A background-thinning-based approach for separating and recognizing connected handwritten digit strings

Article *in* Pattern Recognition · June 1999

DOI: 10.1016/S0031-3203(98)00123-X · Source: DBLP

4 authors, including:

Zheru Chi
The Hong Kong Polytechnic University
191 PUBLICATIONS **2,765** CITATIONS
SEE PROFILE

Wan-Chi Siu
The Hong Kong Polytechnic University
527 PUBLICATIONS **4,756** CITATIONS
SEE PROFILE

Pengfei Shi
Shanghai Jiao Tong University
88 PUBLICATIONS **1,279** CITATIONS
SEE PROFILE

A Background-thinning-based Approach for Separating and Recognizing Connected Handwritten Digit Strings

Zhongkang Lu†, Zheru Chi†, Wan-Chi Siu† and Pengfei Shi‡

†Department of Electronic Engineering

The Hong Kong Polytechnic University

Hung Hom, Kowloon, Hong Kong

‡Institute of Pattern Recognition & Image Processing

Shanghai Jiaotong University

Shanghai, 200030, P. R. of China

Abstract

Most algorithms for segmenting connected handwritten digit strings are based on the analysis of the foreground pixel distributions and the features on the upper/lower contours of the image. In this paper, a new approach is presented to segment connected handwritten two-digit strings based on the thinning of background regions. The algorithm first locates several feature points on the background skeleton of a digit image. Possible segmentation paths are then constructed by matching these feature points. With geometric property measures, all the possible segmentation paths are ranked using fuzzy rules generated from a decision-tree approach. Finally, the top ranked segmentation paths are tested one by one by an optimized nearest neighbor classifier until one of these candidates is accepted based on an acceptance criterion. Experimental results on NIST special database 3 show that our approach can achieve a correct classification rate of 92.5% with only 4.7% of digit strings rejected, which compares favorably with the other techniques tested.

Keywords: Character recognition, Character segmentation, Connected digit strings, Thinning, Fuzzy rules.

1 Introduction

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 [1] in 1995, while he and Shridhar have also written recently on on various approaches for segmenting handwritten characters [2].

Cheriet *et al* [3] proposed a region-based method by which a pair of background regions were identified by independent top-down and bottom-up matchings and a segmentation path was constructed by connecting the matched regions. If no pair of matched regions were found, a vertical segmentation path was constructed either upward from a lower region or downward from an upper region. Strathy *et al* [4] introduced a segmentation algorithm based on contour structure features. In their algorithm, the contour chains of a binary digit image were analyzed and high curvature points were identified. A segmentation path was assumed to pass a pair of high curvature points. Nine features were computed for each segmentation path and used to sort all the possible paths. In our previous work, we also developed a contour curvature based algorithm for separating single- and double-touching handwritten digit strings [5]. In our algorithm, all segmentation path candidates were sorted by a set of features and 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 [6]. Westall and Narasimha presented a vertex directed segmentation algorithm [7]. The algorithm first identified the vertices that were formed by converging vertically oriented edges of adjacent strokes. A segmentation path was 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 to reduce the rejection rate while maintaining a reasonable good recognition rate.

insert Figure 1

In this paper, a new segmentation approach is presented based on the analysis of the background (regions excluding the characters) skeleton of a digit string image. Figure 1 shows the flowchart of our background-thinning-based digit segmentation approach. The paper is organized as follows. In the next section, we briefly explain the forms that two digits are connected. Feature point extraction is discussed in Section 3. In Section 4, we show how segmentation paths are formed by matching feature points. In Section 5, we present a ranking method of using fuzzy rules for ordering segmentation path candidates. Experimental results of our approach on the connected handwritten digit strings extracted from NIST special database 3 are reported and discussed in Section 6. Finally, a conclusion is drawn in Section 7.

2 Types of the Connections in Handwritten Digit Strings

Normally, the connections of adjacent handwritten digits can be categorized into three types [7]:

- 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
- two strokes from two adjacent digits touching end to end as shown in Fig. 2.

insert Figure 2

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. 2. 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. 2. The dotted

line in each case represents a good segmentation path which 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.

3 Feature Point Extraction

In order to get a complete background skeleton, each digit image should be 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 [8]. 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. 3(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. 3(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. 3(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. 4.

- *Fork point:* a point on a segment which has more than two connected branches, such as the points marked by $\bullet$ shown in Fig. 4.

- *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. 4. 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.

- *Corner point:* a point on a segment where the direction of the line changes sharply, such as the points marked by $\circ$ shown in Fig. 4.

