(1) **Name of the workshop:** Artificial Intelligence

(2) **Title:** Handwritten Numeral Recognition Based on Reduced Features Extraction and Fuzzy Membership Functions

(3) **Abstract:** We design and develop a handwritten numeral recognition system based on reduced features extraction and fuzzy membership functions with the intention to find a minimal set of feature primitives without sacrificing the recognition rate. We first perform preprocessing of smoothing and thinning to obtain a skeleton for each image. We then extract the following feature primitives for each skeleton: loop, horizontal, vertical, C-like curve, and D-like curve. Two fuzzy S-functions are used as membership functions to estimate their likelihood of being close to the top and to the bottom. A tree-like classifier based on the reduced feature primitives and fuzzy memberships is then applied to recognize the numerals.

(4)

Authors: Chichang Jou, Tai-Yuan Hsiao, Hung-Chang Lee

Affiliations: Department of Information Management, Tamkang University

Department of Information Management
Tamkang University
151 Ying-chuan Road
Correspondences: Tamsui, Taipei County
Taiwan 251, Republic of China
Tel:+886 (02) 26215656 x 2846
Fax: +886 (02) 26209737
cjou@mail.im.tku.edu.tw, tai@mail.im.tku.edu.tw, hclee@mail.im.tku.edu.tw

(5) **Contact author:** Chichang Jou

(6) **Keywords:** Handwritten Numeral Recognition, Pattern Recognition, Features Extraction, Fuzzy Membership Functions

Handwritten Numeral Recognition Based on Reduced Features Extraction and Fuzzy Membership Functions

Chichang Jou, Tai-Yuan Hsiao, Hung-Chang Lee

Department of Information Management, Tamkang University

Abstract

Structural classification has been adopted to recognize handwritten numerals by extracting feature primitives that characterize each image. We propose a handwritten numeral recognition system based on reduced features extraction and fuzzy membership functions, with the intention to find a minimal set of feature primitives without sacrificing the recognition rate. We first perform preprocessing of smoothing and thinning to obtain a skeleton for each image. For each skeleton, the following feature points are detected: terminal, intersection, and directional. We then extract the following five feature primitives for each skeleton: loop, horizontal, vertical, C-like curve, and D-like curve. Two fuzzy S-functions are used as membership functions to estimate the likelihood of these primitives being close to the top and to the bottom of the image. A tree-like classifier based on the feature primitives and fuzzy memberships is then applied to recognize the numerals. Handwritten numerals in NIST Special Database 19 are recognized with 88.72% correct rate.

Keyword: Handwritten Numeral Recognition, Pattern Recognition, Features Extraction, Fuzzy Membership Function

1. Introduction

With numerous potential commercial applications, handwritten character recognition has been an active research field. With miscellaneous cultural backgrounds and extensive varieties of individual writing styles, the same character could be written in many forms and outlines. Suen et. al. [18] observed that when ordinary people read an incomplete handwritten article, even after training, their error rate of recognition with regard to article contents is still about 4%. It is rather difficult for anyone to write the same character several times without any change. These all demonstrate the difficulty in handwritten character recognition. In this paper, we narrow down the problem to handwritten numeral recognition.

The handwritten numeral recognition problem has been studied from syntactic recognition [1], neural networks [2,10], structural classification [2,5,6,8,12], coding [5,17], fuzzy logic [11,17], etc. Recently, there is a trend of combining two recognition methods to obtain better recognition rate [19,22]. Among the above methodologies, structural classification is the most usually used. It normally obtains the skeleton of each numeral first. Through detection of feature points, like terminal and turning points, it then decomposes each skeleton into several feature primitives. By the characteristics of these primitives and their relative positions, it finally classifies the numerals. In this methodology, the handwritten numeral recognition problem is reduced to: what are the effective feature primitives and how to utilize their characteristics and relative positions to recognize them.

Siy and Chen [17] decomposed images of handwritten numerals into a set of fifteen feature primitives, which are of the following categories: (1) straight lines, including horizontal, vertical, positive slope, and negative slope, (2) circles, including plain circle and circle on the left, on the right, above and below, and (3) open arcs, including C-like, D-like, A-like, V-like, S-like, and Z-like. This decomposition was based on the detection of the following feature points: tips, corners, and junctions. To increase correct recognition rate, Malaviya and Peters [11] increased the number of feature primitives, and used multi-stage feature aggregation to describe each image using fuzzy rules. However, the more the number of feature primitives, the more the computational burden.

