A New Method for Chinese Characters Classification Using Pseudo Skeleton Features

Kuo-Chin Fan[1*], Ming-Gang Wen[1,2], Chin-Chuan Han[3]

1. Institute of Computer Science and Information Engineering,
National Central University, Chungli, Taiwan, R.O.C.
2. Department of Management Information System,
National Lien-Ho Institute of Technology, Miaoli, Taiwan, R.O.C.
3. Department of Computer Science and Information Engineering,
Chung-Hua University, Hsinchu, Taiwan, R.O.C.

kcfan@csie.ncu.edu.tw,mgwen@mail.nlhu.edu.tw,cchan@chu.edu.tw

Abstract

In this paper, we present a novel method to classify machine printed Chinese characters by using p-skeleton features. In our approach, the p-skeletons of characters are extracted first instead of those skeletons extracted from the thinning algorithm. Next, the features of p-skeletons are encoded into two code strings. Last, the edit-distance algorithm is employed to compute the similarity between two characters based on their corresponding encoded strings. The main contribution of this paper is to classify multi-fonts Chinese characters using single-font reference database. Experiments were conducted on 5401 daily-used Chinese characters of various fonts and sizes. Experimental results demonstrate the validity and efficiency of our proposed approach.

1.Introduction

The recognition of Chinese characters has been admitted as a difficult problem. The reasons are listed as follows. First, the number of Chinese characters set is very huge. This generates the reference database of huge size and leads to a time-consuming recognition process [1,2,3,4]. Second, the structure of Chinese characters is more complex than that of English alphabets or numbers. It will increase the complexity of feature extraction process. It also increases the database size and the processing time. Third, the font types of Chinese characters are very rich. The recognition performance on the characters of various fonts decreases rapidly by merely using the reference database of a single specified font. To remedy this problem, many OCR systems utilize the pre-filtering techniques to detect the font types of characters. These approaches must build many reference databases for the characters with various font types in advance. The use of simple features to efficiently classify the characters is the main aim of our work.

In many optical character recognition systems, character skeletons play very crucial roles in both the syntactic and statistical approached. Skeleton extracted by using the thinning algorithm is a well-known approach to obtain the features of skeletons in feature extraction process [8]. However, the aforementioned skeletonization process often results in many unwanted effects, such as spurious branches, the splitting of 4-fork point and so on. Moreover, it is a time-consuming process[9,10]. In this paper, we propose an effective approach base on the contours of characters called pseudo skeletons, or p-skeletons. All features of characters can be extracted from the p-skeletons to classify the Chinese characters.

Our proposed approach consists of three modules, including p-skeleton generation, code string generation, and matching modules, as shown in Fig. 1. The rest of this paper is organized as follows. In section 2, we present the extracting technique to extract the p-skeletons from which the code strings are generated. The edit-distance algorithm to calculate the similarity between two code strings is described in section 3. Experimental results are demonstrated in section 4 to verify the validity of the proposed approach. Finally, concluding remarks are given in section 5.

2.Feature extraction

In the feature extraction process, p-skeleton

* To whom all correspondence should be addressed.

generation, projection of p-skeleton, and code string encoding, are executed to obtain the features of character. There are three advantages in the proposed feature extraction process. First, the normalization process on the character images is needless. Second, the p-skeletons of characters will replace the thinning skeletons. Hence, the thinning step of characters is also ignored to save the execution time in the preprocessing. Finally, the 2-D image features are transformed to two 1-D code string features. We can easily measure the similarity between two Chinese characters by these code strings in the classification process.

In the followings contexts the steps to extract the features embedded in the character images will be stated.

2.1. P-skeleton generation

The skeletons of character images represent both the meaning and structure of Chinese characters. The simplest skeleton extraction approach is the thinning-based algorithm [7,8,9]. Traditionally, the thinning of character images is the preprocessing step in many image processing applications [1,4]. However, the algorithm will lead to unwanted effect, such as the 3-fork, end-forks, and so on. The block-matched skeletonization algorithm is an alternative approach to solve these problems [7]. Referring to the recognition model of human being, people can directly identify the character merely based on contours of characters. It is not necessary to extract the skeletons of characters in the recognition process.

