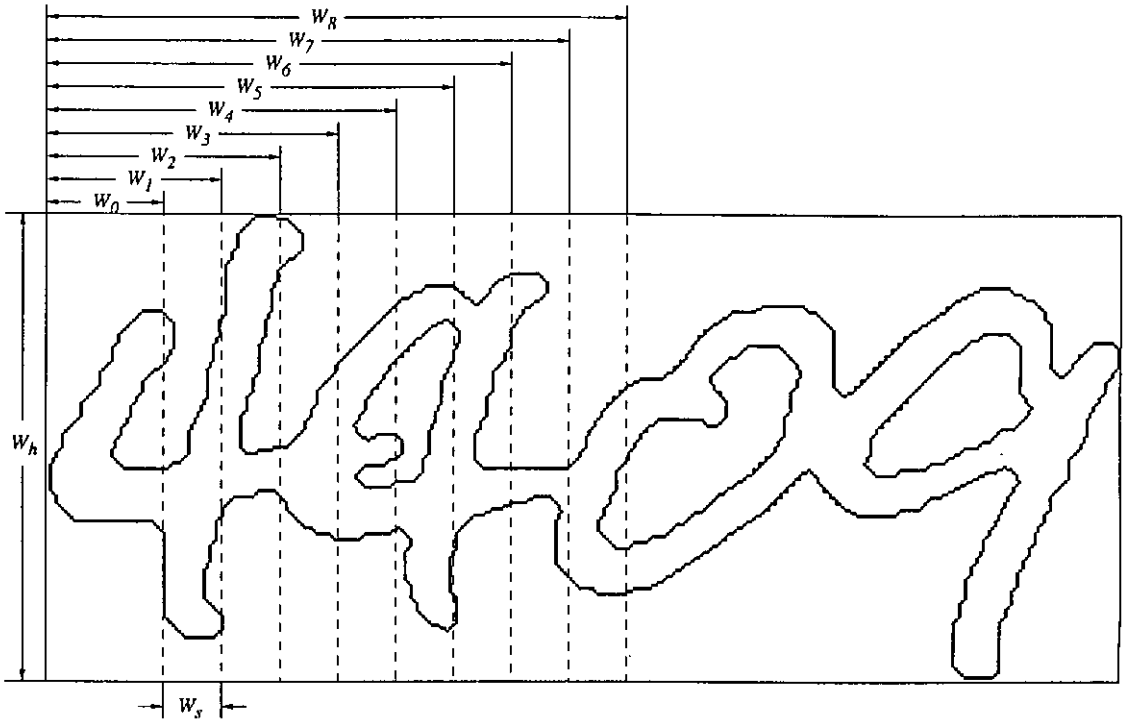


Figure 6.2: Moving windows applied to a 4-digit string.

6.2 Modified Classifier

As shown in Fig. 6.2, it is unavoidable that the parts of neighboring digits are included in the windows, the classifier introduced in Chapter 4, which is primarily designed for isolated digit classification, cannot be applied to this moving window approach without some necessary changes. The changes in the number of modules, feature extraction, and template matching procedure, are discussed in the following sections.

6.2.1 Changes in Classification Modules

Figure 6.3 shows the modified multi-module classifier used in the segmentation-free approach. Compared with Fig. 4.1, we notice that the structure-feature based classifier is removed. The reason is that the connecting structures of neighboring digits degrades the performance of the structure feature based classifier significantly.