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

insert Figure 3

insert Figure 4

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 $\alpha_{\text{pre},l}$ and $\alpha_{\text{post},l}$, which should be between $-\pi$ and $+\pi$. We define

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

$$\alpha_{\text{post},l} = \text{direction}\{p_l, p_{l+1}, \cdots, p_{l+N-1}, p_{p+N}\} \tag{2}$$

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:

$$\alpha_l = \begin{cases} D & : \text{ if } \quad D \leq \pi \\ 2\pi - D & : \text{ if } \quad D > \pi \end{cases} \tag{3}$$

where $D = \left|\alpha_{\text{post},l} - \alpha_{\text{pre},l}\right|$. 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. 4.

4 Construction of Segmentation Paths by Matching Feature Points

insert Figure 5

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 image. A three-step searching scheme shown in Fig. 5 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 exist unmatched feature points on lower segments, it requires to conduct a bottom-up searching from these points 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 together into a segmentation path. As a result, all possible segmentation paths can be identified by this searching scheme.

4.1 Single-direction Search

insert Figure 6

Figure 6 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:

insert Figure 7

- The coordinates of a feature point to be matched, (x, y), should 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. 7(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 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. 7(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. 7(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. 7(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. 7(a).
- There exists one background-foreground transition in the straight line connecting the two points.

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 $y \leq y_{curr} - y_{zone}$.

4.2 Search in Two Directions

insert Figure 8

Figure 8 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. 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 judge the search direction. After the searching, two searched segments and the segment between the unmatched feature points are integrated into a segmentation path. As Example 2 shown in Fig. 7(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 Ranking Segmentation Paths Using Fuzzified Decision Rules

Since Zadeh introduced the fuzzy set theory in 1965 [9], 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 [10].

In separating and recognizing connected handwritten character recognition, a human being performs much better than a machine. We believe that human beings can perform better partially because they have sufficient knowledge on 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 order the segmentation paths obtained using the method discussed in Section 4. In our approach, the membership degree 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 by the path, which will be discussed next. Ranking of the segmentation paths can be helped to decrease the algorithm complexity and improve the recognition accuracy.

5.1 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 [5], which have been 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:

$$F_0 = \begin{cases} N_1/N_2 & : \quad \text{if} \quad N_1 \leq N2, \\ N_2/N_1 & : \quad \text{otherwise.} \end{cases} \tag{4}$$

  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:

$$F_1 = \frac{L_f}{\max(y_L^{(1)}, y_L^{(2)}) - \min(y_H^{(1)}, y_H^{(2)})} \tag{5}$$

  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:

$$F_2 = \frac{x_R^{(1)} - x_L^{(1)}}{\min\left[(x_R^{(1)} - x_L^{(2)}), (x_R^{(2)}, x_L^{(2)})\right]} \tag{6}$$

  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:

$$F_3 = \begin{cases} (y_L^{(1)} - y_H^{(1)})/(y_L^{(2)} - y_H^{(2)}) & : \quad \text{if} \quad (y_L^{(1)} - y_H^{(1)}) \leq (y_L^{(2)} - y_H^{(2)}) \\ (y_L^{(2)} - y_H^{(2)})/(y_L^{(1)} - y_H^{(1)}) & : \quad \text{otherwise.} \end{cases} \tag{7}$$

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

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

$$F_4 = \begin{cases} (x_R^{(1)} - x_L^{(1)})/(x_R^{(2)} - x_L^{(2)}) & : \text{ if } \quad (x_R^{(1)} - x_L^{(1)}) \leq (x_R^{(2)} - x_L^{(2)}) \\ (x_R^{(2)} - x_L^{(2)})/(x_R^{(1)} - x_L^{(1)}) & : \text{ otherwise.} \end{cases} \tag{8}$$

  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):

$$F_6 = \frac{2|x_{lc} - x_C|}{x_R^{(2)} - x_L^{(1)}} \tag{9}$$

  where x_{lc} is obtained by

$$x_{lc} = \frac{\sum_{i=0}^{L} x_i^s}{L} \tag{10}$$

  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.

5.2 Fuzzified Decision Rules

Interactive Dichotomizing decision trees and rules were first present by Quinlan [11]. A rule extracted from a Quinlan's decision tree has the following form:

$$F_1(f_{1j}) \wedge F_2(f_{2j}) \wedge ...F_n(f_{nj}) \Longrightarrow C(L_j) \tag{11}$$

$F_i(f_{ij})$ 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 [12].