Generally, the p-skeletons are defined as the subset of contour pixels, which can be extracted from the character images by using some simple logic operations. The advantage of this approach is that tremendous amount of time to obtain thinned images is saved. Besides, it can be easily implemented by hardware. All features of characters are extracted from the p-skeleton directly. The extraction process of p-skeleton is described as follows.

Consider a character image *A*, the shifted image is obtained by shifting the original image one pixel right and one pixel down as shown in Fig. 2a. Next, the logical operation AND is operated on the shifted and original images (Fig. 2b). The ANDed image is then operated with the original image to obtain the p-skeleton image by performing the Exclusive-OR operation pixel by pixel as shown in Fig. 2c. The pixel $Q_{i,j}$ of p-skeleton image can be generated from the pixels $P_{i,j}$ and $P_{i-1,j-1}$ in the original images by the following logical operation:

$$Q_{i,j} = (P_{i,j} \wedge P_{i-1,j-1}) \oplus P_{i,j} \qquad (1)$$

Here, the subscripts in Eq(1) denote the locations of pixels. After performing the above steps, the contour pixels at the top or left side of character image are remained (Fig. 2d). These pixels, called pseudo-skeleton or p-skeleton, still preserve the information of character strokes. They can easily be identified by human being vision systems. Besides, the extraction process can be implemented by the logical circuits that contain three logic operations including SHIFT, AND, and XOR.

2.2. Projection by p-skeleton

The code strings of characters are extracted from the p-skeletons images by using the projection operations. The horizontal and vertical histograms are generated to obtain two code strings C_h and C_v. To simplify of implementation, the horizontal and vertical histograms are obtained from the horizontal and vertical p-skeletons, respectively. Therefore, the operation in Eq (1) can be decomposed into two sub-operations for the horizontal and vertical p-skeletons.

Consider pixel $P_{i,j}$ in a character image as shown in Fig. 3(a), pixel $Q^v_{i,j}$ of the vertical p-skeleton and the horizontal p-skeletons image $Q^h_{i,j}$ can be generated by the modified operation of Eq(1) as follows:

$$Q^v_{i,j} = (P_{i,j} \wedge P_{i-1,j}) \oplus P_{i,j} \quad \text{for } 0 \le i \le I_w, 0 \le j \le I_h \qquad (2)$$

$$Q^h_{i,j} = (P_{i,j} \wedge P_{i,j-1}) \oplus P_{i,j} \quad \text{for } 0 \le i \le I_w, 0 \le j \le I_h \qquad (3)$$

Where I_w is the image width and I_h is the image height. Fig. 3(b) and Fig. 3(d) illustrate the vertical and horizontal p-skeleton images which are generated from Fig. 3(a), respectively. The projected histograms of Fig. 3(b) and Fig. 3(d) can then be generated from these two p-skeleton images as shown in Fig. 3(c) and Fig. 3(e), respectively.

2.3.Code string Encoding

The last step in feature extraction is to encode the histograms of p-skeleton images H_v and H_h. The encoded string represents the structure features of Chinese characters. Now, let us introduce the converting process of code string as follows. Consider the projection pixel number in the histogram (H) of horizontal or vertical p-skeleton images, the pixel at location θ possesses the maximal value H_θ in histogram H such as $H_\theta = \{\max H_j \quad 0<=j<=|H|\}$, where value |H| is the size of histogram H. A segment G is generated by extending the neighbors from point θ iteratively. Initially, point θ is assigned to segment G of length 1. The left neighbor θ_l is added to segment G if the histogram value at point θ_l is larger than that at the right neighbor θ_r, i.e. $H_{\theta_l} \ge H_{\theta_r}$. Otherwise, the right neighbor is added to segment G. During the extending process, one of the left or right neighbors locating at the end points of sub-segment is selected to be an element of segment by comparing their histogram values. The

extending process will terminate, if the length of segment is equal to a given value.