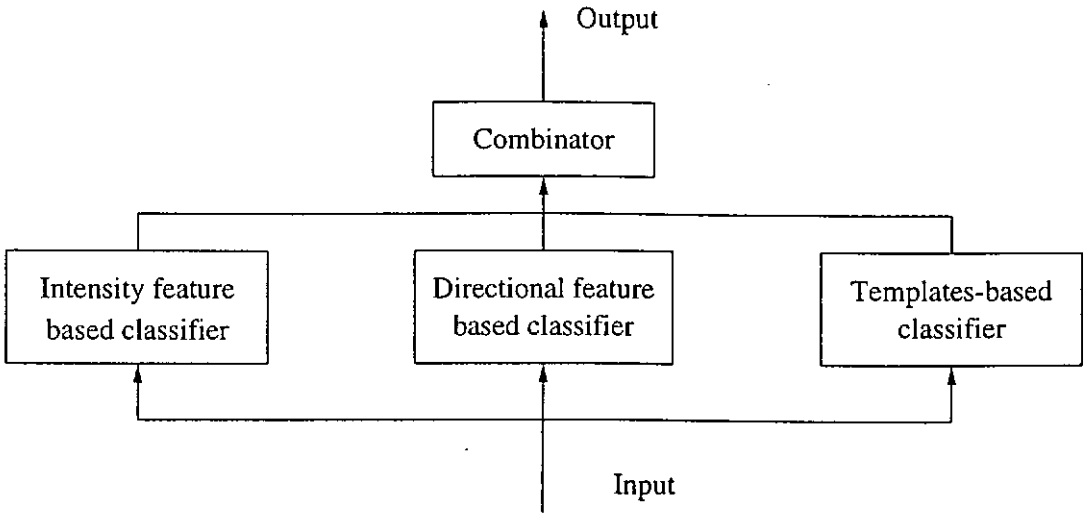


Figure 6.3: The modified multi-module classifier used in the segmentation-free recognition approach.

6.2.2 Masking in the Features Spaces

One changes in feature extraction is that we apply a mask to the original 160×160 feature spaces of the intensity feature based classifier and the directional feature based classifier. Let $f(x, y)$ ($-1.0 \leq x \leq 1.0$ and $-1.0 \leq y \leq 1.0$) be the original feature domain, and $m(x, y)$ be a mask, the masked feature space $\check{f}(x, y)$ is given by

$$\check{f}(x, y) = f(x, y) \times m(x, y) \tag{6.4}$$

where mask $m(x, y)$ must meet the following constraint:

$$\frac{[\cos(\alpha)x + \sin(\alpha)y]^2}{\gamma_x^2} + \frac{[-\sin(\alpha)x + \cos(\alpha)y]^2}{\gamma_y^2} + \frac{m^2(x, y)}{\gamma_z^2} = 1.0 \tag{6.5}$$

As shown in Equation 6.5, mask $m(x, y)$ is an ellipsoid with x-radius γ_x, y-radius γ_y, and x-radius γ_z. Considering the slant of handwritten digit, there is a rotation angle of α to z axis applied. In our experiments, we set $\gamma_x = 1.0$, $\gamma_y = 2.0$, $\gamma_z = \sqrt{3.0}$, and $\alpha = \pi/4$. Figure 6.4 shows the mask in gray scale image. We can see that the center part of the feature space is enhanced and the influence from neighboring digits is weakened.

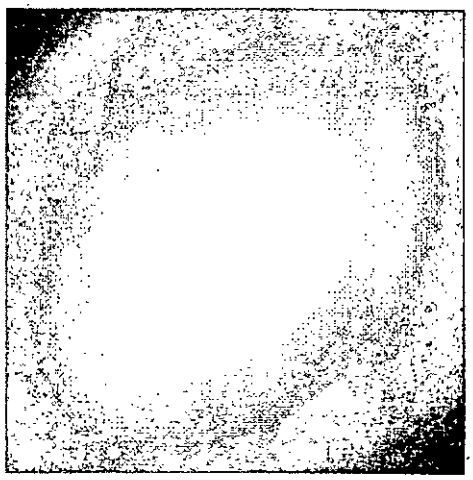


Figure 6.4: The feature mask in grey scale.

6.2.3 Modified Template-Based Classifier

To reduce the influence from the neighboring digits, instead of directly using Equation 4.21 for matching, some processing is performed. A normalization method is first applied. The maximal and minimal y values of the foreground, y^{max} and y^{min}, is found in the region $\frac{W_w}{5} < x < \frac{4W_w}{5}$. In the mean time, y_j^{max} and y_j^{min} is found in the region of template j that has $S_j(x,y) \geq 0.5$. The window is then scaled and translated so that $y^{max} = y_j^{max}$ and $y^{min} = y_j^{min}$. A change is also made to Equation 4.21. Instead of the whole image, the matching is only performed in the region $\Omega = \{S_j(x,y) \geq \theta\}$, where θ is a threshold within [0, 1]. Therefore Equations 4.23 and 4.22 are replace by

$$\phi_{2,j} = \frac{\oint\limits_{\Omega} \cos^2 [\beta_j(x,y)]\, dx\, dy}{\oint\limits_{\Omega} dx\, dy} \tag{6.6}$$

and

$$v_j = \frac{\oint\limits_{\Omega} [S_j(x,y) - g(x,y)]^2\, dx\, dy}{\oint\limits_{\Omega} g^2(x,y)\, dx\, dy} \tag{6.7}$$

