

Classification of Chinese Characters Using Pseudo Skeleton Features

MING-GANG WEN^{*,†}, KUO-CHIN FAN^{*} AND CHIN-CHUAN HAN

**Institute of Computer Science and Information Engineering*

National Central University

Chungli, 320 Taiwan

E-mail: kcfan@csie.ncu.edu.tw

†Department of Information Management

Department of Computer Science and Information Engineering

National United University

Miaoli, 360 Taiwan

E-mail: mgwen@nuu.edu.tw

In this paper we present a novel method to classify machine printed Chinese characters by matching the code strings generated from pseudo skeleton features. In our approach, the pseudo skeletons of Chinese characters are extracted rather than using skeletons extracted by traditional thinning algorithms. The features of the pseudo skeletons of both input and template characters are then encoded into two code strings. Finally, the edit-distance algorithm is employed to compute the similarity between the two characters based on their corresponding encoded strings. The main contribution of this paper is to effectively classify multi-fonts Chinese characters using a 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 method for classifying Chinese characters.

Keywords: optical character recognition (OCR), coarse classification, pseudo skeleton, projection histogram, edit distance algorithm

1. INTRODUCTION

Compared to the recognition of letters of the Latin alphabet or Arabic numerals, the recognition of Chinese characters is significantly more difficult for several reasons. First, there are a great many more Chinese characters. There are 5401 daily-used Chinese characters and more than 47000 characters in the Kang-Xi Chinese dictionary. This results in a very large reference database, which leads to a time-consuming recognition process [1-4]. Second, the structure of Chinese characters is considerably more complex than that of Latin letters and Arabic numbers. The number of strokes and intersection points of Chinese characters are often more than ten, increasing the complexity of feature extraction and also increasing the database size and processing time. Third, there are many font types used for Chinese characters. The recognition performance for characters of various fonts decreases rapidly when using a reference database for a single specified

Received February 11, 2003; revised April 30, 2003; accepted May 20, 2003.

Communicated by H. Y. Mark Liao.

font. To remedy this problem, many OCR systems utilize pre-filtering techniques to detect the font types of Chinese characters before recognition. However, these approaches must have many reference databases for the characters of various font types built in advance.

By carefully analyzing the problems in designing Chinese OCR systems, it is easy to find drawbacks, such as being time-consuming, requiring a large memory space, and more complex implementation. Coarse classification is an efficient strategy to reduce the number of candidates so as to reduce the recognition time. Characters in the smaller candidate sets can thus be recognized by using simpler algorithms. The use of effective features to efficiently classify Chinese characters is the main goal of our work.

Statistical and structural methods are two major approaches in OCR systems. Statistical methods use mathematical features, such as pixel number, projection histogram, and crossing count number to classify the character. Similarity calculation, neural networks, and template matching are popular matching techniques adopted in statistical methods. The major advantage of statistical methods is faster recognition speed. However, the recognition accuracy will drop off when the character set is huge or character font type vary very much. On other hand, the types of features used in structural methods include topological, geometrical, directional, and structural features. Syntactic rules inference and graph matching are typical techniques for recognition [12]. High accuracy and a human-like recognition model are the main advantages of these methods. On the other hand, difficult feature extraction, design complexity, and time-consuming processing are drawbacks of structural methods. Recently, statistical and structural approaches have been combined for character recognition [13, 14] to improve classification performance.

In many optical character recognition systems, character skeletons play crucial roles in both syntactic and statistical approaches. Skeletons extracted by traditional thinning algorithms are commonly used features for classification [8]. However, these features often suffer from many unwanted effects, such as spurious branches, the splitting of a 4-fork point into two 3-fork points, and so on. Moreover, it is time consuming [9, 10]. However, people fluent in Chinese characters can recognize them merely from the contours of the characters. In this paper, we propose a novel method based on the contours of characters called pseudo skeletons (abbreviated as p-skeletons) from which the features of characters are extracted to effectively classify Chinese characters.

Our proposed approach consists of three modules including p-skeleton generation, code string generation, and matching modules as shown in Fig. 1. In our approach, the p-skeleton of an input character is generated first in the p-skeleton generation module. Next, the 2-D p-skeleton image is projected along the x and y directions to generate two profile histograms. The features used in the classification process are extracted from these two histograms. These features are then encoded and represented by three kinds of codes in the code string generation module. Based on these code strings, the OCR classification problem is transformed into a 1-D string matching problem instead of a 2-D image classification problem. In the training phase, the extracted features are stored in the reference database. In the classification phase, the edit-distance algorithm is applied to measure the similarity between an unknown pattern and those in the reference database. Finally, the character candidate list is generated in the classification process.

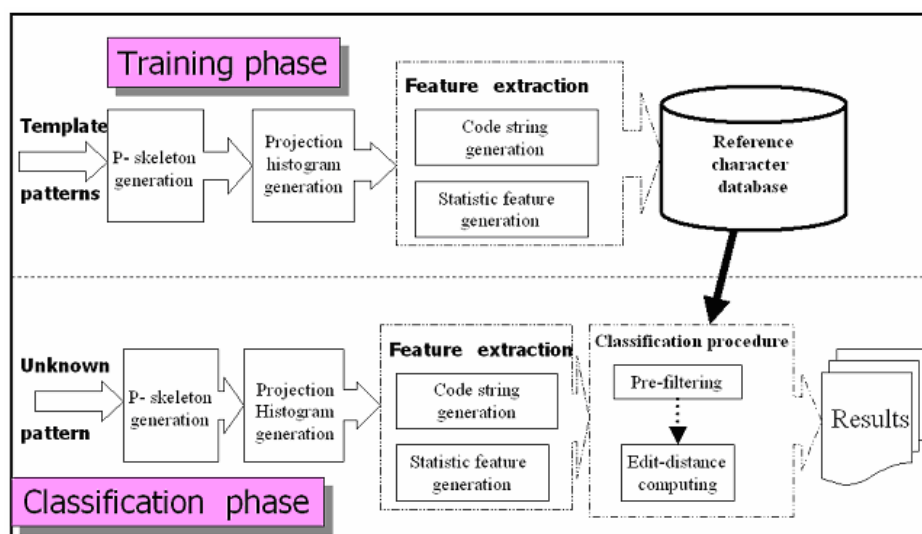


Fig. 1. System overview.