Consider a segment G of length |G| in histogram H, the segment G is assigned to one of three types according to the following rules.

$$Type\ of\ segment G\ T(G)=\begin{cases} L & if \sum_{i\in[\theta-\theta_1,\theta+\theta_2]} H_i \ge \Phi_L, and |G| = \theta_L = \theta_1+\theta_2+1 \\ M & if \sum_{i\in[\theta-\theta_1,\theta+\theta_2]} H_i \ge \Phi_M, and |G| = \theta_M = \theta_1+\theta_2+1 \\ S & if \sum_{i\in[\theta-\theta_1,\theta+\theta_2]} H_i \ge \Phi_S, and |G| = \theta_S = \theta_1+\theta_2+1 \\ U & otherwise \end{cases}$$

where values θ_1 and θ_2 are two positive integer. In our approach, the threshold values are selected according to the type of segment as stated below.

$$\begin{cases} \Phi_L = 0.85 \mid H \mid, \theta_L = \alpha \bullet \Phi_L \\ \Phi_M = 0.5 \mid H \mid, \theta_M = \alpha \bullet \Phi_M \\ \Phi_S = 0.3 \mid H \mid, \theta_S = \alpha \bullet \Phi_S \end{cases}$$

Here, value α is a fixed value and defined to be $Sin(\pi/8)$ for the tolerance of the slant strokes in the original image. Furthermore, the weight of segment w(G) is also defined as 4,2, and 1 when the type of segment T(G) is L,M, and S, respectively.

Next, let us generate the code string of a character which is represented by symbols L, M, and S. First, we find the highest peak θ in histogram H, and initialize the segment to the point θ whose state is set to be U as shown in Fig. 4. The pixel is extended to a larger segment of length θ_S. The rule of type S is checked. If the segment satisfies the rule of type S, record all information of current segment, and set the state to node S. Otherwise, the current segment stays at its original state, i.e., state U. Then, we continue the extension process for the rules of types M and L. The new larger segment of length θ_M is obtained by the extending process. If this current segment satisfies the rules of type M, store all data of current segment and the segment comes to state M. Otherwise, the segment stays at the original state, i.e., state U or S. Similarly, the rules of type L will be checked after the checking processes of types M and S. Finally, the segment is assigned to the type of the last state, and set all histogram values of this segment of the assigned type to 0.

After the code generation process, two code strings C_h and C_v are obtained from the two histograms of p-skeleton H_h and H_v, respectively. In additional to two code stings C_h and C_v, three statistical features are designed for the pre-filtering process. These three statistical features f_1 (the relative pixel number of p-skeleton), f_2 (the weighted value of code string C_h), and f_3 (the weighted value of code string C_v) can be generated from the following formula.

$$f_1 = \sum_{i\in H_h} H_i / I_w + \sum_{i\in H_v} H_i / I_h, \quad f_2 = \sum_{G\in H_h} w(G), \quad f_3 = \sum_{G\in H_v} w(G)$$

In the above features, features f_1, f_2, and f_3 are generated from the statistical results of code strings.

For example, the statistical features and code strings of character 王 as shown in Fig. 3 can be generated as f_1 =3.55 , f_2=10, f_3 =2, C_h=LML, C_v=M after the extraction process.

3.Characters Classification

In this section, we classify Chinese characters by using the code strings C_h and C_v of characters. In the training phase, the reference features stored in the database are created based on the extracted code strings in feature extraction. All the training samples of a specified font are processed to get their code strings and store them in the database. After creating the reference database, an unknown sample of different font is tested by our purposed methods. When an unknown character is inputted, three statistical features and two code strings are obtained in the feature extraction process. The pre-filtering process using these three statistical features is first performed. Then, the code string matching process is executed by using the edit-distance algorithm. The details will be stated as follows.