In experiments, we set $\theta = 0.25$.

Table 6.1: Experimental results of the segmentation-free recognition of handwritten digit strings with unknown length (ANCS: Average Number of Composition Sets).

Data set	Total	ANCS	Rejected	Correct	Recognition rate (%)
2-digit strings	4555	4.8	34	3783	83.1
3-digit strings	355	19.3	7	274	77.2
4-digit strings	48	47.2	1	30	62.5
all	4958	6.3	42	4087	82.4

6.3 Dynamic Programming Algorithm for Segmentation-Free Approach

The dynamic programming algorithm used in this segmentation-free approach is very similar to that in the segmentation-based approach discussed in Chapter 5. The only difference is the segmentation paths $\{Q : q_0, q_1, \cdots, q_{N_s+1}\}$ are replace by a set of steps $\{W : w_0, w_1, \cdots, w_{N_{step}+1}\}$, where $N_{step} = N_w/N_h$. Same as Q, w_0 and $w_{N_{step}+1}$ are the left and right edges of the image, respectively.

6.4 Experimental Results Using Segmentation-Free Algorithm

The experimental results of the segmentation-free recognition of handwritten digit strings with unknown length are shown in Table 6.1. Compared with Table 5.4 for the segmentation-based recognition, the performance of the segmentation-free recognition is much lower. However, the computational requirement for the segmentation-based approach is much higher.

An error analysis of the segmentation-free recognition on connected digit strings with unknown length are shown in Table 6.2. Compared with Table 5.5 for the segmentation-based approach, the ratio of the number of wrong classified

Table 6.2: Error analysis of the segmentation-free recognition of handwritten digit strings with unknown length.

Data set	total	correct length		incorrect length	
		number	rate (%)	number	rate (%)
2-digit strings	738	539	73.04	199	26.96
3-digit strings	74	45	60.81	29	39.19
4-digit strings	17	10	58.82	7	41.18
all	829	594	71.65	235	28.35

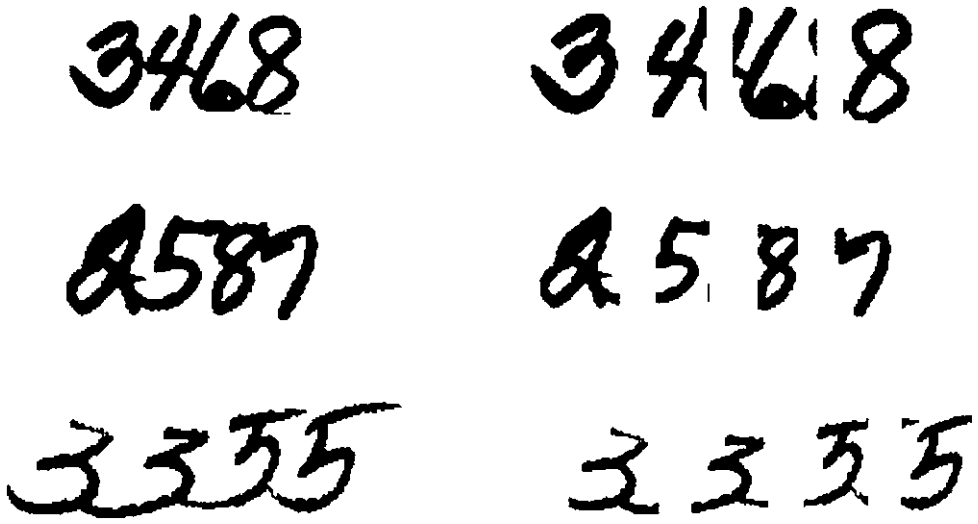
Table 6.3: A comparison of different classifiers on the segmentation-free recognition of handwritten digit strings.