The rest of this paper is organized as follows. In section 2, we present the extracting technique of 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

Feature extraction is one of the key issues in pattern recognition. In this section, we present the proposed feature extraction method to extract the features of character images for coarse classification. In the feature extraction process, the task of p-skeleton generation, p-skeleton projection, and code string encoding are executed to obtain the features of an input character. There are three advantages in our proposed feature extraction method. First, the normalization procedure operating on character images is not needed. Generally, many statistical OCR methods perform a size normalization procedure on the images of different sizes before thinning to resolve the scale variation problem. Second, the p-skeletons of characters replace the skeletons generated by traditional thinning algorithms. Hence, the thinning procedure of characters is also omitted to save the execution time in preprocessing. Finally, the 2-D image features are transformed to two 1-D code string features. We can apply the edit-distance algorithm directly on these code strings to easily measure the similarity between two Chinese characters in the classification process. Three useful features for the pre-filtering process, including the relative skeleton pixel count, the pseudo stroke count of horizontal pseudo stroke, and the pseudo stroke count of vertical pseudo stroke are also extracted from the p-skeleton.

In the following contexts, the detailed descriptions of feature extraction embedded in character images will be stated.

2.1 P-Skeleton Generation

The skeletons of character images represent both the semantics and structure of Chinese characters. Several researchers have presented algorithms to extract the skeleton from an image. The simplest skeleton extraction approach is the thinning algorithm [7-9]. Traditionally, the thinning of character images is the preprocessing step in many image processing applications [1, 4]. A traditional thinning algorithm repeatedly scans an image from left to right and top to bottom to successively remove one pixel from the boundary until there is only a one pixel wide skeleton left. However, the algorithm often leads to unwanted effects, such as spurious branches, the splitting of a 4-fork point into two 3-fork points, and so on. The contour-matching skeletonization algorithm is an alternative approach to solve these problems [7]. No matter whether a thinning-based or a skeletonization-based algorithm is used, it takes tremendous amount of time to extract the complete skeletons of character images. However, people can identify the character merely based on its contour. So, actually, it is not necessary to extract the complete skeletons of characters in the recognition process. In this paper, a novel skeleton, call p-skeleton, is devised to remedy the time-consuming drawback inherent in traditional thinning-based or skeletonization-based algorithms.

Generally, p-skeletons are defined as a subset of contour pixels, which can be extracted from character images by using some simple logic operations. The advantage of this approach is that tremendous amount of time in obtaining thinned images can be saved. Besides, it can be easily implemented by hardware. In our work, all features of characters are extracted from the p-skeleton directly to classify the Chinese characters. The extraction process of p-skeleton is stated as follows.

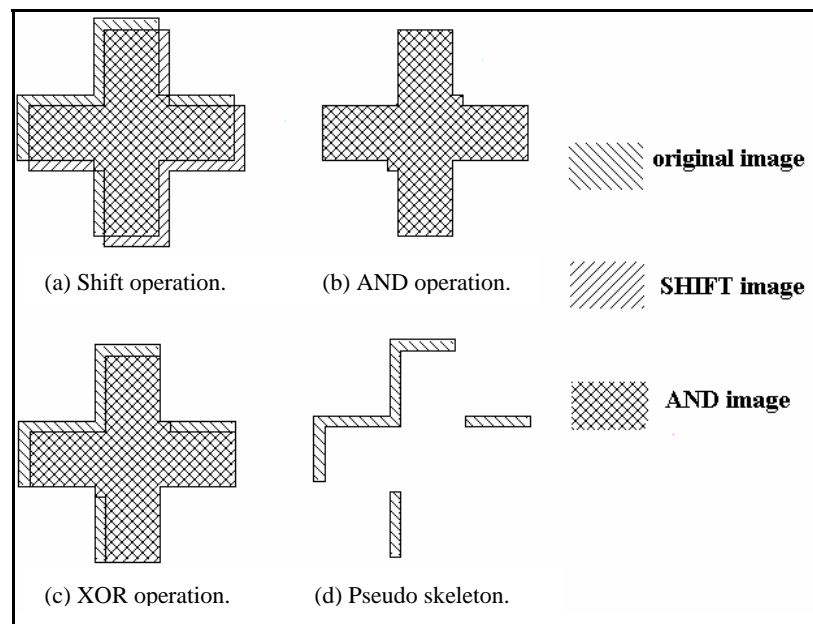


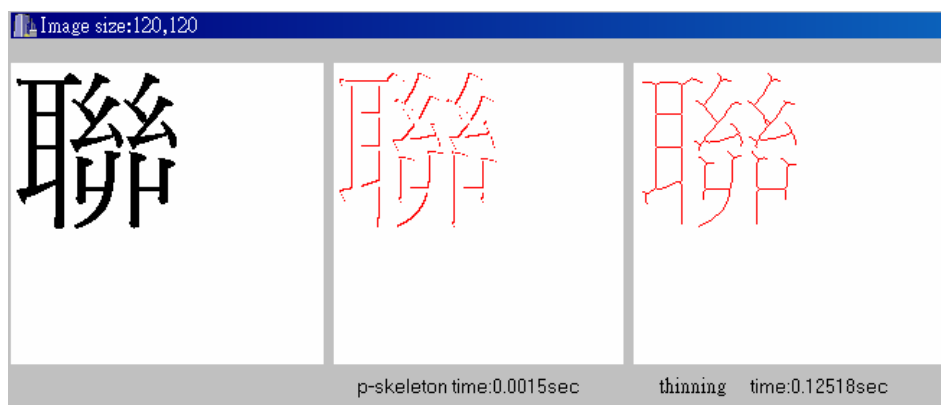
Fig. 2. Illustration of p-skeleton extraction.