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

$$m(F_i) = \begin{cases} 1.0 & : \text{ if } \quad F_i > f_{ij} \\ 0.0 & : \text{ if } \quad F_i \leq (1-\alpha) f_{ij} \\ \frac{F_i - (1-\alpha) f_{ij}}{\alpha f_{ij}} & : \text{ if } \quad (1-\alpha) f_{ij} < F_i \leq f_{ij} \end{cases} \tag{12}$$

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

$$m(F_i) = \begin{cases} 1.0 & : \text{ if } \quad F_i \leq f_{ij} \\ 0.0 & : \text{ if } \quad F_i > (1+\alpha) f_{ij} \\ \frac{(1+\alpha) f_{ij} - F_i}{\alpha f_{ij}} & : \text{ if } \quad f_{ij} < F_i \leq (1+\alpha) f_{ij} \end{cases} \tag{13}$$

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, $F_0, F_1, ..., F_8$, discussed in Section 5.1 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.

5.3 Defuzzification

Centroid defuzzification technique is adapted:

$$O_p = \frac{\sum_{i=1}^{N} w_i D_p^i O^i}{\sum_{i=1}^{N} w_i D_p^i} \tag{14}$$

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 the rule i is right. and O_p represents the output of pattern p. In this application, we have two classes, good or bad segmentation path. 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 which presented by pattern p is regarded as a "good segmentation", otherwise, it is 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,

$$D_p^i = \prod_{k=1}^{M_i} m_{ki} \tag{15}$$

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 P_0, which reflects the degree to which a candidate is a good segmentation path, to rank all possible segmentation paths.

5.4 Removing Redundant Segmentation Paths

Because all possible segmentation paths are extracted, some paths may be very 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 5.1) of the ith path and L_f^j is the L_f of the jth path.

$$C_{ij} = \frac{N_{ij}}{L_f^i + L_f^j} \tag{16}$$

Two thresholds, $T_{N_{ij}}$ and $T_{C_{ij}}$ are set to determine if 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 P_0 will be removed.

6 Experimental Results and Discussion

All the parameters used in the construction of segmentation paths should 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 two-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.

insert Table 1

A comparison of the correct path classification rates among the fuzzified decision rules, straightforward decision trees (unpruned and pruned) [11], and multi-layer perceptron (MLP) classifier are shown in Table 1. The inputs of these classifiers are the nine properties of a path given in Section 5.1 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 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 in a string based on the digit recognition results in order to determine a good enough

segmentation path. The separated digits were recognized using an optimized nearest-neighbor classifier proposed by Yan [13]. Sixty-four intensities on the 8×8 image, rescaled from an original 160×160 normalized binary image, were 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 [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 two-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 two-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 two-digit strings, averagely 8.1 and 7.4 segmentation paths were constructed from each digit string before and after removing redundancy.

insert Table 2

Suppose that the tolerant radius is a. If the recognition measure is smaller than aM_i, then the classification is accepted. Table 2 shows the experimental results with different recognition measure factors. Greater the rejection measure factor, the lower the rejection rate and correct classification.

For a comparison, we also applied a few other algorithms to separate the same two-digit strings used in our experiments. These algorithms include the MCM (Modified Curvature Method) of Chi *et al* [5], and those of Fenrich [14], Fujisawa *et al* [15] and Zhao *et al* [16]. Table 2 summarizes the experimental results of these algorithms together with ours.

insert Table 3

We can see from Table 3 that our background-thinning-based approach for segmenting

and recognizing connected 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 92.4% with only 4.7% of digit strings rejected.

insert Figure 9

7 Conclusion

In this paper, a background-thinning-based approach is presented to separate and recognize connected handwritten two-digit strings. In this approach, a set of feature points on the background skeleton of an digit string image are used to trace all possible segmentation paths based on a three-step searching process. Fuzzified decision rules, which are generated from training samples, are used to rank segmentation paths. The top ranked paths are then tested by an optimized nearest neighbor classifier and a good enough segmentation path is accepted based on pre-set acceptance criteria. 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 which is compared favorably with those of other techniques tested.

Acknowledgment

The authors would like to thank the financial support from The Hong Kong Polytechnic University.

References

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

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

[3] 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.

[4] 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.

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

[6] 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.

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

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

[9] L. A. Zadeh. Fuzzy sets. *Information and Controls*, 8:338 – 353, 1965.

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

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

[12] 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.

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

[14] 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.

[15] H. Fujisawa, N. Yasuaki, and K. Kurino. Segmentation methods for character recognition: from segmentation to document structure analysis. *Proc. IEEE*, 80(7):1079 – 1092, 19922.

[16] 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.

Table Captions:

Table 1: A comparison of the correct path classification rates among fuzzified decision rules, straightforward decision trees (unpruned and pruned) and multi-layer perceptron classifier.

Table 2: Experimental results of test samples with different recognition measure radius ($M_i, i = 0, 1, ..., 9$, are the mean values of recognition measures for digits 0, 1, ..., 9).

Table 3: A performance comparison of our approach with other digit separation algorithms.