Classifier	rejection rate (%)	recognition rate (%)
Multi-Module classifier	0.85	82.4
Yan's optimized prototypes	3.03	76.5
MLP (64-100-10)	9.12	59.5

the samples with correct length estimation to that with incorrect length estimation are higher, which suggests that the performance of classifier play a more important role on the segmentation-free approach.

Same as in Chapter 5, a further comparison of the performance of different classifiers for the segmentation-free approach is given in Table 6.3. We can see that the performance of Yan's optimized prototypes and the 64-100-10 MLP classifier degrades more seriously than that of our multi-module classifier (refer to Table 5.6). This has again demonstrated that our multi-module classifier, which is discussed in Chapter 5, is more reliable.

Some correctly recognized and mis-recognized (rejected or un-recognized) 4-digit strings are shown in Figs. 6.5 and 6.6, respectively. In Fig. 6.6, the sample on the left is not correctly recognized because the writing style of the sample is not included in the training set. The sample on the right is due to serious degradation of the image.



(a) (b)

Figure 6.5: Examples of correctly recognized 4-digit strings, (a) original 4-digit strings; (b) separated digits.

Figure 6.6: Examples of mis-recognized 4-digit strings.

6.5 Concluding Remarks

A segmentation-free recognition approach is presented in this chapter in order to deal with some hard-to-segment connected digit strings. The main idea of the algorithm is to apply a classifier on a rectangle window (variable widths) of a digit string image with the window slided along the string. With the modified multi-module classifier and a dynamic programming technique, the approach can achieve a recognition rate of 82.4% on handwritten digit strings with unknown length.

Chapter 7

Conclusions

This thesis presents our research on segmentation and recognition of handwritten digit strings, which is of many potential applications but a very challenging problem.

7.1 Summary of Contributions

Our contributions can be summarized as follows:

1. A neural network based approach for effective length (the number of digits in a string) estimation of handwritten digit strings was developed. Experimental results show that our approach can achieve reliable estimation and provide very useful information for the successive processing including digit segmentation and recognition.

2. We developed a multi-module classifier to recognize isolated digits. Among four modules, a template-based classifier based on the rational B-spline surface representation of the Pixel-to-Boundary Distance Map (PBDM) was developed to improve the performance of the classifier, in particular, in rejecting non-digit patterns. To extract optimized templates, we used a two-stage algorithm based on a neural network and an evolutionary algorithm. The classifier can reliably distinguish non-digit patterns from digits,

which is a desirable feature for recognizing handwritten digit strings. The classifier has been applied together with a segmentation-based recognition algorithm or a segmentation-free recognition algorithm. Experimental results show that the designed multi-module classifier compares favorably with other classification techniques tested, including an optimized nearest neighbor classifier and a multi-layer perceptron classifier. However, the main limitation of this multi-module classifier is its high computational cost in both storage space and computing time.

3. A new segmentation algorithm based on background-thinning analysis was developed to separate a handwritten digit string into separated digits.

4. Based on the designed multi-module classifier and the background-thinning-based segmentation algorithm, a segmentation-based recognition approach was developed. A dynamic programming algorithm was applied in this approach. Experimental results show that our approach can achieve more favorable classification performance.

5. A segmentation-free recognition approach with a dynamic programming algorithm was also developed for dealing with hard-to-segment handwritten digit strings.

7.2 Future Work

The work discussed in this thesis has set up a solid foundation for a more thorough investigation into the still sparsely studied recognition of connected handwritten characters. Future work will concentrated on some of the following aspects:

- More research should be carried out on fast realization of our algorithms so that they can be applied to real-world problems.

- As presented in this thesis, the performance of the classifier is the key factor to determine the performance of the whole recognition system. More investigations on feature extraction and classifier structure are required for designing a more reliable classifier.

- Develop a system for segmentation and recognition of connected handwritten Chinese characters. Recognition of Chinese characters is very different from the recognition of Latin based characters. New approach may be required to deal with specific problems in recognizing connected handwritten Chinese characters.