From a character image A , the shifted image A' is obtained by shifting the original image A one pixel right and one pixel down as shown in Fig. 2 (a). Next, the logical operation AND (denoted by symbol \wedge) is applied to the shifted and original images (Fig. 2 (b)). The ANDed image $A \wedge A'$ is then exclusive-ORed (denoted by symbol \oplus) operation pixel by pixel with the original image to obtain the p-skeleton image A^\oplus as shown in Fig. 2 (c). The pixel Q_{ij} of the p-skeleton image can be generated from pixels P_{ij} and $P_{i-1,j-1}$ in the original images by the following logical operation:

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

Here, the subscripts in Eq. (1) denote the locations of pixels. After performing the above operations, only the contour pixels at the top or left side of the character image remain (Fig. 2 (d)). These pixels, called pseudo-skeleton or p-skeleton, still preserve the information of the character strokes. They can easily be identified by human beings. The skeleton extraction process can be simply implemented merely by the three logic operations SHIFT, AND, and XOR.

To illustrate the benefit of p-skeleton, a comparison was made between the proposed p-skeleton and the skeleton generated by a typical thinning algorithm [11]. Experiments were conducted on a 366MHz Pentium II computer. Fig. 3 shows an example illustrating the comparison of execution times of p-skeleton extraction and the thinning process from the character 聯 with size 120 by 120. From the experimental results, we know that the proposed p-skeleton algorithm is faster by a factor of about 83 compared to the traditional thinning algorithm of reference [11]. This speedup is not surprising because the p-skeleton algorithm does not use iteration, no matter how wide the pattern is. The traditional thinning algorithm must strip away pixels layer by layer.



(a) Source image. (b) The p-skeleton. (c) The thinning skeleton.

Fig. 3. Comparison of the skeletons generated by the p-skeleton and thinning approaches.

2.2 Projection by P-Skeleton

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

(a) Vertical p-skeleton

Consider pixel $P_{i,j}$ in a character image as shown in Fig. 4 (a), pixel $Q_{i,j}^v$ of the vertical p-skeleton can be generated by modifying operation Eq. (1) as follows.

$$\left. \begin{aligned} P'_{i,j} &= P_{i-1,j}; && \text{(SHIFT)} \\ P^*_{i,j} &= P_{i,j} \wedge P'_{i,j}; && \text{(AND)} \\ Q^v_{i,j} &= P_{i,j} \oplus P^*_{i,j}, && \text{(XOR) for } 0 \leq i \leq I_w, 0 \leq j \leq I_h \end{aligned} \right\} \quad (2)$$

where I_w is the image width and I_h is the image height. Fig. 4 (b) illustrates the vertical p-skeleton image which is generated from Fig. 4 (a).

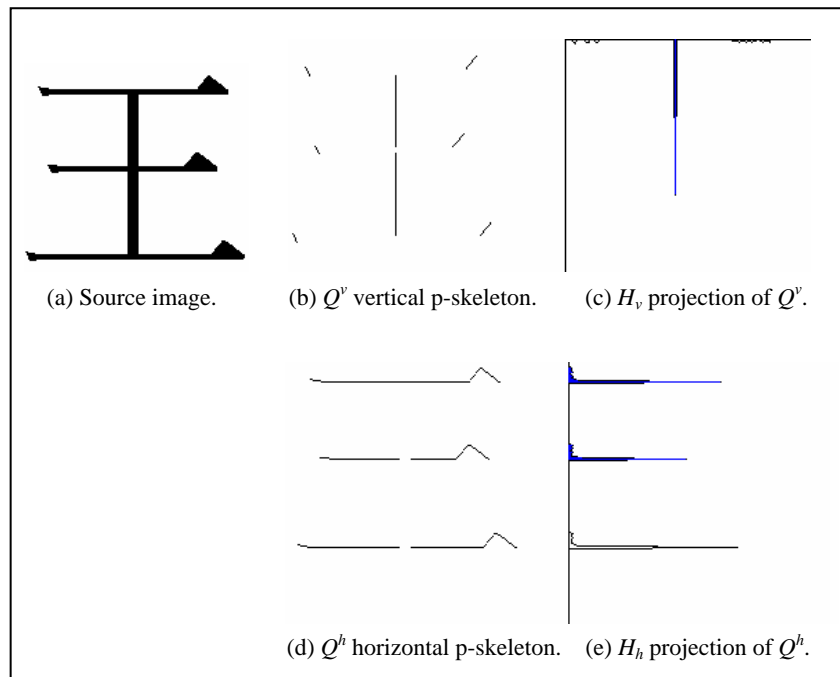


Fig. 4. The generation of p-skeletons Q^v and Q^h and their projections.

(b) Horizontal p-skeleton

Similarly, the horizontal p-skeleton image $Q_{i,j}^h$ can be generated by the following.

$$\left. \begin{aligned} P'_{i,j} &= P_{i,j-1}; & (\text{SHIFT}) \\ P''_{i,j} &= P_{i,j} \wedge P'_{i,j}; & (\text{AND}) \\ Q_{i,j}^h &= P_{i,j} \oplus P''_{i,j}, & (\text{XOR}) \text{ for } 0 \leq i \leq I_w, 0 \leq j \leq I_h \end{aligned} \right\} \quad (3)$$

Shown in Fig. 4 (d) is the horizontal p-skeleton image generated from Fig. 4 (a). Shown in Figs. 4 (c) and 4 (e) are the vertical projection histogram H_v and horizontal projection histogram H_h generated from the two p-skeleton images of Figs. 4 (b) and 4 (d), respectively. The set equation for the histogram is $H = \{h_i \mid \text{the pixel count of p-skeleton at position } i, \text{ where } 0 \leq i \leq \text{image size}\}$.

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 structural features of the Chinese character. The process of string encoding is as follows. Consider the projection pixel count 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 that $H_\theta = \max_{j \in [0, |H|]} H_j$, where value $|H|$ is the number of bins 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. $H_{\theta_l} \geq H_{\theta_r}$. Otherwise, the right neighbor is added to segment G , if $H_{\theta_l} < H_{\theta_r}$. During the extending process, one of the left or right neighbors located at the end points of the sub-segment is selected to be an element of the segment by comparing their histogram values. The extending process will terminate if the length of the segment is equal to a given threshold 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.

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

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

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

Here, α is a fixed value and defined to be $\sin \frac{\pi}{8}$ for the tolerance of the slant strokes in the original image. Furthermore, the weight of the segment is also defined in the following definition depending on the type of segment.

$$w(G) = \begin{cases} 4, & \text{if } T(G) = L \\ 2, & \text{if } T(G) = M \\ 1, & \text{if } T(G) = S \\ 0, & \text{otherwise} \end{cases}$$

The assignment of segments to type L involves a higher priority than the other types because the constraints for type L are stricter than those for types M and S . For the same reason, the assignment for each segment of type M should be executed before that of type S .

The major step in feature extraction for our approach is to convert the p-skeleton histogram of 2-D images into two 1-D code strings. Now we describe how we use a state machine to generate the code string of a character which is represented by symbols L , M , or S . During the conversion process, the criterion for type S should be checked by using a bottom-up strategy because the segment length of type S is the shortest of the three types. First, we find the highest peak θ in histogram H and initialize the segment to the point θ whose state is set to U as shown in Fig. 5. The pixel is extended to a larger segment of

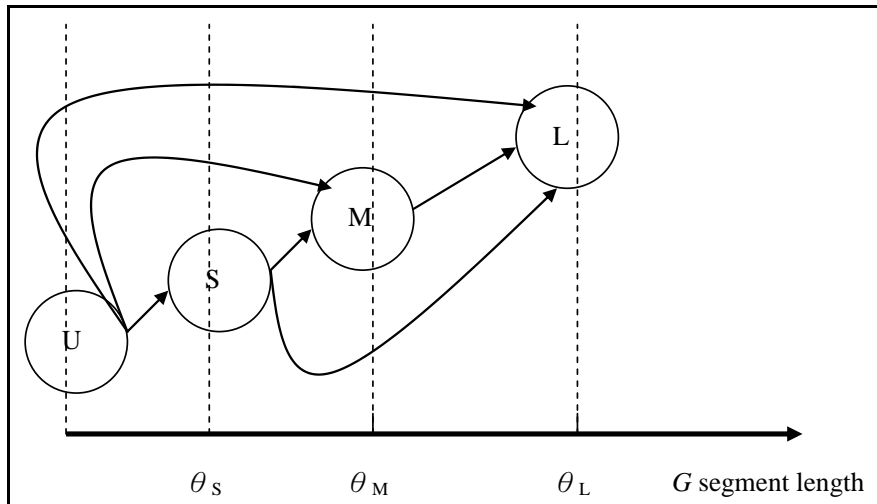


Fig. 5. States for code string generation corresponding to segment type.

length θ_s . The rule of type S is checked, and if the segment satisfies the rule, all information of the current segment is recorded and the state is set to S . Otherwise, the current segment remains in its original state, i.e., state U . Then, we extend the process for the rules of types M and L . The new larger segment of length θ_M is obtained by extending the process. If this current segment satisfies the rules for type M , store all data of the current segment, and proceed to state M . Otherwise, the segment stays in the original state, i.e., states U or S . Similarly, the rules for type L are checked after having checked the rules for types M and S . Finally, the segment is assigned to the type of the final state, and set all histogram values of this segment of the assigned type to 0.

The algorithm for code string generation is given below.

Input: Horizontal projection histogram H_h (or vertical projection histogram H_v)

Output: code string array $Hcode[]$ of H_h (or $Vcode[]$ of H_v)

Algorithm: code string generation

Step 1: Find the location θ which has the maximum value H_θ in unmarked H_h (or H_v).

Set the value at θ to $Total_Pixel$ and mark it as segment G

Step 2: If can't find θ , then goto Step 9

Step 3: Select the larger value from the neighbor of G , mark it and add it to G and add the value in H_h (or H_v) to $Total_Pixel$

Step 4: If $length(G) < rangL$, then goto Step 3

else if $Total_Pixel \geq thresholdL$, then

$code[\theta] = 'L'$

goto Step 1

else

reset (G) as unmarked and set $Total_Pixel$ as H_θ

Step 5: Select the larger value from the unmarked neighbor of G , mark it and add it to G and add the value in H_h (or H_v) to $Total_Pixel$

Step 6: If $length(G) < rangM$, then goto Step 5

else if $Total_Pixel \geq thresholdM$, then

$code[\theta] = 'M'$

goto Step 1

else

reset (G) as unmarked and set $Total_Pixel$ as H_θ

Step 7: Select the larger value from the neighbor of G , mark it and add it to G and add the value in H_h (or H_v) to $Total_Pixel$

Step 8: If $length(G) < rangS$, then goto Step 7

else if $Total_Pixel \geq thresholds$, then

$code[\theta] = 'S'$

goto Step 1

Step 9: Set $Hcode[]$ (or $Vcode[]$) = $code[]$

return

After the code string 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 addition to two code

strings C_h and C_v , three statistical features are designed for the pre-filtering process. These three statistical features f_1 , f_2 , and f_3 can be generated according to the following formula.

$$f_1 \text{ (the relative pixel count of p-skeleton)} = \sum_{i \in H_h} H_i / I_w + \sum_{i \in H_v} H_i / I_h$$

$$f_2 \text{ (the weighted value of code string } C_h) = 4 \times |\text{the count of code } L \text{ in } C_h| + 2 \times |\text{the count of code } M \text{ in } C_h| + |\text{the count of code } S \text{ in } C_h| = \sum_{G \in H_h} w(G)$$

$$f_3 \text{ (the weighted value of code string } C_v) = 4 \times |\text{the count of code } L \text{ in } C_v| + 2 \times |\text{the count of code } M \text{ in } C_v| + |\text{the count of code } S \text{ in } C_v| = \sum_{G \in H_v} w(G)$$

where I_w is the image width and I_h is the image height.

Actually, features f_1 , f_2 , and f_3 are generated based on the statistical results of code strings.

For example, the statistical features and code strings of character 王 as shown in Fig. 4 are $f_1 = 3.55$, $f_2 = 10$, $f_3 = 2$, $C_h = LML$, $C_v = M$ after performing the feature extraction process.

3. CHARACTER CLASSIFICATION

Before presenting the classification process, three properties of the features used in this paper are briefly described. First, p-skeleton is a new and efficient skeleton-like feature for the pre-classification process. The main advantages of the proposed method are its efficiency and robustness and its ease of implementation. Second, since the matching rules are derived based on the projection of the p-skeleton, the classification results will not be affected even under the situation of broken strokes. Third, since the slant angle of slant strokes in different font types are not very stable, it is not necessary to generate slant p-skeletons and their corresponding projection histograms. Another reason is that slant strokes already contribute different weights in the vertical and horizontal p-skeleton histograms naturally.

In this section, we describe the classification of Chinese characters by using the code strings C_h and C_v . In the training phase, the reference features to be stored in the database are created based on the extracted code strings. All the training samples of a specified font are processed to get their code strings and store in the database. After creating the reference database, an unknown sample of a different font is tested by our proposed methods. When an unknown character is input, three statistical features and two code strings are obtained in the feature extraction process. In the classification phase, 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 given in the following subsections.

3.1 Pre-Filtering

First, the pre-filtering process is performed in the classification phase to reduce the number of candidate characters. Since the size of the reference database of Chinese characters is huge, it needs a long time to find the candidates. The pre-filtering process is thus utilized to reduce the number of matching characters to be tested in the following code string matching process. After performing the feature extraction process, three statistical features f_1 , f_2 , and f_3 of an unknown sample are extracted. The differences of f_1 , f_2 , and f_3 between candidate characters and an unknown pattern must be smaller than threshold values t_1 , t_2 , and t_3 . Table 1 demonstrates the benefits of the three pre-filtering features. The first column denotes the test characters of Ming (M) font and Kai (K) font. The different features, f_1 , f_2 , and f_3 , are applied to reduce the numbers of candidates as listed in the second column. The remaining columns represent the result of pre-filtering. The column titled as average represents the average remaining number of candidates after pre-filtering 5401 characters by using features f_1 , f_2 , and f_3 . The captions $n_1 \sim n_2$ in the table represent the remained candidates between n_1 and n_2 . The values below captions $n_1 \sim n_2$ represent the cases of testing characters remained.

Table 1. The benefits of statistical features for pre-filtering.

Font	Statistical feature	The size of candidate set											
		average	1~500	501~1000	1001~1500	1501~2000	2001~2500	2501~3000	3001~3500	3501~4000	4001~4500	4501~5000	5001~5401
M20	f_1	1723	425	595	767	1064	2494	56	0	0	0	0	0
	f_2	1688	437	623	819	1087	2403	32	0	0	0	0	0
	f_3	1665	438	647	852	1123	2321	20	0	0	0	0	0
M24	f_1	34	5401	0	0	0	0	0	0	0	0	0	0
	f_2	33	5401	0	0	0	0	0	0	0	0	0	0
	f_3	33	5401	0	0	0	0	0	0	0	0	0	0
M28	f_1	1061	901	1307	1845	1348	0	0	0	0	0	0	0
	f_2	1034	927	1361	2049	1064	0	0	0	0	0	0	0
	f_3	1017	944	1415	2118	924	0	0	0	0	0	0	0
K20	f_1	3489	275	262	337	304	361	367	382	520	582	815	1196
	f_2	3432	275	266	335	327	356	409	378	515	621	892	1027
	f_3	3415	275	272	333	340	345	406	389	526	637	964	919
K24	f_1	3426	297	274	345	350	371	376	370	474	574	822	1148
	f_2	3359	297	282	344	360	371	376	421	509	606	952	897
	f_3	3338	297	287	347	354	384	368	439	508	619	1034	764
K28	f_1	3513	290	240	337	314	355	313	382	512	569	825	1264
	f_2	3446	290	242	344	323	364	376	369	516	595	957	1027
	f_3	3421	290	249	342	329	368	375	388	519	604	1060	881

3.2 Code-String Matching

In this section, an edit-distance algorithm [5, 6], which is implemented based on dynamic programming, 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 of transferring one string into another. During transfer of a source string A to a target string B , three editing operations, including insert (I), delete (D), and replace (R), are performed on the original string A . The manipulation of edit-distance algorithm is to transfer the source string into a new string which is the same as the target string by a sequence of editing operations. Each operation in the editing sequence needs a cost to achieve the transformation. The main goal of the edit-distance algorithm is to find the sequence of operations with the minimal cost. The minimal cost is considered to be the required effort to convert the code string of test character into the code string of a template character. For example, 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. 6 (a). The editing sequence is expressed as $R_{L \rightarrow M}, R_{M \rightarrow L}$. Another operating sequence as shown in Fig. 6 (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 Figs. 6 (a) and 6 (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.