3.1.Pre-filtering process

Since the size of reference database of Chinese characters set is very huge, it needs tremendous time in finding the candidates. The pre-filtering process thus adopted to reduce the number of matching characters in the later code string matching process. Comparing features f_1, f_2, and f_3 of the unknown character with those features in the reference database, the differences of f_1, f_2, and f_3 between candidate characters and unknown pattern must be smaller than threshold values say as t_1, t_2, and t_3. After the pre-filtering process, the size of candidate set is drastically decreased, and the time needed in the code string matching process is rapidly decreased accordingly.

3.2.Code-string matching

In this section, an edit-distance algorithm [5,6], which is implemented based on the dynamic programming technique, is adopted to measure the similarity of code strings between an unknown character and those in the candidate set. The edit-distance algorithm is a well-known approach to compute the cost that transfers a string to another string. During transferring a source string ***A*** to another target string ***B***, three editing operations including insertion (**I**), deletion (**D**), and replacing (**R**) are performed on the original string ***A***. Each operation in the editing sequence needs a

cost to achieve the transformation. The main goal of edit-distance algorithm is to find the sequence operation with the minimal cost. The minimal cost is considered as the effort to convert the code string of test character to the code string of template characters. For example, if string ***A***='LML' is transferred to a new string ***B***='MLL' by applying the sequence of operations on string ***A*** such as replacing symbol L by M at the first position, and replacing symbol M by L at the second position as shown in Fig. 5(a). The editing sequence is expressed as $\mathbf{R}_{L\to M}$, $\mathbf{R}_{M\to L}$. Another operating sequence as shown in Fig. 5(b) can be executed by deleting the first symbol L, and then inserting a new symbol L after symbol M at the second position. Here, the costs of operations (insert L, insert M, delete L, delete M, replace L with M, replace M with L) are defined respectively as (I_L, I_M, D_L, D_M, R_{LM}, R_{ML}). Thus, the cost needed in Fig. 5(a) and Fig. 5(b) are $V_1= R_{LM}+ R_{ML}$ and $V_2=D_L+I_M$. The minimal cost of editing-distance for transferring string ***A*** to string ***B*** is obtained as follows.

$$V = \min_i V_i \qquad \forall i \in \{1,2,3,...\}$$

As stated in the previous section, two code strings C_h and C_v represent the feature of a character image. They represent the structure of character by using the long (L), middle (M), short (S) strokes. Two edit-distances V_h and V_v are computed from the code strings C_h and C_v of test characters and the code strings of template characters in the database, respectively.

In traditional edit-distance algorithm, the sub-costs of operations are the same. It dose not consider the operation content. In our proposed approach, the code string represents the features of Chinese characters by using the three kinds of p-strokes. The operating character of content is considered. Let us formally define the editing distance as follows:

Definition

Cost V: the cost of transferring string ***A*** to string ***B***

Operation type: **deletion**, **insertion** and **replacement** a character

Cost of single operation:

Delete a character (L,M,S): (wL,wM,wS)

Insert a character (L,M,S): (wL,wM,wS)

Replace a pair of characters ((L,M),(M,L),(L,S),(S,L),(M,S),(S,M)) : (wLM,wML,wLS,wSL,wMS,wSM)

Here, the replacing operations in our approach are defined as the same cost when character X is replaced with Y and vice versa.

The edit-distance algorithm is described as follows:

algorithm: edit distance (dynamic programming)
Input: Strings A[m],B[n]
Output: Cost V[m,n]

Step 1 :

$$V[0,0]=0$$
$$V[0,j]=V[0,j-1]+\begin{cases} wL & \text{if B[j]='L'} \\ wM & \text{if B[j]='M'} \\ wS & \text{if B[j]='S'} \end{cases}$$
$$V[i,0]=V[i-1,0]+\begin{cases} wL & \text{if A[i]='L'} \\ wM & \text{if A[i]='M'} \\ wS & \text{if A[i]='S'} \end{cases}$$