We propose a feature extraction by a set of five feature primitives to classify handwritten numerals correctly without sacrificing the recognition rate. As far as we know, this is the minimal set of feature primitives in handwritten numeral recognition. We first integrate the smoothing algorithm of Hu et. al. [7] and the thinning algorithm of Datta and Parul [3] to obtain a skeleton for the bitmap image of each numeral. Feature points and paths proposed by Hu and Yan [6] are then detected for each skeleton. Following Nishiba and Mori [14], for each skeleton, we check continuously whether the current path and its concatenated path could be merged into one. After the above operations, we obtain several paths for each handwritten numeral. Each path is then classified into one of the following five feature primitives: loop, horizontal, vertical, C-like curve, and D-like curve. Two fuzzy S-functions are used as membership functions to estimate their likelihood of being close to the top and to the bottom of the image. Previous experiences in OCR [13] suggested that the recognition rate of characters would be low if only one of feature extraction and classification algorithm is used. We thus propose a tree-like classification based on characteristics of these feature points, primitives and membership functions.

In Section 2, we will introduce the smoothing and thinning algorithms in the preprocessing. We then discuss skeleton decomposition in Section 3 with the illustration of feature points detection, path division, and feature primitive classification. The two fuzzy membership functions and the tree-like classifier are demonstrated in Section 4. In Section 5, the test database NIST Special Database 19 is introduced, and test result is compared with that of other research. Section 6 concludes and points out future research directions.

2. Smoothing and Thinning

To overcome unavoidable noise in digitized images, we adopt Hu et. al.'s algorithm [7] to smooth image boundaries and to compensate their stroke width. It first eliminate the short and extra spurs and fill in small holes. A stroke width compensation algorithm is then applied to ensure that the width of each horizontal and vertical stroke is larger than three pixels by filling pixels with large influence value.

Image thinning should preserve connectivity and should not shorten skeletal legs. Thinning will save the amount of memory space in storing the image, and also simplify later processing. However, thinning itself is easy to distort the images, and has a direct impact on the recognition rate. Depending on whether pixels could be processed simultaneously, thinning algorithms are classified as: sequential or parallel [9]. Recent researches in this field focus on the parallel thinning algorithms. We choose Datta and Parul's thinning algorithm [3] to transform the smoothed bitmap image of handwritten numerals into a skeleton. This algorithm is fast and stable, ensures one-pixel wide result, and preserves the connectivity. Figure 1 displays the five templates for determining whether one point could be eliminated.

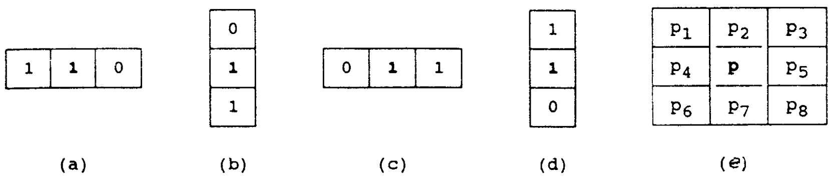


Figure 1: Templates for Thinning Algorithm

By checking neighboring pixels, when a point P satisfies any of the following conditions, P's elimination will destroy the original connectivity of the image:

(1) P satisfies template (a) and ((p2 =0 and p3 =1) or (p7=0 and p8=1))

(2) P satisfies template (b) and ((p4 =0 and p1 =1) or (p5=0 and p3=1))

(3) P satisfies template (c) and ((p2 =0 and p6 =1) or (p7=0 and p6=1))

(4) P satisfies template (d) and ((p4 =0 and p6 =1) or (p5=0 and p8=1))

(5) P satisfies template (e) and p1+p2+p3+p4+p5+p6+p7+p8 <= 1

The thinning algorithm iteratively eliminates points satisfying conditions (1) to (4), until no more points could be eliminated. After smoothing and thinning, we obtain a one-pixel wide skeleton for each image.

3. Skeleton Decomposition

We adopt the structural approach [4] to decompose images into several simple feature primitives and then reconstruct these primitives for later processing. We first detect the skeleton's feature points that reflect a change of characteristics, like change of directions or termination of a stroke. From these feature points, we decompose the skeleton into a set of feature primitives. Siy and Chen [17] proposed the set of fifteen sufficient primitives in Figure 2 for handwritten numeral recognitions. The number of their primitives is large, which increases the computational burden. We instead propose in Section 3.3 a set of five feature primitives for the classification.