Figure Captions:

Figure 1: Flowchart of our background-thinning-based approach for segmenting handwritten digit strings.

Figure 2: Connection type 3: two strokes touch end to end.

Figure 3: Examples of background skeletons with think lines indicating (a) base segments; (b) branch segments; and (c) hole segments.

Figure 4: Examples of feature points on background skeletons.

Figure 5: Flowchart of the construction of segmentation paths by a three-step searching process.

Figure 6: Flowchart of the top-down searching of segmentation paths.

Figure 7: 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).

Figure 8: Flowchart for the searching beginning at a feature point on a hole segment.

Figure 9: Examples of connected digit strings with (a) all possible segmentation paths; (b) segmentation paths after redundancy removing, and (c) winning segmentation paths (dark lines).

Table 1:

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

Table 2:

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%)

Table 3:

Techniques	High rejection rate		Low rejection rate	
	Rejection (%)	Wrong (%)	Rejection (%)	Wrong (%)
Our algorithm	28.6	3.0	4.7	7.5
MCM of Chi [5]	32.7	4.9	3.7	10.8
Fenrich algorithm [14]	53.3	5.8	5.3	27.1
Fujisawa algorithm [15]	33.5	7.7	0.7	21.9
Zhao *et al* algorithm [16]	38.3	4.4	3.4	13.3

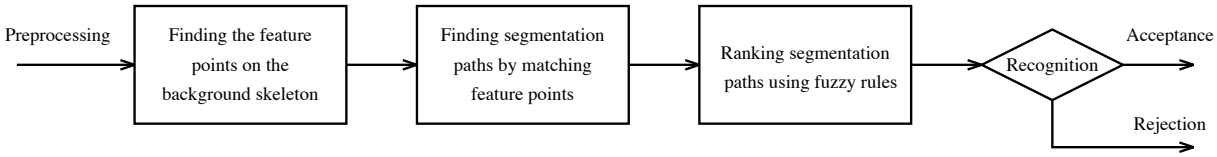


Figure 1:

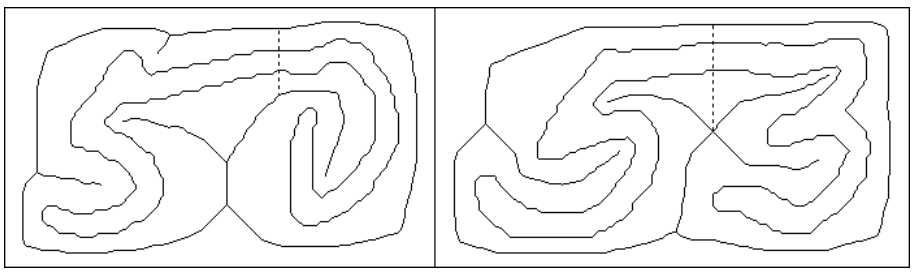
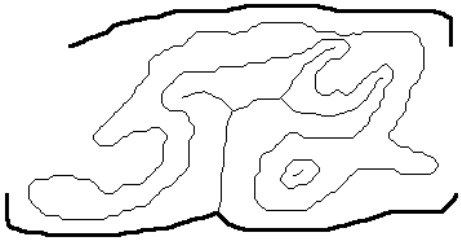
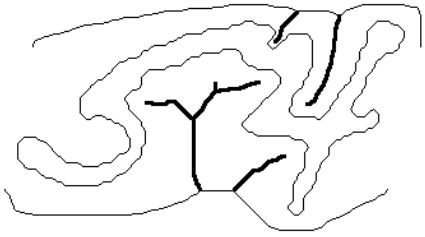


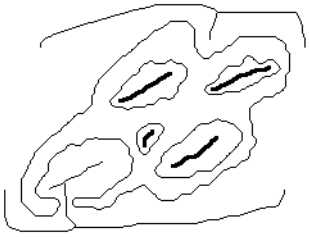
Figure 2:



(a)



(b)



(c)

Figure 3:

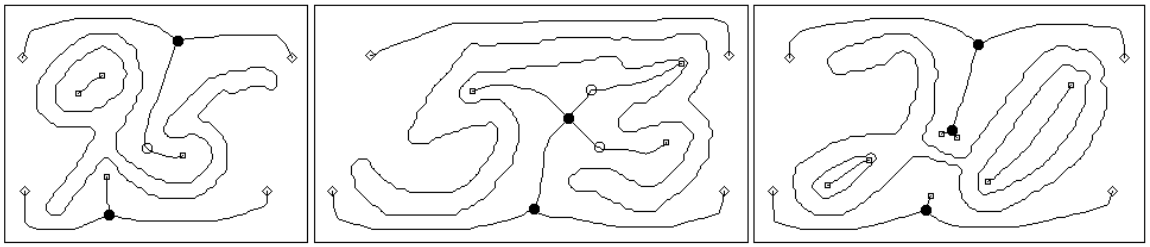


Figure 4:

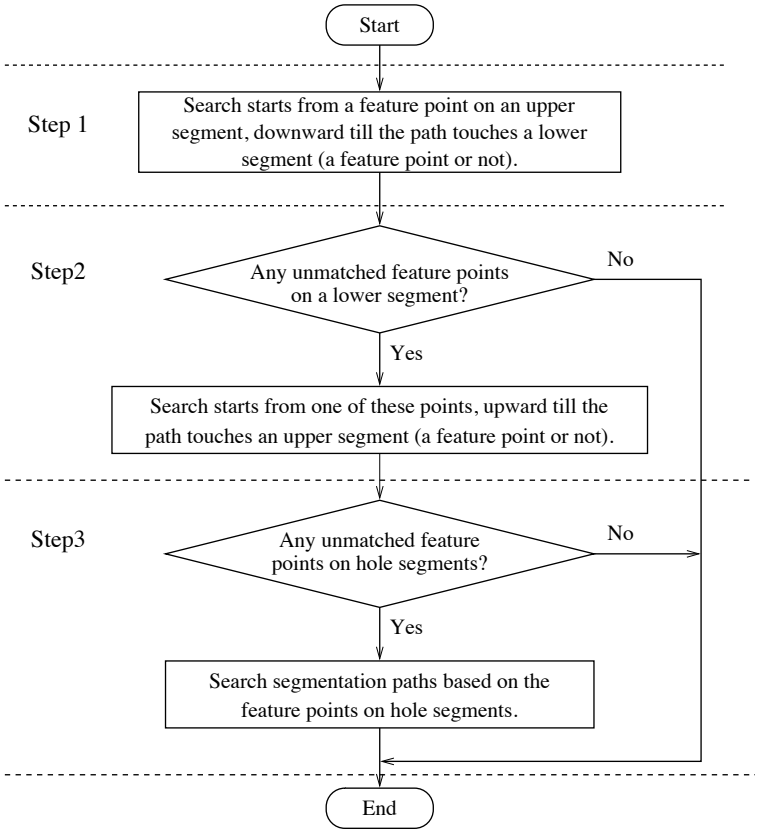

Start
Step 1
Search starts from a feature point on an upper segment, downward till the path touches a lower segment (a feature point or not).
Step2
Any unmatched feature points on a lower segment?
No
Yes
Search starts from one of these points, upward till the path touches an upper segment (a feature point or not).
Step3
Any unmatched feature points on hole segments?
No
Yes
Search segmentation paths based on the feature points on hole segments.
End

Figure 5:

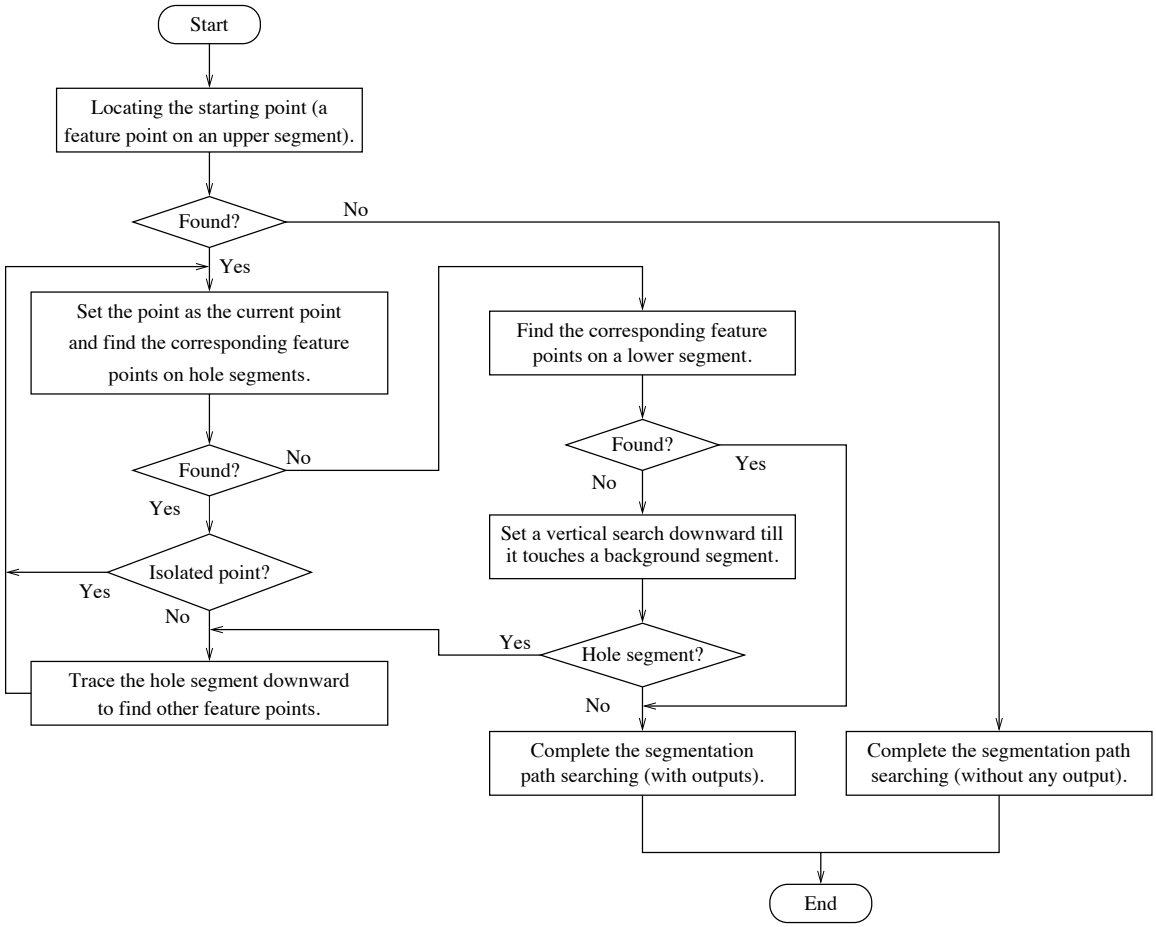
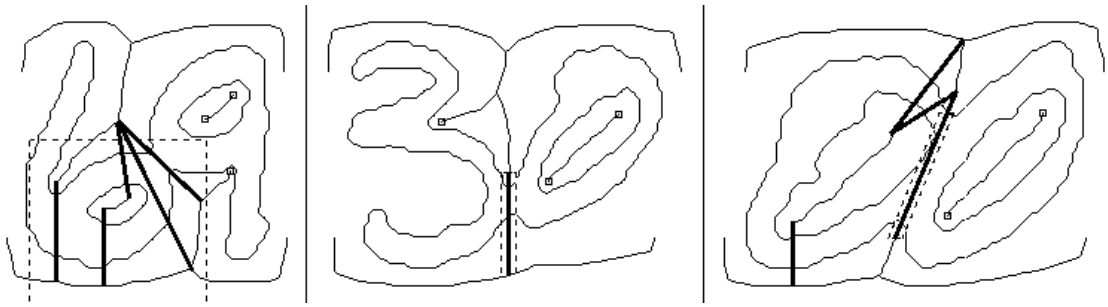
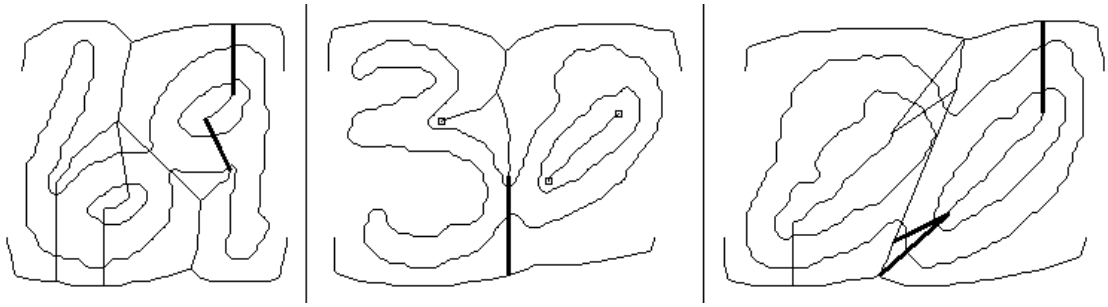

Start
Locating the starting point (a feature point on an upper segment).
Found?
No
Yes
Set the point as the current point and find the corresponding feature points on hole segments.
Found?
No
Yes
Isolated point?
Yes
No
Trace the hole segment downward to find other feature points.
Find the corresponding feature points on a lower segment.
Found?
Yes
No
Set a vertical search downward till it touches a background segment.
Yes
Hole segment?
No
Complete the segmentation path searching (with outputs).
Complete the segmentation path searching (without any output).
End