Step 2

$$X=V[i-1,j]+\begin{cases} wL & \text{if A[i]='L'} \\ wM & \text{if A[i]='M'} \\ wS & \text{if A[i]='S'} \end{cases}$$

$$Y=V[i,j-1]+\begin{cases} wL & \text{if B[j]='L'} \\ wM & \text{if B[j]='M'} \\ wS & \text{if B[j]='S'} \end{cases}$$

$$Z=V[i-1,j-1]+\begin{cases} 0 & \text{if A[i]=B[j]} \\ wLM & \text{if } conditionA \\ wLS & \text{if } conditionB \\ wMS & \text{if } conditionC \end{cases}$$

$$V[i,j]=\mathbf{min}(X,Y,Z)$$

conditionA=(A[i]='L' and B[j]='M')or(A[i]='M' and B[j]='L')
conditionB= (A[i]='L' and B[j]='S')or(A[i]='S' and B[j]='L')
conditionC= (A[i]='S' and B[j]='M')or(A[i]= 'M' and B[j]='S')

In the edit-distance algorithm, the length of two code strings may be different. Here, the dynamic programming technique is utilized to reduce the computational time. All minimal transferring costs for the code strings of an unknown character with those of pre-filtered characters in the database are calculated by using the edit-distance algorithm. There are two edit-distances for each test character which are computed from C_h and C_v. Sort all costs between the test character and the template characters, and the candidates with smaller costs are selected as the classification results. Generally, the smaller the edit-distance between an unknown character and the candidate characters, the larger the similarity. The classification results are the selection of candidate set in which all edit-distances are smaller than a given threshold value V_θ. The ranking criterion is defined as the summation of the two edit-distances of code strings C_h and C_v.

4.Experimental results

In our experiments, the features of reference database are constructed from the character images of 5401 Chinese characters of Ming font with image size 41×41. The complete classification process of an unknown character is demonstrated in section 4.1. The experiments conducted on the large character sets are given in section 4.2.

4.1. The classification process

In this section, we will describe how to classify the

test character 左, whose font type is Ming font or Kai font with image size being set to be 41×41, or 132×132. When the font type and font size of the testing character are the same as those in the reference database, the classification results are shown in Fig. 6. In this figure, the code strings C_h and C_v of the test character are the same as those of the first candidate character. However, when the font type and size of test character are different from those in the reference database, the code strings C_h and C_v for p-skeleton should be different. The classification results for various font types and sizes of test characters are shown in Fig. 7, 8, and 9, respectively.

4.2. Statistics of experimental results

As mentioned in section 4.1, the reference database consists of 5401 Chinese characters of Ming font with image size 41×41 in our experiments. 5401×6 testing samples of 6 kinds of fonts and sizes are conducted to evaluate the performance of our proposed methods. The experimental results are tabulated in Table 1. In this table, the first column denotes the test characters of Ming (M) font and Kai (K) font. The test characters of various image sizes are listed at the second column. The numbers of failure characters that do not appear at the top twenty ranks are tabulated at the third column. The rest columns at the right side of table 1 represent the character numbers of successful cases from rank 1 to rank 20. In table 1, the accuracy rate of classification results of the first rank is 100% in the same font type and the same font size conditions. The accuracy rate of classification results is more than 98.7% at top 10 ranks, in the same font type but different font size conditions. When the font type is different, the accuracy rate is more than 95% at top 20 ranks.

The character strokes of Kai font are more skew and shorter than those of Ming font. It will leads to the lower accuracy rate. On the other hand, the failure case illustrated in table 2 represent the characters of more complex structure. The projection histogram of these characters cannot efficiently represent the stroke structure.

5.Conclusions

In this paper, we present a classification method using the p-skeleton features. Based on the p-skeleton features, the characters of various fonts and sizes can be classified by making of the templates of a specified font. In the classification stage, we modify the edit-distance algorithm to measure the similarity between two Chinese characters. In the experiments, 5401 × 6 characters of six kinds of fonts and sizes are conducted to evaluate the validity of our proposed approach. The recognition accuracy is larger than 95%.

Although the font type and size of characters are varied between the reference database and test characters, there exist some failure cases in our experiments. We plan to use the meshing technology or the more detail histograms to improve the accuracy in the future.