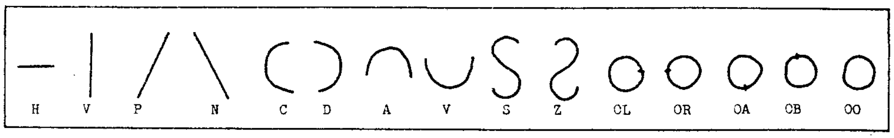


Figure 2: Sufficient Feature Primitives for Handwritten Numeral Recognition

3.1. Detection of Feature Points

We follow Hu and Yan's algorithm [6] to find a set of feature points, which will be used for skeleton decomposition in the next subsection. The definitions of these feature points are as follows:

Definition: *T* (terminal) points are the points with exactly one black neighbor among their 8 neighbors. *I* (intersection) points are those points with more than or equal to two black points among their 8 neighbors. A *path* is a sequence of continuous black points, where both its starting point and ending point being a T or I point. *D* (directional) points are those points that change directions in either x-axis or y-axis in a path.

Note that we no longer detect the bending points in Hu and Yan's algorithm to speed up the algorithm. Figure 3 shows the modified steps in the detection of these feature points: Starting from the left upper corner of the skeleton, we calculate for all the black points, from left to right, and then from top to bottom, the number of black points among their eight neighbors to determine the set of all T and I points. Then for each T point, we find all the D points in its path until another T or I point is met. Similarly, for each I point, we find all the D points in its path until another T or I point is met. Figure 4 demonstrates the resulting paths for a skeleton of an image of '2'.

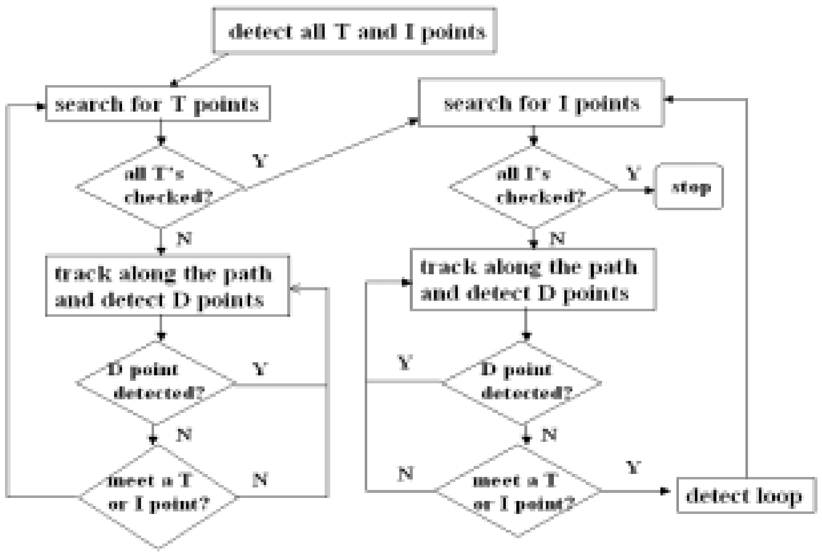


Figure 3: Detection of Feature Points

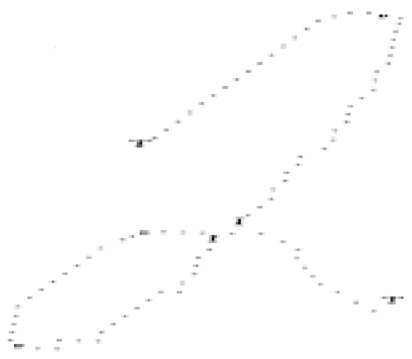


Figure 4: Feature Points of an “2”

3.2. Merging of the Paths

We follow the right-hand rule proposed by Nishida and Mori [14] to decide whether two connecting paths could be merged. Figure 5 demonstrates how to use the rule as follows:

$$\Delta(a,b) = sign\begin{vmatrix} x_a - x_p & x_b - x_p \\ y_a - y_p & y_b - y_p \end{vmatrix}$$

$$\text{sign}x = \begin{cases} -1 & \text{if } x < 0 \\ 0 & \text{if } x = 0 \\ 1 & \text{if } x > 0 \end{cases}$$