Bibliography

[1] C. Y. Suen, M. Berthod, and S. Mori. Automatic recognition of handprinted characters - the state of art. *Proceedings of IEEE*, 68(4):469 – 487, April 1980.

[2] S. Mori, C. Y. Suen, and K. Yamamoto. Historical review of ocr research. *Proceedings of IEEE*, i80(7):1029 – 1058, July 1992.

[3] C. J. C. Burges, J. I. Ben, and J. S. Denker. Off line recognition of handwritten postal words using neural networks. *International Journal of Pattern Recognition and Artificial Intelligence*, 7(4):689 – 704, 1993.

[4] W. Cho, S. W. Lee, and J. H. Kim. Modeling and recognition of cursive words with hidden Markov models. *Pattern Recognition*, 28(12):1941 – 1953, 1995.

[5] K. Aas and L. Eikvil. Text page recognition using grey-level features and hidden Markov models. *Pattern Recognition*, 29(6):977 – 985, 1996.

[6] R. Impedovo, L. Ottaviano, and S. Occhinegro. Optical character recognition - a survey. *International Journal of Pattern Recognition and Artificial Intelligence*, 5(1 & 2):1 – 24, 1991.

[7] Z. Chi, M. Suter, and H. Yan. Handwritten digit recognition using combined id3-derived fuzzy rules and Markov chains. *Pattern Recognition*, 29(11):1821 – 1833, 1996.

[8] A. W. Senior and F. Fallside. Using constrained snakes for feature spotting in off-line cursive script. In *1993 Int'l Conf. Document Analysis and Recognition*, pages 305 – 310, March 1993.

[9] A. W. Senior and A. J. Robinson. An off-line cursive handwriting recognition system. *IEEE trans. on Pattern Analysis & Machine Intelligence*, 20(3), March 1998.

[10] V. K. Govindan and A. P. Shivaprasad. Character recognition - a review. *Pattern Recognition*, 23(6):671 – 683, June 1990.

[11] Q. D. Trier, A. K. Jain, and T. Taxt. Feature extraction methods for character recognition - a survey. *Pattern Recognition*, 29(4):641 – 662, 1996.

[12] S. Lee and J. C. Pan. Unconstrained handwritten numeral recognition based on radial basis competitive and cooperative networks with spatio-temporal features representation. *IEEE Transaction on Neural Networks*, 7(2):455 – 474, Feb. 1996.

[13] S. W. Lee, C. H. Kim, and Y. Y. Tang. Multiresolution recognition of unstrained handwritten numerals with wavelet transform and multilayer cluster neural network. *Pattern Recognition*, 29(12):1953 – 1961, 1996.

[14] D. Cheng and H. Yan. Recognition of handwritten digits based on contour information. *Pattern Recognition*, 31(3):235 – 255, 1998.

[15] H. Yan. Design and implementation of optimized nearest neighbor classifiers for handwritten digit recognition. *Pattern Recognition*, 26, 1993.

[16] T. Wakahara. Shape matching using lat and its application to handwritten numeral recognition. *IEEE trans. on Pattern Analysis & Machine Intelligence*, 16(6), June 1994.

[17] H. Nishida. A structural model of shape deformation. *Pattern Recognition*, 28(10):1611 – 1620, October 1995.

[18] K. W. Cheung, D. Y. Yeung, and R. T. Chin. A unified framework for handwritten character recognition using deformable models. In *Proceedings of Second Asian Conf. Computer Vision*, volume 1, 1995.

[19] M. Revow, C. K. I. Williams, and G. E. Hinton. Using generative models for handwritten digit recognition. *IEEE Transaction on Pattern Analysis and Machine Intelligence*, 18(6):592 - 606, 1996.

[20] Anil K. Jain, Y. Zhong, and Sridhar Lakshmanan. Object matching using deformable templates. *IEEE Transaction on Pattern Analysis and Machine Analysis*, 18(3), Match 1996.

[21] Anil K. Jain and Douglas Zongker. Representation and recognition of handwritten digits using deformable templates. *IEEE Transaction on Pattern Analysis and Machine Analysis*, 19(12), December 1997.