References

[1]O.D. Trier, A.K. Jain, and R. Taxt, “Feature Extraction Methods for Character Recognition -- A Survey,” Pattern Recognition, vol. 29, no. 4,pp. 641-662, 1996.

[2]S. Mori, C.Y. Suen, and K. Yamamoto, “Historical Review of OCR Research and Development,” Proc. IEEE, vol. 80, pp. 1029-1058, 1992.

[3]G. Nagy, “At the Frontiers of OCR,” Proc. IEEE, vol. 80, pp. 1093-1100,1992.

[4]Chin-Chuan Han, Kuo-Chin Fan, and Yao-Lung Tseng (1995), "Coarse classification of Chinese characters via stroke clustering method," Pattern Recognition Letters, Vol. 16, pp. 1079-1089, October 1995.

[5]Wanger R. A., and M. J. Fischer, “The string-to-string correction problem,” Journal of the ACM, 21, pp.168-173

[6]Eric Sven Ristad, and Peter N. Yianilos, “Learning String-Edit Distance,” IEEE Trans. On Pattern Analysis and Machine Intelligence, vol. 20, no. 5, pp. 522-532, May 1998.

[7]Kuo-Chin Fan, Ming-Gang Wen, and Deng-Fong Chen, "Skeletonization of binary images with nonuniform width using block decomposition and contour vector matching method," Pattern Recognition, Vol. 31, No. 7, pp. 823-838, July 1998.

[8]T. Pavlidis, "A thinning algorithm for discrete binary images," Comput. Graphics Image Process. 13, 142-157 (1980).

[9]Y. S. Chen and W. H. Hsu, A modified fast parallel algorithm for thinning digital patterns, Pattern Recognition Letter 7, 99-106 (1987).

[10] B. K. Jeng and R. T. Chin, One-pass parallel thinning: analysis, properties, and quantitative evaluation, IEEE Trans. Pattern Anal. Mach. Intell. 14(11), 1120-1140 (1992).

Table 1. The classification results of various fonts and sizes.

Font	Size	failure	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
M20	33X33	0	1883	862	787	646	464	331	157	83	77	43	27	22	9	6	2	2	0	0	0	0
M24	40X40	0	5401	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
M28	47X47	0	2555	884	699	504	350	186	121	57	17	10	10	5	1	2	0	0	0	0	0	0
K20	34X33	118	100	157	311	442	482	360	329	376	311	296	294	266	230	200	193	176	143	123	125	112
K24	40X40	125	97	157	306	394	471	388	384	370	338	287	280	250	245	212	196	175	133	142	110	105
K28	48X47	91	97	157	306	394	471	388	384	370	338	287	280	250	245	212	196	175	133	142	110	105

Table 2. The misclassification characters of font K28 in Table 1.

禿啜委放泗肩肯峒彥撢毒看趴唧夏爹梟芻衰貢
邕啊婁專裊書莖貫章壹惠景畫筆絮臺萋菴萇費
貴買傷募彙睪腳舅葷賈墓實煽磁臺蓆慧暫碾翩
賢賣霄曇熹縈羲蕈蕭雯賽檯燾薹闐鯉璽賓薊罍
囊靡蠢轟響鼙疊蘭纖矗鸞