$$V = \min_i V_i \quad \forall i \in \{1, 2, 3, \dots\}$$

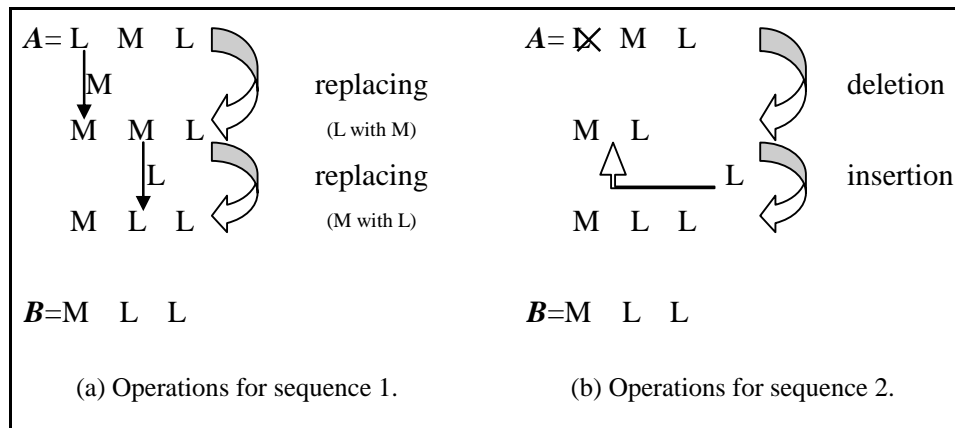


Fig. 6. Examples illustrating edit-sequence.

As stated in the previous section, the two code strings C_h and C_v represent the feature of a character image. They represent the structure of a character by using the long

(L), middle (M), and 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 the traditional edit-distance algorithm, the costs of operations are defined for deletion, insertion, and replacement operations. It does not consider the operation content, i.e., what character is operated on. In our proposed approach, the code string can represent the features of Chinese characters by using the three kinds of p-strokes. The content of the operating character is thus 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** of 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 having the same cost when character X is replaced with Y and vice versa.

The edit-distance algorithm is described as follows:

Input: Strings $A[m], B[n]$

Output: Cost $V[m, n]$

Algorithm: Edit distance (dynamic programming)

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:

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

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

$$Z = V[i-1, j-1] + \begin{cases} 0 & \text{if } A[i] = B[j] \\ wLM & \text{if } (A[i] = 'L' \text{ and } B[j] = 'M') \text{ or } (A[i] = 'M' \text{ and } B[j] = 'L') \\ wLS & \text{if } (A[i] = 'L' \text{ and } B[j] = 'S') \text{ or } (A[i] = 'S' \text{ and } B[j] = 'L') \\ wMS & \text{if } (A[i] = 'S' \text{ and } B[j] = 'M') \text{ or } (A[i] = 'M' \text{ and } B[j] = 'S') \end{cases}$$

$$V[i, j] = \min \begin{cases} X \\ Y \\ Z \end{cases}$$

In the edit-distance algorithm, the length of two code strings may be different. In our work, the dynamic programming technique is utilized to reduce the computational time. All minimal transforming 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 . All the costs between the test character and the template characters are sorted, then 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 result is the set of candidates for which the edit-distances are smaller than a given threshold value V_θ . The ranking criterion is defined as the sum of the two edit-distances for code strings C_h and C_v .

4. EXPERIMENTAL RESULTS

In this section, experimental results are given to illustrate the performance of our proposed method. In our experiments, the features in the reference database are constructed from the character images of 5401 Chinese characters of the Ming font with image size 40×40 . First, the complete classification process of an illustrative 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 illustrate the classification process using the test character 左, whose font type is the Ming font or Kai font with an image size set to 41×41 , or 132×132 . When the font type (Ming font) and font size (41×41) of the test character are the same as those in the reference database, the classification results are shown in Fig. 7. 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 the test character are different from those in the reference database, the code strings C_h and C_v for the p-skeleton should be different. The classification results for various font types and sizes of the test characters are shown in Figs. 8, 9, and 10, respectively.

In Fig. 8, the font type of the test character is set to Ming font and its image size is set to 132×132 . The code strings $C_h = 'LLL'$, and $C_v = 'MM'$ are obtained. Next, the pre-filtering features $f_1 = 3.8939$, $f_2 = 12$, and $f_3 = 4$ are generated. The candidate character set is shown at the lower-left corner of Fig. 8. The first rank of candidates contains character 左 and their corresponding edit-distances are 0.

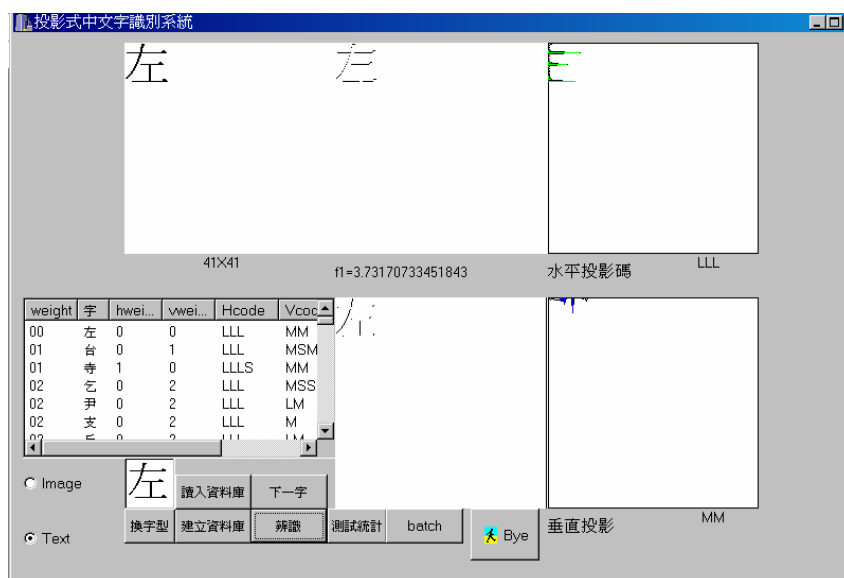


Fig. 7. Experimental result 1. The test character of Ming font and size 41 by 41.

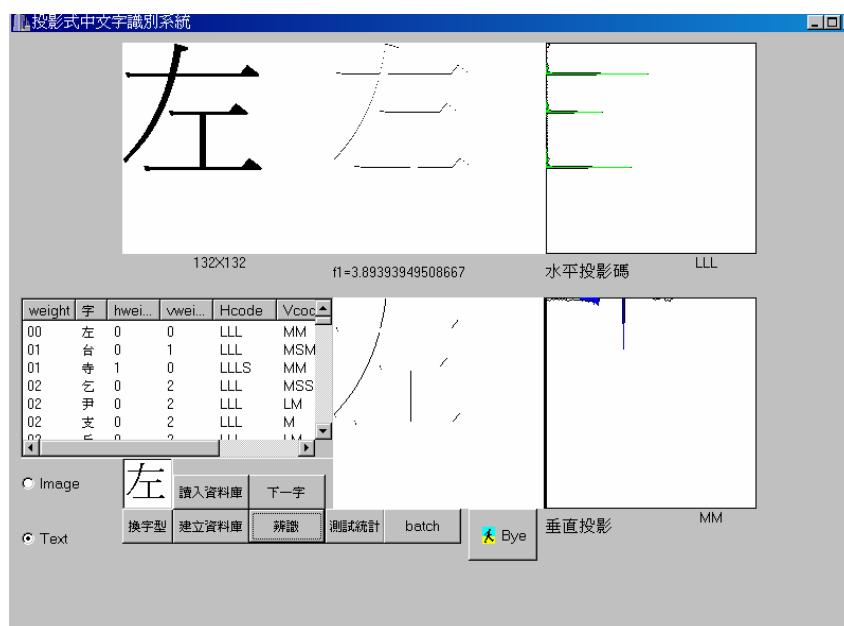


Fig. 8. Experimental result 2. The test character of Ming font and size 132 by 132.

When the font type of the test character is set to the Kai font and its image size is set to 41×41 , the classification results are tabulated in Fig. 9. Similarly, if the font type and size of the test character are set to Kai font and 132×132 , the classification results are shown in Fig. 10.



Fig. 9. Experimental result 3. The test character of Kai font and size 41 by 41.



Fig. 10. Experimental result 4. The test character of Kai font and size 132 by 132.

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 40×40 in our experiments. The characters of this font type are adopted to build up the templates because they are the most commonly used

characters in Chinese documents. For evaluating the performance of our proposed method, we use a test set consisting of two kinds of fonts, each at three different sizes of 5401 different Chinese characters of each font and size, for a total of 32406 characters. The experimental results are tabulated in Table 2. The first column denotes the test characters of Ming (M) font and Kai (K) font. The test characters of various image sizes are listed in the second column. The numbers of failures are tabulated at the third column. The remaining columns represent the character numbers of successful cases from rank 1 to rank 20. In this case, the experimental results of testing on different font types and sizes are presented. In Table 2, 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 above 98.7% in the top 10 ranks under the condition of same font type but different font size. When the font type is different, the accuracy rate is still above 93% in the top 20 ranks.

Table 2. The classification results of various fonts and sizes.

Font	Size		failure	correct classification in first <i>th</i> rank								
				1st	2nd	3rd	4th	5th	6th	7th	8th	9th
M20	33X33	numbers	0	1883	2745	3532	4178	4642	4973	5130	5213	5290
		percent	0.00%	34.86%	50.82%	65.40%	77.36%	85.95%	92.08%	94.98%	96.52%	97.94%
M24	40X40	numbers	0	5401	5401	5401	5401	5401	5401	5401	5401	5401
		percent	0.00%	100.00%	100.00%	100.00%	100.00%	100.00%	100.00%	100.00%	100.00%	100.00%
M28	47X47	numbers	0	2555	3439	4138	4642	4992	5178	5299	5356	5373
		percent	0.00%	47.31%	63.67%	76.62%	85.95%	92.43%	95.87%	98.11%	99.17%	99.48%
K20	34X33	numbers	118	100	257	568	1010	1492	1852	2181	2557	2868
		percent	2.18%	1.85%	4.76%	10.52%	18.70%	27.62%	34.29%	40.38%	47.34%	53.10%
K24	40X40	numbers	125	97	254	560	954	1425	1813	2197	2567	2905
		percent	2.31%	1.80%	4.70%	10.37%	17.66%	26.38%	33.57%	40.68%	47.53%	53.79%
K28	48X47	numbers	91	128	308	633	1106	1614	2075	2511	2896	3221
		percent	1.68%	2.37%	5.70%	11.72%	20.48%	29.88%	38.42%	46.49%	53.62%	59.64%

Font	Size		correct classification in first <i>th</i> rank																
			10th	11th	12th	13th	14th	15th	16th	17th	18th	19th	20th						
M20	33X33	numbers	5333	5360	5382	5391	5397	5399	5401	5401	5401	5401	5401	5401	5401	5401	5401		
		percent	98.74%	99.24%	99.65%	99.81%	99.93%	99.96%	100.00%	100.00%	100.00%	100.00%	100.00%	100.00%	100.00%	100.00%	100.00%		
M24	40X40	numbers	5401	5401	5401	5401	5401	5401	5401	5401	5401	5401	5401	5401	5401	5401	5401		
		percent	100.00%	100.00%	100.00%	100.00%	100.00%	100.00%	100.00%	100.00%	100.00%	100.00%	100.00%	100.00%	100.00%	100.00%	100.00%		
M28	47X47	numbers	5383	5393	5398	5399	5401	5401	5401	5401	5401	5401	5401	5401	5401	5401	5401		
		percent	99.67%	99.85%	99.94%	99.96%	100.00%	100.00%	100.00%	100.00%	100.00%	100.00%	100.00%	100.00%	100.00%	100.00%	100.00%		
K20	34X33	numbers	3164	3458	3724	3954	4154	4347	4523	4666	4789	4914	5026						
		percent	58.58%	64.03%	68.95%	73.21%	76.91%	80.49%	83.74%	86.39%	88.67%	90.98%	93.06%						
K24	40X40	numbers	3192	3472	3722	3967	4179	4375	4550	4683	4825	4955	5040						
		percent	59.10%	64.28%	68.91%	73.45%	77.37%	81.00%	84.24%	86.71%	89.34%	91.37%	93.32%						
K28	48X47	numbers	3512	3801	4057	4297	4499	4693	4868	4998	5123	5210	5285						
		percent	65.02%	70.38%	75.12%	79.56%	83.30%	86.89%	90.13%	92.54%	94.85%	96.46%	97.85%						

Table 3. The misclassification characters of font K28 in Table 1.



Since character strokes of the Kai font are more curved and shorter than those of the Ming font, it causes a lower accuracy rate. On the other hand, failure cases illustrated in Table 3 represent the characters of more complex structures. The projection histograms of these characters cannot efficiently represent the stroke structures.

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 use of the templates of a specified font. In the classification phase, 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 tested to evaluate the validity of our proposed approach. The recognition accuracy is larger than 93%.

Although the font type and size of characters can be different between the reference database and test characters, there still exist some cases in which our algorithm fails. We plan to use meshing technology or more detailed histograms to improve the classification accuracy in the future. Another popular issue in character recognition is rotation invariance. Using the projection histogram and edit distance algorithm cannot resolve the rotation problem. We plan to apply an analysis of the p-skeleton structure and design a new classification approach to overcome the rotation problem 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, 1996, pp. 641-662.
2. S. Mori, C. Y. Suen, and K. Yamamoto, "Historical review of OCR research and development," in *Proceedings of IEEE*, Vol. 80, 1992, pp. 1029-1058.
3. G. Nagy, "At the frontiers of OCR," in *Proceeding of IEEE*, Vol. 80, 1992, pp. 1093-1100.
4. C. C. Han, K. C. Fan, and Y. L. Tseng, "Coarse classification of Chinese characters via stroke clustering method," *Pattern Recognition Letters*, Vol. 16, 1995, pp. 1079-1089.
5. R. A. Wanger and M. J. Fischer, "The string-to-string correction problem," *Journal of the ACM*, Vol. 21, 1974, pp. 168-173.
6. E. S. Ristad and P. N. Yianilos, "Learning string-edit distance," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 20, 1998, pp. 522-532.
7. K. C. Fan, D. F. Chen, and M. G. Wen, "Skeletonization of binary images with non-uniform width using block decomposition and contour vector matching method," *Pattern Recognition*, Vol. 31, 1998, pp. 823-838.
8. T. Pavlidis, "A thinning algorithm for discrete binary images," *Computer Graphics and Image Processing*, Vol. 13, 1980, pp. 142-157.
9. Y. S. Chen and W. H. Hsu, "A modified fast parallel algorithm for thinning digital patterns," *Pattern Recognition Letter*, Vol. 7, 1987, pp. 99-106.

10. B. K. Jeng and R. T. Chin, "One-pass parallel thinning: analysis, properties, and quantitative evaluation," *IEEE Transactions on Pattern Analysis Machine Intelligence*, Vol. 14, 1992, pp. 1120-1140.
11. C. S. Lin, C. F. Wang, and J. S. Huang, "A new Chinese character thinning algorithm based on tracing the boundary," Technical Report of TR-88-08, Institute of Information Science, Academia Sinica, 1988.
12. C. D. Stefano, A. D. Cioppa, and A. Marcelli, "Character preclassification based on genetic programming," *Pattern Recognition Letters*, Vol. 23, 2002, pp. 1439-1448.
13. L. Heuttl, T. Paquet, J. V. Moreau, Y. Lecourtier, and C. Olivier, "A structural/statistical feature based vector for handwritten character recognition," *Pattern Recognition Letters*, Vol. 19, 1998, pp. 629-641.
14. P. Foggia, C. Sansone, F. Tortorella, and M. Vento, "Combining statistical and structural approaches for handwritten character description," *Image and Vision Computing*, Vol. 17, 1999, pp. 701-711.



Ming-Gang Wen (溫敏淦) was born in Hsinchu, Taiwan, on 1 February 1965. He received his B.S. degree in Computer Science and Information Engineering from National Chiao Tung University, Taiwan, in 1989. He started his graduate studies in Computer Science and Electronic Engineering at National Central University in 1991 and received the M.S. degree in 1993. In 1993, he joined the Department of Information Management at National Lien-Ho Institutes of Technology as a lecture. He received his Ph.D. degree in the Institute of Computer Science and Information Engineering from National Central University, Taiwan in 2003. Currently, he is an associate professor and chairman of the Department of Information Management at National United University. His research interests include optical character recognition, image processing, and digital watermarking.



Kuo-Chin Fan (范國清) was born in Hsinchu, Taiwan, on 21 June 1959. He received his B.S. degree in Electrical Engineering from National Tsing Hua University, Taiwan, in 1981. In 1983, he worked for the Electronic Research and Service Organization (ERSO), Taiwan, as a Computer Engineer. He started his graduate studies in Electrical Engineering at the University of Florida in 1984 and received the M.S. and Ph.D. degrees in 1985 and 1989, respectively. From 1984 to 1989 he was a Research Assistant in the Center for Information Research at University of Florida. In 1989, he joined the Institute of Computer Science and Information Engineering at National Central University where he became professor in 1994. He was chairman of the department during 1994-1997. Currently, he is the director of Software Research Center and Computer Center at National Central University. Professor Fan is a member of IEEE and a member of SPIE. His current research interests include image analysis, optical character recognition, and document analysis.



Chin-Chuan Han (韓欽銓) received the B.S. degree in Computer Engineering from National Chiao Tung University in 1989, and an M.S. and a Ph.D. degree in Computer Science and Electronic Engineering from National Central University in 1991 and 1994, respectively. From 1995 to 1998, he was a Postdoctoral Fellow in the Institute of Information Science, Academia Sinica, Taipei, Taiwan. He was an Assistant Research Fellow in the Applied Research Lab., Telecommunication Laboratories, Chung-hwa Telecom Co. in 1999. He is currently an Associate Professor in the Department of Computer Science and Information Engineering, National United University. His research interests are in the areas of face recognition, biometrics verification, 2D image analysis, computer vision, and pattern recognition.