[22] Marie-Pierre Dubuisson-Jolly Anil K. Jain, Yu Zhong. Deformable template models: A review. *Signal Processing*, 71(2):109–129, December 1998.

[23] L. Xu. Methods of combining multiple classifiers and their application to handwriting recognition. *IEEE Trans. on Sys., Man., Cybern.*, 22:418 - 4335, 1992.

[24] T. S. Huang and C. Y. Suen. Combination of multiple experts for the recognition of unconstrained handwritten numerals. *IEEE transaction on Pattern Analysis and Machine Intelligence*, 17(1):90 - 94, 1995.

[25] T. K. Ho, J. J. Hull, and S. N. Srihari. Decision combination in multiple classifier system. *IEEE transaction on Pattern Analysis and Machine Intelligence*, 16(1):66 - 75, 1994.

[26] C. Ji and S. Ma. Combinations of weak classifiers. *IEEE transaction on Neural Networks*, 8(1), Jan. 1997.

[27] K. Woods, W. P. Kegelmeyer, and K. Bowyer. Combination of multiple classifiers using local accuracy estimates. *IEEE transaction on Pattern Analysis and Machine Intelligence*, 19(4), 1997.

[28] G. S. Ng and H. Singh. Data equalization with evidence combination for pattern recognition. *Pattern Recognition Letters*, 19(3), Match 1998.

[29] S.N. Geok and H. Singh. Democracy in pattern classifications: combinations of votes from various pattern classifiers. *Artificial Intelligence in Engineering*, 12(3):189 – 204, July 1998.

[30] Andrew William Senior. *Off-line Cursive Handwriting Recognition using Recurrent Neural Networks.* PhD thesis, University of Cambridge, 1994.

[31] K. M. Sayre. Machine recognition of handwritten words: A project report. *Pattern Recognition*, 5:213 – 228, 1973.

[32] R. E. Ehrick and K. J. Koehler. Experiments in the contextual recognition of cursive scripts. *IEEE transactions on Computers*, 24(2):182 – 194, February 1975.

[33] M. Shridhar and A. Badreldin. Recognition of isolated and simply connected handwritten numerals. *Pattern Recognition*, 1(19):1 – 12, 1986.

[34] M. Shridhar and A. Badreldin. Context-directed segmentation algorithm for handwritten numeral strings. *Image Vision Computing*, 5(1):3 – 9, 1987.

[35] R. M. Bozinovic and S. N. Srihari. Off-line cursive script word recognition. *IEEE transactions on Pattern Analysis & Machine Intelligence*, 11(1):68 – 83, Janurary 1989.

[36] M. Mohammed Beglou, M. J. J. Holt, and S. Datta. Slant indenpendent letters segmentation for cursive script recognition. In *1991 International Workshop on Frontiers of Handwriting Recognition*, pages 375 – 380, Bonas, France, 1991.

[37] R. Nag, K. H. Wong, and F. Fallside. Script recognition using hidden Markov models. In *Proceedings of ICASSP'86*, pages 2071 – 2074, 1986.

[38] A. Kundu and P. Bahl. Recognition of handwritten script: A hidden Markov model based approach. In *Proceedings on ICASSP'88*, pages 928 – 931, 1988.

[39] A. Kaltenmeier, T. Caesar, J. M. Gloger, and E. Mandler. Sophisticated topology of hidden Markov models for cursive script. In *Proceedings of ICDAR'93*, pages 139 – 142, 1993.

[40] M. Y. Chen, A. Kundu, and J. Zhou. Off-line handwritten word recognition using a hidden Markov model type stochastic network. *IEEE transaction on Pattern Analysis and Machine Intelligence*, 16(5):481 – 497, 1994.

[41] M. Mohamed and P. Gader. Handwritten word recognition using segmentation free hidden Markov modeling and segmentation-based dynamic programming technique. *IEEE transaction on Pattern Analysis and Machine Intelligence*, 18(5):548 – 554, 1996.

[42] H. S. Park and S. W. Lee. Off-line recognition of large-set handwritten characters with multiple hidden Markov models. *Pattern Recognition*, 29(2):231 – 244, 1996.

[43] R. Hoch and T. Kieninger. On virtual partitioning of large dictionaries for contextual post-processing to improve character recognition. *International Journal of Pattern Recognition and Artificial Intelligence*, 10(4):273 – 289, 1996.

[44] G. Kim and V. Govindaraju. A lexicon driven approach to handwritten word recognition for real-time applications. *IEEE transaction on Pattern Analysis & Machine Intelligence*, 19(4), 1997.

[45] Y. Lu. Machine printed character segmentation – an overview. *Pattern Recognition*, 28(1):67 – 80, 1995.

[46] Y. Lu and M. Shridhar. Character segmentation in handwritten words - an overview. *Pattern Recognition*, 29(1):77 – 96, 1996.

[47] M. Cheriet, Y. S. Huang, and C. Y. Suen. Background region-based algorithm for the segmentation of connect digits. In *Proc. 11th Int. Conf. on Pattern Recognition*, pages 619 – 622, Sept. 1992.

[48] D. Yu and H. Yan. Separation of touching handwritten numeral strings based on structural analysis. In *Proceedings of Real World Computing Symposium'97*, pages 238 – 245, 1997.

[49] N. W. Strathy, C. Y. Suen, and A. Krzyzak. Segmentation of handwritten digits using contour features. In *Proceedings of ICDAR'93*, pages 577 – 580, 1993.

[50] P. D. Gader, M. P. Whalenand M. J. Ganzberger, and D. Hepp. Handprinted word recognition on a nist data set. *Machine Vision and Its Applications*, 8:31 – 40, 1995.

[51] A. Pervez and C. Y. Suen. Segmentation of unconstrained handwritten numeric postal zip codes. In *Proceedings of the 6th International Conference on Pattern Recognition*, pages 545 –547, 1990.

[52] C. B. Bose and S. S. Kuo. Connected and degraded text recognition using hidden Markov model. *Pattern Recognition*, 27(10):1345 – 1363, 1994.

[53] M. Schenkel and M. Jabri. Low resolution, degraded document recognition using neural networks and hidden markov models. *Pattern Recognition Letters*, 19(3-4), Match 1998.

[54] J. Rocha and T. Pavlidis. A solution to the problem of touching and broken characters. In *Proceedings of ICASSP'93*, pages 602 – 605, 1993.

[55] J. Rocha and T. Pavlidis. A shape analysis model with applications to a character recognition system. *IEEE transaction on Pattern Analysis and Machine Intelligence*, 16(4):393 – 404, 1994.

[56] J. Rocha and T. Pavlidis. Character recognition without segmentation. *IEEE transaction on Pattern Analysis and Machine Intelligence*, 17(9):903 – 909, 1995.

[57] Didier Guillevic. *Unconstrained Handwriting Recognition Applied to the Processing of Bank Cheques.* PhD thesis, Concordia University, 1995.

[58] C. Y. Suen. Distinctive features in automatic recognition of handprinted characters. *Signal Processing*, 4(2 & 3):193 – 207, 1982.

[59] J. Serra. *Image Analysis and Mathematical Morphology.* Academic Press, New York, 1983.

[60] R. M. Haralick, S. R. Stenberg, and X. Zhuang. Image analysis using mathematical morphology. *IEEE transaction on Pattern Analysis and Machine Intelligence*, 9(4):532 – 550, 1987.

[61] J. Song and E. J. Deep. The analysis of morphological filters with multiple structuring elements. *Computer Vision, Graphics and Image Processing*, 50:308 – 328, 1990.

[62] R. C. Gonzalez and R. E. Woods. *Digital Image Processing.* Addison-wesley Publishing Company, Boston, 1992.

[63] W. K. Pratt. *Digital Image Processing.* John Wiley & Sons, New York, 1991.

[64] R. Fenrich. Segmentation of automatically located handwritten numeric strings. In *From Pixels to Features III: Frontiers in Handwriting Recognition*, pages 47 –59. Elsever Science, 1992.

[65] M. D. Garris and R. A. Wilkinson. Nist speical database 3 handwritten segmented characters. Technical report, National Institute of Standards and Technology (NIST), 1992.

[66] K. J. Versprille. *Computer-Aided Design Applications of The Rational B-spline Approximation Form.* PhD thesis, Syracuss University, 1975..

[67] K. F. Lai and R. T. Chin. On modelling, extraction, detection and classification of deformable contours from noisy images. *Image and Vision Computing*, 16(1), Jan. 1998.

[68] Horace H. S. Ip and D. Shen. An affine-invariant active contour model (ai-snake) for model-based segmentation. *Image and Vision Computing*, 16(2), Feb. 1998.

[69] H. Yan. Handwritten digit recognition using optimized prototypes. *Pattern Recognition Letters*, 15, 1994.

[70] H. A. Hakopian B. D. Bojanov and A. A. Sahakian. *Spline Functions and Multivariate Interpolation.* Kluwer Academic Publisher, London, 1993.

[71] C. M. Fonseca and P. J. Fleming. Multiobjective optimization and multiple constrain handling with evolutionary algorithms part i: A unified formulation. *IEEE transaction on System, Man and Cybernetics, Part A: System and Humans*, 28(1), January 1998.

[72] Thomas Bäck and Hans-Paul Schwefel. An overview of evolutionary algorithms for parameter optimization. *Evolutionary Computation*, 1(1), January 1993.

[73] D. B. Fogel. An introduction to simulated evolutionary optimization. *IEEE transaction on Neural Networks*, 5(1), 1994.

[74] T. Kozek and T. Roska L. O. Chua. Genetic algorithm for cnn template learning. *IEEE transaction on circuits and system - I: Fundamental, Theory and Applications*, 40(6), June 1993.

[75] M. Shackleton. Learned deformable templates for object recognition. In *Proceedings of IEE Colloquium on 'Genetic Algorithm in Image Processing and Vision'*, volume 7, pages 1 – 6, London, UK, 1994.

[76] K. Delibasis and P. Undrill. Genetic algorithm and deformable geometric models for anatomical object recognition. In *Proceedings of IEE Colloquium on 'Genetic Algorithm in Image Processing and Vision'*, volume 8, pages 1 – 7, London, UK, 1994.

[77] J. W. Kim, B. H. Kang, P. M. Kim, and M. S. Cho. Human face location in image sequences using genetic templates. In *Proceedings of 1997 IEEE International Conference on System, Man and Cybernetics*, volume 3, pages 2985 – 2988, 1997.

[78] M. Sarkar, B. Yegnanarayana, and D. Khemani. A clustering algorithm using an evolutionary programming-based approach. *Pattern Recognition Letters*, 18:975 – 986, 1997.

[79] Talcott Parsons. *The Social System.* Routledge, London, 1991.

[80] Z. Chi and H. Yan. ID3-derived fuzzy rules and optimized defuzzification for handwritten numeral recognition. *IEEE Transactions on Fuzzy Systems*, 4(1):24–31, 1996.

[81] Z. Chi, M. Suters, and H. Yan. Separation of single- and double-touching handwritten numeral strings. *Optical Engineering*, 34(4):1159 – 1165, 1995.

[82] J. M. Westall and M. S. Narasimha. Vertex directed segmentation of handwritten numerals. *Pattern Recognition*, 26(10):1473 – 1486, 1993.

[83] L. A. Zadeh. Fuzzy sets. *Information and Controls*, 8, 1965.

[84] Zheru Chi, Hong Yang, and Tuan Pham. *Fuzzy Algorithms: with Applications to Image Processing and Pattern Recognition.* World Scientific, Singapore, 1996.

[85] J. R. Quilan. Introduction of decision trees. *Machine Learning*, 1(1), 1986.

[86] H. Fujisawa, N. Yasuaki, and K. Kurino. Segmentation methods for character recognition: from segmentation to document structure analysis. *Proc. IEEE*, 80(7):1079 – 1092, 1992.

[87] Z. Zhao, M. Suters, and H. Yan. Connected handwritten digit separation by optimal contour partition. In *Proc. DICTA-93 Conference on Digital Image Computing: Techniques and Applications*, pages 786 – 793, 1993.

[88] King Sun Fu, Y. T. Chien, and G. Cardillo. A dynamic programming approach to sequential pattern recognition. *IEEE transactions on Electronic Computers*, EC16, December 1967.