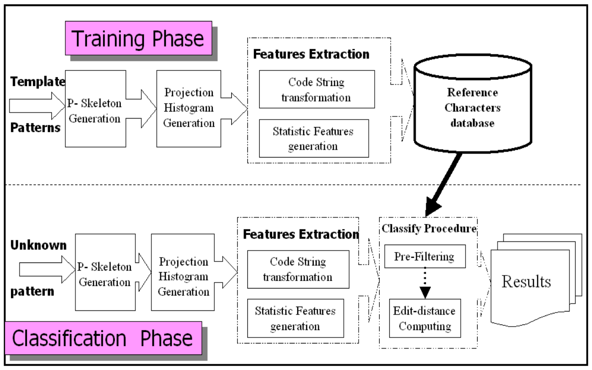

Fig. 1 System overview

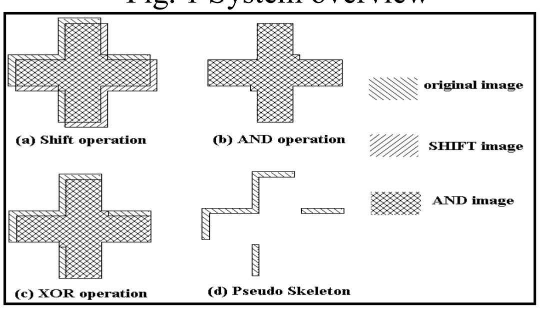

Fig. 2 The extraction of P-skeleton.

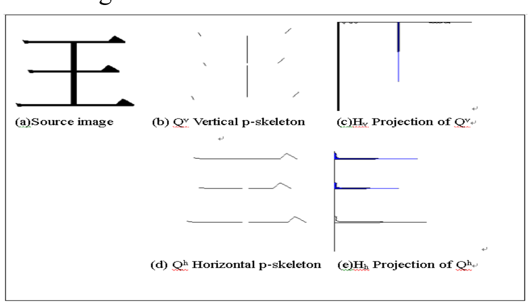

Fig. 3 The generation of p-skeletons Q^v and Q^h and their projections.

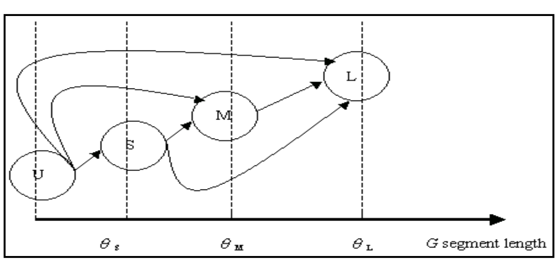

Fig. 4 States of Code string generation.

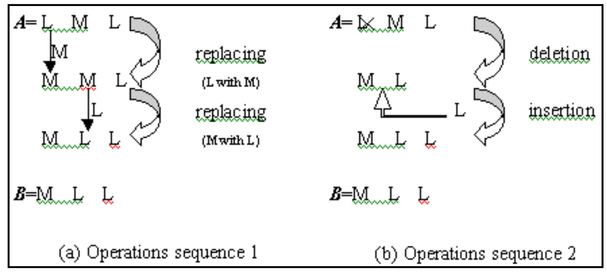

Fig. 5 Examples of edit-sequence.

Fig. 6 Experimental result 1. The test character of Ming font and size 41 by 41.

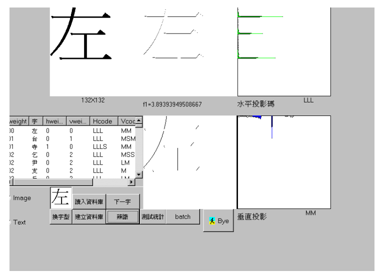

Fig. 7 Experimental result 2. The test character of Ming font and size 132 by 132.

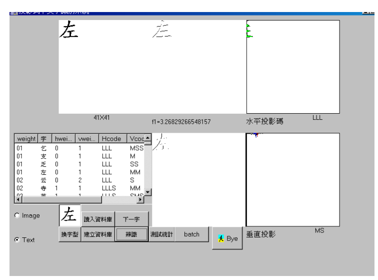

Fig. 8 Experimental result 3. The test character of Kai font and size 41 by 41.

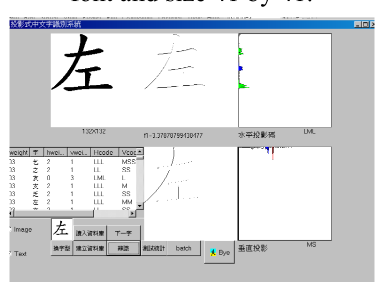

Fig. 9 Experimental result 4. The test character of Kai font and size 132 by 132.