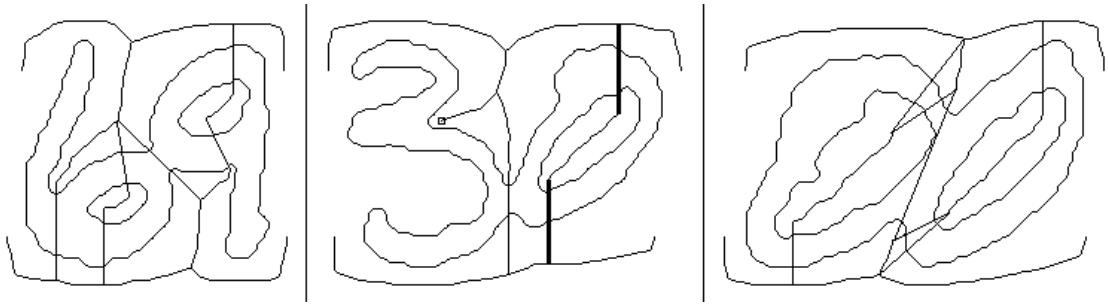
Figure 6:



(a)



(b)



(c)

Figure 7:

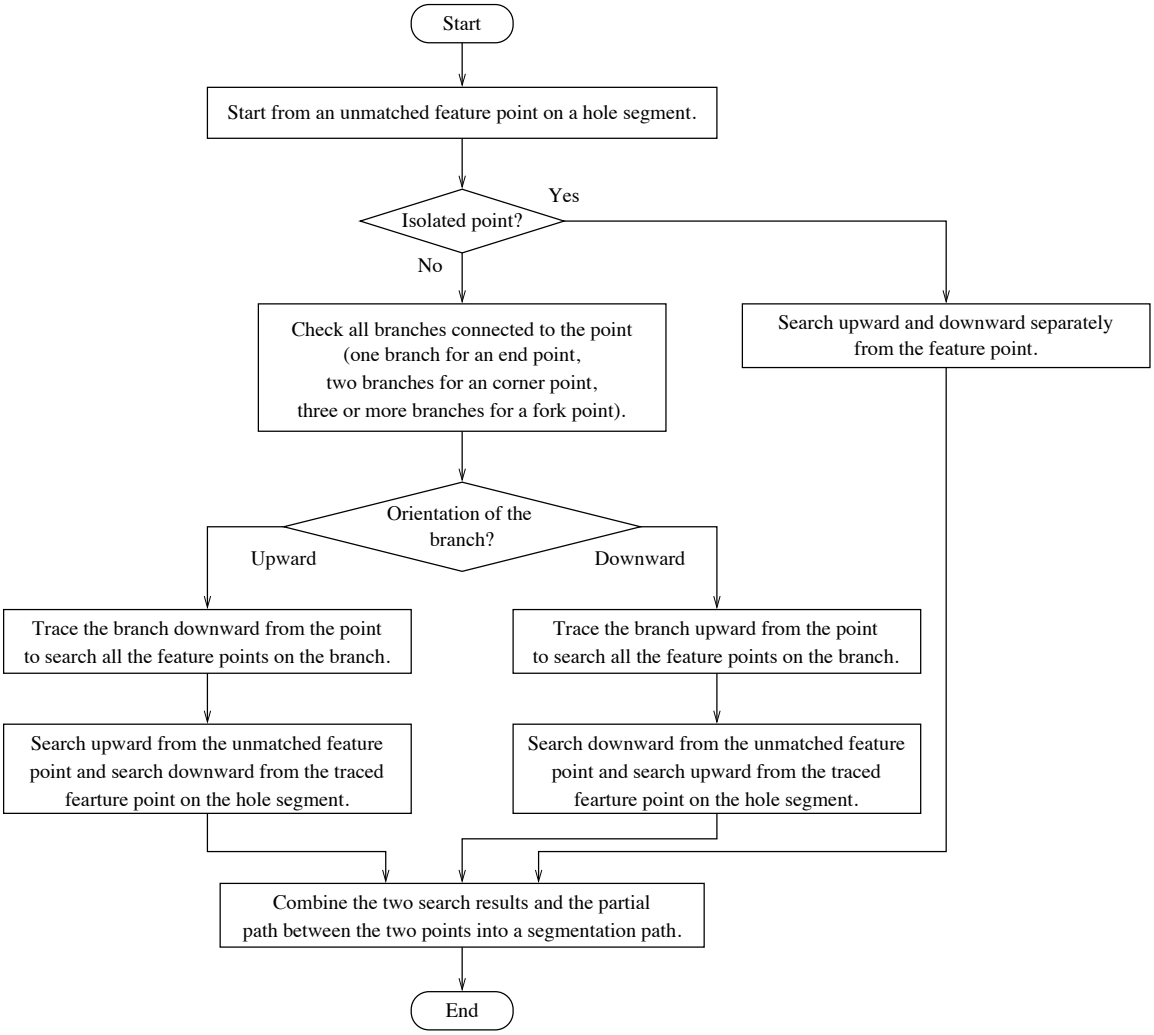

Start
Start from an unmatched feature point on a hole segment.
Isolated point?
Yes
No
Check all branches connected to the point
(one branch for an end point,
two branches for an corner point,
three or more branches for a fork point).
Search upward and downward separately
from the feature point.
Orientation of the
branch?
Upward
Downward
Trace the branch downward from the point