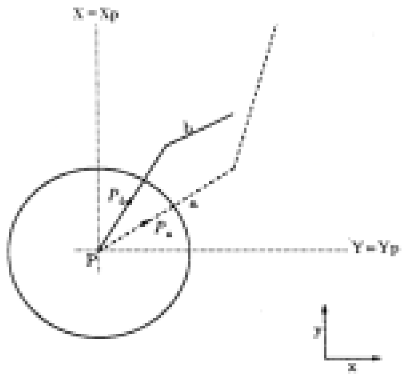


Figure 5: Right-hand Rule of Path Merging

Suppose *a* and *b* are two connecting paths, and *P* is the intersection point of *a* and *b*. Let P_a and P_b be points sufficiently close to *P* such that P_a and P_b are contained in only *a* and only *b*, respectively. Let (x_p, y_p), (x_a, y_a), and (x_b, y_b) be the coordinates of *P*, P_a, and P_b, respectively. If $\Delta(a, b) = 1$, then these two paths could be merged as one, using *a*'s starting point as its starting point, and *b*'s ending point as its ending point.

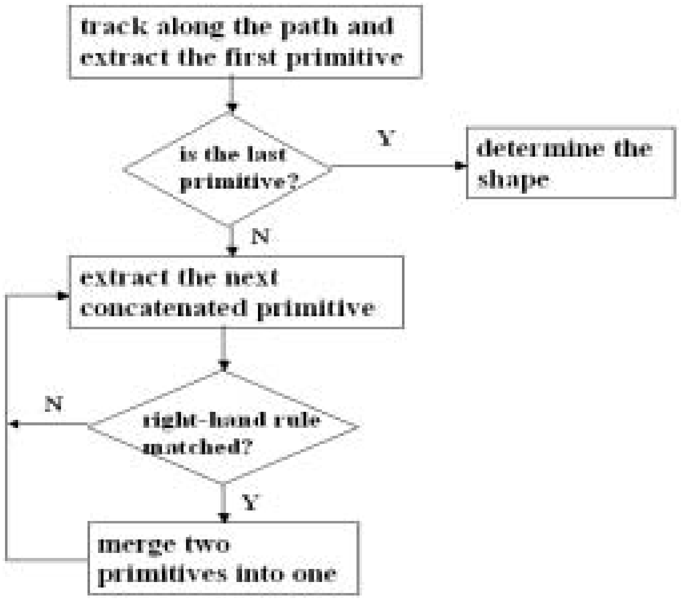

track along the path and extract the first primitive
is the last primitive?
Y
determine the shape
N
extract the next concatenated primitive
right-hand rule matched?
N
Y
merge two primitives into one

Figure 6: The Path-Merging Algorithm

The merging algorithm is shown in Figure 6. For each skeleton, we number all its paths from left to right and from top to bottom. We test from its first path continuously whether the current path and its concatenated path could be merged according to right-hand rule. Those checked paths are marked during this process. Then we continue from the ending point of all paths of the skeleton to test those unmarked paths. According to this merging algorithm, the skeleton in Figure 5 could be decomposed into primitives similar to D, OL, and H in Figure 2.

3.3. Feature Primitive Classification

After the detection of feature points and merging of paths, each skeleton is decomposed into several paths. Let the x-coordinates and y-coordinates of all points in a skeleton be bounded by X_{min}, X_{max}, and Y_{min}, Y_{max}, respectively. We eliminate those unhelpful paths, like those short paths with length less than $0.3*(Y_{max} - Y_{min})$.

We then classify these paths into five primitives by investigating their characteristics. If the starting point and ending point of a path are the same, then this path is classified a 'loop'. For the rest non-loop paths, we first classify them into lines or curves. The distance $d(P,L)$ between a point P and a straight line L is defined as the minimal of the distances $d(P,P')$ between P and all points P' in L. We then define the distance between a path and a straight line L as the maximal of the distances $d(P,L)$ between L and all points P in the path. For each path, we find its closest straight line L by applying the least squares method [16] using the above distance definition. Suppose their distance is ε_1, and the length between their two intersecting points is ε_2. Let R be the ratio of ε_1 and ε_2,, and R_T be a predefined threshold. For each non-loop path, if its R is greater than R_T, meaning the path's vertical variation over horizontal variation is larger than the threshold, then we classify it as a curve. Otherwise, we classify it as a line.

We classify a line into a vertical or a horizontal by checking the angle between the line and the x-axis. If the angle is less than a predefined threshold θ_T, then we classify the line as a horizontal. Otherwise, it is classified as a vertical. For curves, we classify them into C-like or D-like curves by checking the x-coordinates of its starting and ending points. If the starting and ending points are both in the right hand side of the D-point of the curve, then it is classified as a C-like curve. Otherwise, it is classified as a D-like curve.

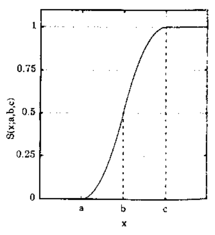
4. Tree-like Classification

Before our investigation into the relative positions of feature primitives, we normalize the image to the same size. As in Section 3.3, let the x-coordinates and y-coordinates of all points in a skeleton be bounded by X_{min}, X_{max}, and Y_{min}, Y_{max}. The height and width of the skeleton are $(Y_{max} - Y_{min})$ and $(X_{max} - X_{min})$, respectively. The coordinates (X, Y) of the starting and ending points of the primitives would then be normalized to be within [0,1] by the following formulae:

$$X_n = (X - X_{min}) / (X_{max} - X_{min}) \quad \text{and} \quad Y_n = (Y - Y_{min}) / (Y_{max} - Y_{min})$$

Using the above normalized coordinates, we then train two S-functions like the one in

Figure 7 as the fuzzy membership functions to determine the likelihood of a point being close to the top and the bottom of the skeleton. By averaging the values of above membership functions for the starting and ending points of a primitive, we estimate how close is the primitive to the top and the bottom of the image.



$$S(x;a,b,c)=\begin{cases}0\\2\left(\frac{x-a}{c-a}\right)^2\\1-2\left(\frac{x-c}{c-a}\right)^2\\1\end{cases}$$

Figure 7: Fuzzy Membership Function

We utilize the tree-like classifier in Figure 8 to classify the images, where ‘*o*’ represents the loop, ‘-‘ the horizontal, ‘｜’ the vertical, ‘（‘ the C-like curve, ‘）’ the D-like curve, and ‘ϕ’ no primitives found.

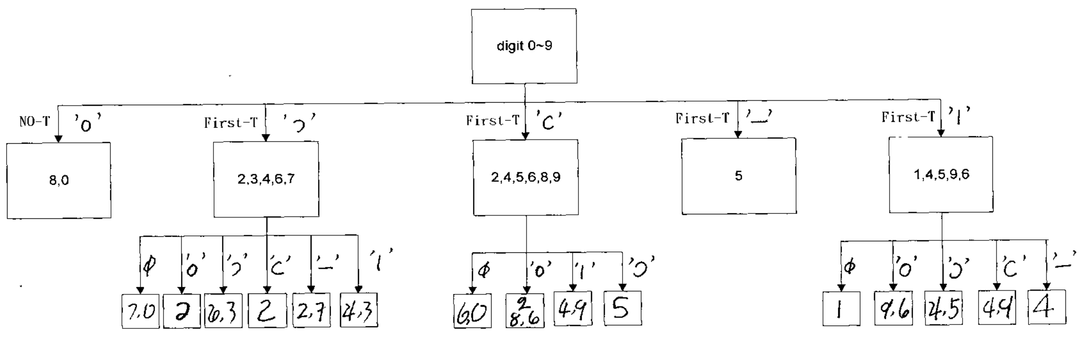


Figure 8: The Tree-like Classifier

The mechanism works as follows: We scan the skeleton from left to right, and then from top to bottom, to find the first T-point. If a T-point is found, then its associated primitive must belong to one of the following four: ‘（‘, ‘）’, ‘｜’, and ‘-‘.

Otherwise, we assign loop ‘*o*’ as its associated primitive. According to the above primitive, we classify the image into the first layer nodes. For first layer nodes with more than three digits, we classify them according to the next connecting primitive to obtain the second layer nodes. For end nodes with one digit, the classification finishes. For end nodes with two or three digits, the notation *node*(d_1, d_2) denotes the node with digits d_1 and d_2, and *node*(d_1,d_2,d_3) the node with digits d_1, d_2, and d_3. We use the following method to further classify them:

node(8,0): If we could find more than one ‘*o*’ primitive, the image is classified as ‘8’. Otherwise, it is classified as ‘0’.

node(7,0): If ‘）’ is close to the bottom, then it is classified as ‘7’. Otherwise, it is classified as ‘0’.

node(6,3): If the second ‘)’ is close to the bottom, then it is classified as ‘6’. Otherwise, it is classified as ‘3’.

node(2,7): If ‘-’ is close to the bottom, then it is classified as ‘7’. Otherwise, it is classified as ‘2’.

node(4,3): If ‘)’ is close to the top, then it is classified as ‘3’. Otherwise, it is classified as ‘4’.

node(6,0): If ‘(’ is close to the top, then it is classified as ‘0’. Otherwise, it is classified as ‘6’.

node(2,8,6): If the middle point of primitive ‘*o*’ is close to the top, then it is classified as ‘2’. Otherwise, if the I-point is in the central location of the image, then it is classified as ‘8’. Otherwise, it is classified as ‘6’.

node(4,9): If ‘（’ is close to the bottom, then it is classified as ‘4’. Otherwise, it is classified as ‘9’.

node(9,6): If the middle point of ‘*o*’ is close to the top, then it is classified as ‘9’. Otherwise, it is classified as ‘6’.

node(4,5): If ‘）’ is close to the top, then it is classified as ‘4’. Otherwise, it is classified as ‘5’.

5. Experimental Results

The handwritten numerals we use to test our system are from NIST (National Institute of Standards and Technology) Special Database 19 [15]. It collected 810,000 handwritten characters enclosed in forms of 3,600 people. Each character was scanned in 300 dpi resolutions to obtain a 128*128 bitmap image. The number of handwritten numerals in it is 60,089. We use 5,000 skeletons as our training set to obtain the values for the thresholds R_T and θ_T , and the parameters a, b, c of the two S-functions with best classification result.

We compute the following performance parameters: (1) Correct rate: the ratio of the number of correctly classified numerals and the number of tested numerals. (2) Error rate: the ratio of the number of wrongly classified numerals and the number of tested numerals. (3) Rejected rate: the ratio of the number of unclassifiable numerals and the number of tested numerals. (4) Reliability: the ratio of correct rate and the sum of correct rate and error rate. Table 1 demonstrates our result for each digit, compared with that of Hu and Yan [6]. Note that Hu and Yan used a different database, NIST Special Database 3, with only 20,852 images of handwritten numerals.

Table 1: Our Recognition Result Compared with That of Hu and Yan

Numeral	Correct rate(%)	Error rate(%)	Rejected rate(%)	Reliability(%)
0	91.62	6.76	1.41	93.13
1	92.56	7.27	0.17	92.72
2	87.78	11.26	0.96	88.63
3	82.33	13.46	2.56	84.49
4	85.16	7.48	1.38	85.48
5	91.54	7.77	0.98	92.45
6	90.47	5.97	1.76	92.09
7	90.67	9.71	3.36	93.82
8	88.25	8.78	2.04	90.09
9	86.78	9.36	4.44	90.81
Total (our)	88.72	9.36	1.91	90.37
Total (Hu and Yan)	88.79	0.25	10.96	99.72

Even though the number of primitives is reduced from Siy and Chen's fifteen to five, our correct rate is only a little lower than that of Hu and Yan. The fuzzy closeness of primitives to the top or the bottom from our membership functions helps in the final classification for the choice among two or three digits. Because the number of final primitives is few, our rejected rate is low. That caused our error rate higher than that of Hu and Yan. On the other hand, with a higher rejected rate, Hu and Yan's error rate is lower.

6. Conclusions

We designed and implemented a tree-like classifier for handwritten numeral recognition based on reduced feature primitives and fuzzy memberships. We integrated the smoothing algorithm of Hu et. al. and the thinning algorithm of Datta and Parul to obtain a skeleton for each image. Hu and Yan's algorithm was then applied to catch feature points of each skeleton and to decompose it into several paths. Nishida and Mori's right-hand rule is then checked to merge these paths. We classified the resulting paths into a set of five feature primitives. As far as we know, this number of feature primitives is minimal. Two S-functions were then applied to these primitives to estimate their likelihood of being close to the top and bottom of the image. Finally, a tree-like classifier is utilized to classify these numerals based on the feature points, feature primitives, and membership functions. Handwritten numerals in NIST Special Database 19 are recognized by our system with 88.72% correct rate.

The following are some future research ideas: Our training to obtain the parameters in the two S-functions and the thresholds R_T and θ_T takes time and needs to be enhanced. The studies of recognizing overlapping characters and the impact of form lines on handwritten character recognition [20] would be very useful in commercial applications. Another interesting topic is to study the influence of cultural background on the features of handwritten characters.

References

[1] Ali, F. and T. Pavlidis, "Syntactic recognition of handwritten numerals," IEEE Trans. On System, Man, and Cybernetics, Vol. 7, pp. 537-541(1977).

[2] Cao, Jun, and M. Ahmadi and M. Shridhar, "Recognition of handwritten numerals with multiple feature and multistage classifier," Pattern Recognition, Vol. 28, No. 2, pp. 153-160 (1995).

[3] Datta, A. and S.K. Parul, "A robust parallel thinning algorithm for binary images", Pattern Recognition, Vol. 27, No. 9, pp. 1181-1192 (1994).

[4] Fu, K. S., "Syntactic Pattern Recognition and Application," Englewood Cliffs, NJ, Prentice-Hall, 1982.

[5] Hu, J. and H. Yan, "Structural primitive extraction and coding for handwritten numeral recognition," Pattern Recognition, Vol. 31, No. 5, pp. 493-509(1998).

[6] Hu, J. and H.Yan, "Structural decomposition and description of printed and handwritten characters," Proc. 13th Int. Conf. Pattern Recognition, Vienna, Austria, Vol.Ⅲ, Track C,

pp. 230-234(1996).

[7] Hu, J., D. Yu and H. Yan, “Algorithm for stroke width compensation of handwritten characters,” Electronics Letters, Vol.32, No.24, pp.2221-2222 (1996).

[8] Lam, L., and C. Y. Suen, “Structural classification and relaxation matching of totally unconstrained of handwritten ZIP-code numbers,” Pattern Recognition, Vol. 21, pp. 19-31 (1988).

[9] Lam, Louisa, Seong-Whan Lee and Ching Y. Suen, “Thinning methodologies－a comprehensive survey,” IEEE Trans. on PAMI, Vol. 14, No. 9, pp. 869-885(1992).

[10] Lim, Kil-Taek, S. I. Chien and S. J. Kang, “Multiple novelty input neural networks for unconstrained handwritten numeral recognition,” Electronics Letters, Vol. 34, No. 11, pp. 1112-1113 (1998).

[11] Malaviya, Ashutosh and Liliane Peters, “Fuzzy feature description of handwriting Patterns,” Pattern Recognition, Vol. 30, No. 10, pp. 1591-1604 (1997).

[12] Marcelli, Angelo and Natasha Likhareva, Theo Pavlidis, “Structural indexing for character recognition,” Computer Vision and Image Understanding, Vol. 66, No. 3, pp. 330-346(1997).

[13] Mori, S., C. Y. Suen and K. Yamamoto, “Historical review of OCR research and development,” Proc. IEEE 80, pp. 1029-1158(1992).

[14] Nishida, Hirobumi and Shunji Mori, “Algebraic description of curve structure,” IEEE Trans. on PAMI, Vol. 14, No. 5, pp. 516-533(1992).

[15] NIST Special Database 19, www.nist.gov/srd/nistsd19.htm.

[16] Press, W.H., S.A.Teukolosky, W.T. Vetterling, and B.P. Flannery, Numeric Recipes in C: The Art of Scientific Computing, Cambridge, Cambridge University Press (1996).

[17] Siy, Pepe and C.S. Chen, “Fuzzy logic for handwritten numeral character recognition,” IEEE Trans. on Systems, Man and Cybernetics, pp. 520-574 (1974).

[18] Suen, C. Y., R. Shinghal and C. C. Kwan, “Dispersion factor: a quantitative measurement of the quality of handprinted characters,” Proc. Int. Conf. Cybernetics and

Society, pp. 681-685(1977).

[19] Wang, Jianguo and H. Yan, “A hybrid method for unconstrained handwritten numeral recognition by combining structural and neural ‘gas’ classifiers,” Pattern Recognition Letters, Vol. 21, No. 6-7, pp. 625—635 (2000).

[20] Yu, D. and H. Yan, “Separation of single-touching handwritten numeral strings based on structural features,” Pattern Recognition, Vol. 31, No. 12, pp. 1835-1847(1998).

[21] Zadeh, L.A., “Calculus of fuzzy restrictions,” in Fuzzy Sets and Their Applications to Cognitive and Decision Processes, L.A. Zadeh et. al. eds, pp. 1-39, Academic Press, New York (1975).

[22] Zhang, P. and L. Chen, “A novel feature extraction method and hybrid tree classification for handwritten numeral recognition,” Pattern Recognition Letters, Vol. 23, No. 1-3, pp. 45-56 (2002).