to search all the feature points on the branch.
Trace the branch upward from the point
to search all the feature points on the branch.
Search upward from the unmatched feature
point and search downward from the traced
fearture point on the hole segment.
Search downward from the unmatched feature
point and search upward from the traced
fearture point on the hole segment.
Combine the two search results and the partial
path between the two points into a segmentation path.
End

Figure 8:

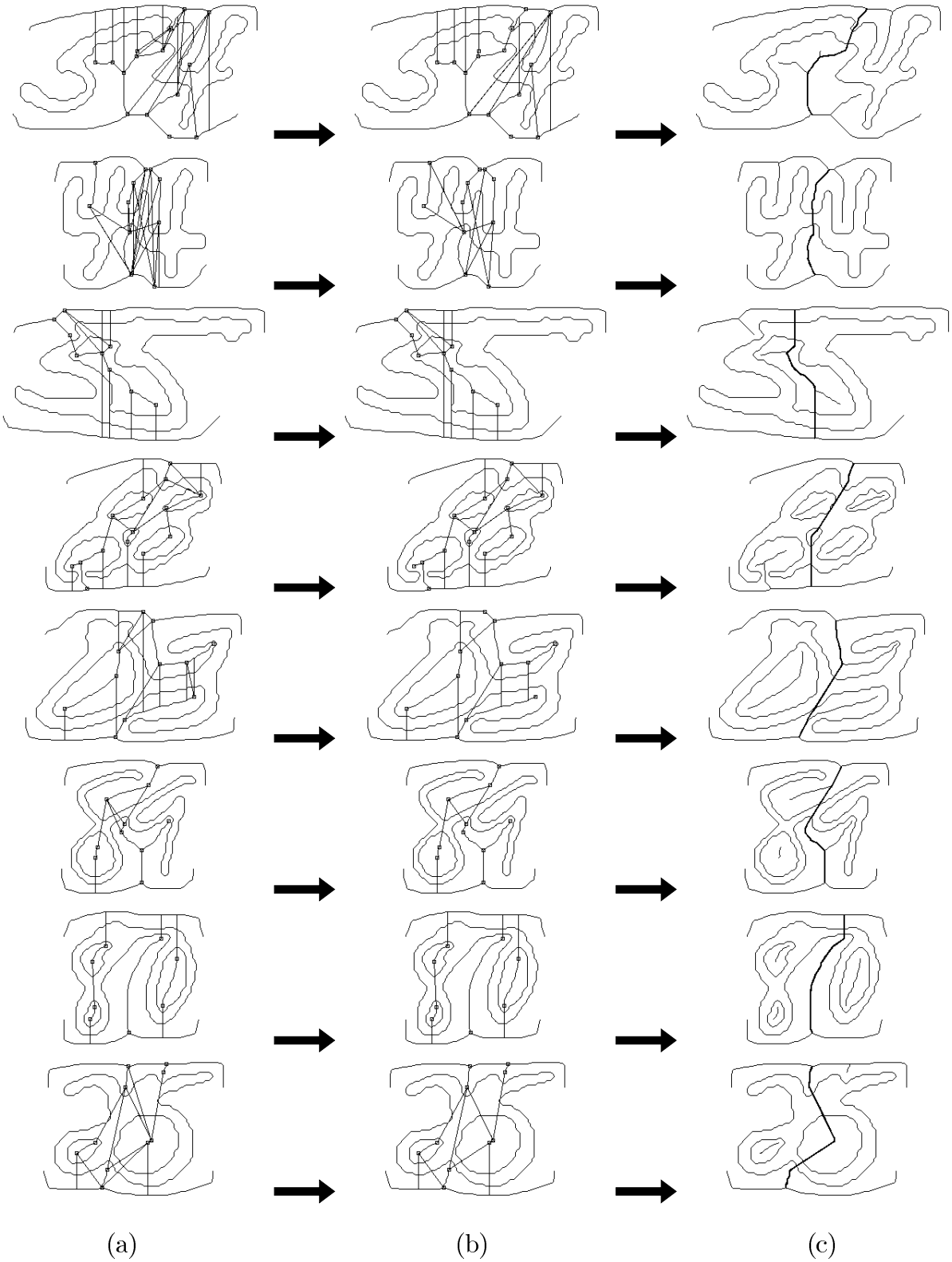


Figure 9: