ELSEVIER

Pattern Recognition Letters 16 (1995) 1023–1032

Pattern Recognition Letters

Preclassification of handwritten Chinese characters based on basic stroke substructures

Rei-Heng Cheng, Chi-Wei Lee, Zen Chen *

Institute of Computer Science and Information Engineering, National Chiao Tung University, Hsinchu, Taiwan 30050

Received 1 November 1994; revised 2 March 1995

Abstract

A method for preclassification of handwritten Chinese characters is presented. A set of basic stroke substructures is defined using the consistent stroke connection relations. A knowledge guided recognition process is employed to identify the types of the extracted basic stroke substructures found in a handwritten character. Then a 1-D character coding scheme is given to represent the character and the code can be also used for character preclassification.

Keywords: Preclassification; Handwritten Chinese character; Basic stroke substructure; 1-D string coding

1. Introduction

Handwritten Chinese character recognition is a difficult task. There are two major problems: (1) the large character set and (2) the handwriting variation. For the first problem, one may take advantage of the fact that a complex Chinese character is generally composed of subcharacters. A careful use of a set of subcharacters can lead to the partition of the entire character set into classes that contain only a small number of characters each. This process is often referred to as preclassification (or, coarse classification) of the characters (Lin and Fan, 1994; Cheng and Wang, 1993; Jeng et al., 1987). A new preclassification method will be proposed to take the following major issues of the preclassification problem into consideration.

(a) Ease in the extraction of subcharacters from a handwritten Chinese character. It is quite well known that a set of subcharacters, often called radicals, is used in an ordinary Chinese dictionary. However, these radicals are considered too complicated for the machine to extract from a character, especially when a radical contains disconnected subparts. On the other hand, any stringent constraints imposed on handwriting, such as strictly preserving the T-type connection in the character will hamper the writing freedom and will slow down the writing speed. We shall define a set of simple but reliable subcharacters that are connected and easy to extract.

(b) Recognition rate and speed of the extracted subcharacters. After a subcharacter is extracted, it must be identified. The stroke structures of the subcharacters become simpler, but the handwriting variation is still a problem. Recognition rate and speed are two major concerns. We shall use a fast knowledge guided approach for recognizing a predefined

* Corresponding author. Email: zchen@csie.nctu.edu.tw

SSDI 0167-8655(95)00031-3

set of subcharacters with a high recognition rate.

(c) Performance measure of the preclassification result. The number of preclassification classes (or clusters) should better be large and the expected class size better be small. These two factors are the measure of performance of the preclassification method. Generally, statistical methods based on the peripheral feature (Umeda, 1982; Maeda et al., 1982), background feature (Oka, 1982), complexity measure (Zhang et al., 1990), stroke center (Jeng et al., 1987), or structural feature vector (Gan and Lua, 1992) will preclassify the set of some thousands of commonly used Chinese character into at most a few hundreds of clusters. The number of clusters obtained by these results is considered insufficient. Our method will produce a very good preclassification result in terms of the above two factors.

(d) Usefulness of the preclassification features in the ensuing final (or detailed) classification (Li and Zhao, 1986). If the paradigms for the preclassification and the final classifications are quite different, it takes more effort and time to accomplish the whole job. The proposed preclassification method can be extended easily to the final classification (Cheng et al., 1994).

For the second problem, i.e., the influence of handwriting variation on the preclassification, the statistical recognition method is generally not effective to deal with the handwriting variation (Lu et al., 1991). A structural method based on the stroke information is considered better. Lu et al. (1991) proposed a method to decompose a character into branches and used spatial relations of character branches to classify an inputted unknown character. The method could tolerate some variations in the stroke slope and stroke connection relations. However, when the character contains the scattering single-segment strokes, the presumed T-connection relations or parallel relations of these strokes are not quite consistent at all. This may result in misclassification. On the other hand, Cheng and Wang (1993) used only the peripheral shape information to avoid the effect of the variations of the inner strokes and they achieved good preclassification results. Nevertheless, when there are scattering single-segment strokes in the peripheral area, they used the T-connection relation, stroke slope and stroke length of these strokes to derive the peripheral shape information. Obviously, these data are not very reliable and may cause misclassification.

Our method is primarily based on the character structural information. We intend to handle the effect of handwriting variations. We shall make the following assumptions which we think are fair in the ordinary handwriting.

(a) The different strokes, including linear and curved strokes of a character, must be written as separate strokes.

(b) The existence/nonexistence of an intersection relation between two strokes must be followed.

On the other hand, we allow the following writing freedom.

(a) The existence/nonexistence of a T-type connection between two strokes is not necessarily followed.

(b) The length, slope and curve shape of a stroke can vary to a reasonable degree.

(c) The stroke writing sequence can be changed.

(d) The small hook at the tip of certain strokes may or may not be present.

In the experiments reported, the learning phase of our method finds 4112 reference classes for a given set of 5401 commonly used Chinese characters, and the expected class size is 2.13 characters. In the testing phase of our method, we achieve a high preclassification rate of 98.85% for a test set of 5940 character samples.

The remainder of this paper is organized as follows. Section 2 defines the basic stroke substructures used in our preclassification method. The extraction and recognition of basic stroke substructures are also given. The detail of the proposed 1-D character coding and the preclassification method are described in Section 3. Section 4 includes the experimental results and discussions. Section 5 is the conclusions.

2. Extraction and recognition of basic stroke substructures

2.1. Definition of a basic stroke substructures

Now we are about to define the basic stroke substructures and show how to extract and recognize them. First of all, a character is treated as a set of

strokes and each stroke is represented by single or multiple linear line segments. Any two (linear) line segments can have one of the four possible relations:

(1) being intersected (two meet at a middle point),

(2) being connected end to end (including ㄴ, ㄱ, ㄏ, and ㄴ),

(3) being T-type connected (one segment's end point meets another segment's middle point),

(4) none of the above relations.

Among them, the connection types of ㄏ, ㄴ, and T-type are not consistent relations, because the two segments are normally written in two pen movements rather than one. In the case of a hook at the tip of a stroke, such as the connection in the form of ㄱ, the hook will be absorbed and can be removed during the stroke extraction process.

A basic stroke substructure is defined to be a set of connected line segments such that they are related pairwise by (1) being intersected, (2) being ㄴ-type or ㄱ-type connected (hereafter, these two are called the L-type in short). To extract a basic stroke substructure in a handwritten character, we check the connection type between the two connected line segments, and select those line segments that are intersected or L-type connected.

Next, given the above definition of basic stroke substructures, we can collect all the possible stroke substructures that exist in the 5401 character set by scanning through the entire set to look for all possible basic stroke substructures. In this way, we can obtain a set of 64 basic stroke substructures beforehand (see Table 1) and then construct a knowledge base to describe the stroke organization for all the basic stroke substructures. In order to identify the basic stroke substructures in a character, the strokes must be extracted first. The stroke extraction process is given below.

Table 1
The code table of the 64 basic stroke substructures

1		2		3		4		5		6	
7		8		9		10		11		12	
13		14		15		16		17		18	
19		20		21		22		23		24	
25		26		27		28		29		30	
31		32		33		34		35		36	
37		38		39		40		41		42	
43		44		45		46		47		48	
49		50		51		52		53		54	
55		56		57		58		59		60	
61		62		63		64					

2.2. Stroke extraction

For an off-line input character, we can get the stroke information by using some preprocessing techniques such as thinning (Chu and Suen, 1986; Chen and Hsu, 1989), and stroke segmentation (Lu et al., 1991; Ogawa and Taniguchi, 1982).

To simplify the problem, we use the on-line input stroke data in the current implementation of our method. The on-line input stroke data may not be 8-connected, an interpolation method is applied first to make the strokes 8-connected. Then we use the following three steps to extract each stroke information.

(i) A line fitting to the pixel points of each input stroke written in a pen movement (see Fig. 1).

Table 2
Stroke extraction results of possible curved strokes

# of line direction changes	The first turning direction	Stroke samples	Stroke type	stroke extraction results
1	Turn right			
1	Turn left			
2	Turn right			
2	Turn left			
3	Turn right			
4	Turn right			

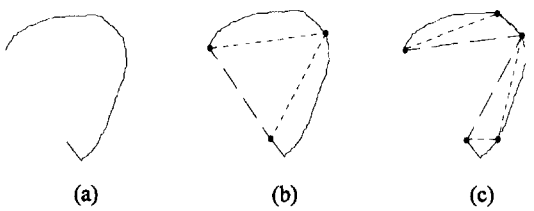

Fig. 1. Line fitting: (a) original stroke, (b) the first fitting result, and (c) the consecutive fitting results.

(ii) A count of the line direction changes by tracing the fitted line segments of the input stroke and classify the stroke into one of the pre-specified types. The possible stroke types are shown in Table 2. Take the curved strokes shown in Fig. 1 as an example; the number of extracted line segments may be different, but the segment sequence always makes a right turn, so the count of line direction changes is one.
Thus the stroke is classified as ⌝ and the stroke is refitted with two strokes of the ⌝ standard stroke.

(iii) An intersection checking on the pixels of the stroke to see if any intersection with other strokes exists. The check is done by examining each pixel of the input stroke.

2.3. Recognition of basic stroke substructure

After the process of stroke extraction described above, a stroke substructure consisting of stroke segments, that are connected through one of the three relations: (i) ⌝-connectedness, (ii) ⌞-connectedness, and (iii) intersection, can be obtained. A knowledge-guided recognition process developed in (Cheng et al., 1994) can be used to recognize the 64 basic stroke substructures. The procedure of this method is outlined below.

Step 1. In the given basic stroke substructure, find the stroke which has the maximum number of intersecting strokes and denote it as the salient stroke. If there is a tie between two strokes with the same maximum number of intersecting strokes, we select the one which contains the maximum number of ⌝-shaped or ⌞-shaped intersecting strokes; if a tie again, choose the one which contains an intersecting stroke with the highest priority according to a stroke ordering list; and, finally, select the one according to the ordering of the stroke possible positions, if there is still a tie.

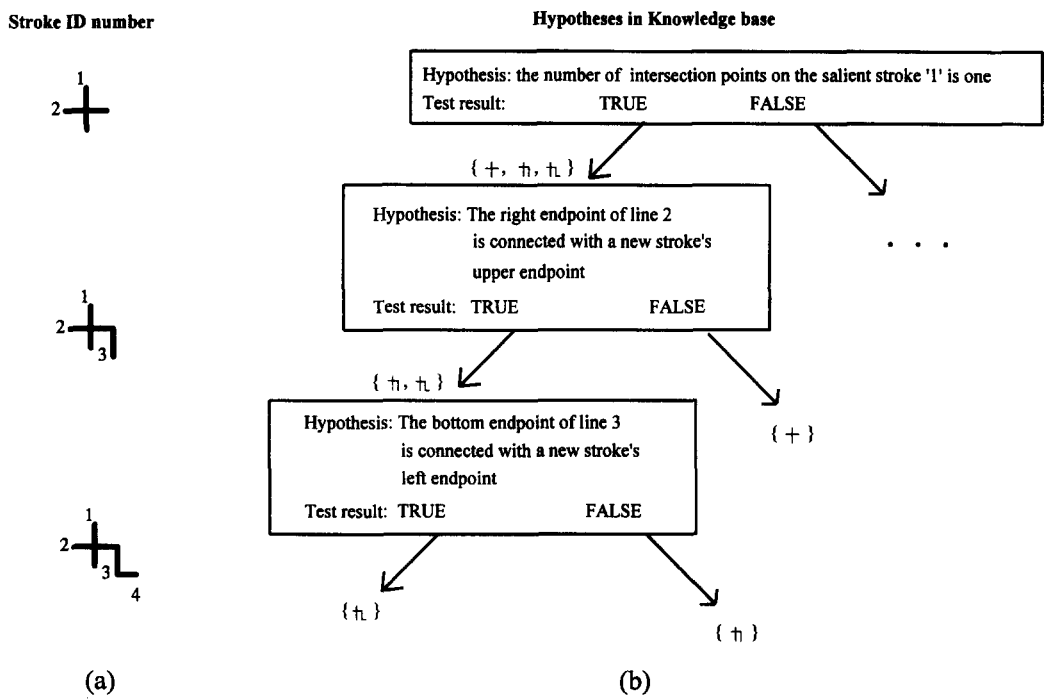

Fig. 2. The hypothesis and test steps for recognizing stroke substructures +, ㄇ and ㄩ organized in a decision tree format.

Step 2. Based on the precompiled knowledge base of the stroke organization of the 64 basic stroke substructures, a hypothesis about the existence of a second stroke and its location is proposed (see Fig. 2), and the hypothesis is actually verified. If the second stroke is correctly identified, then check if a unique identification of the given basic stroke substructure is possible. If yes, terminate with success, if not, a further hypothesis-and-test process is repeated; if the hypothesis fails, then an alternative hypothesis, if available, will be suggested by the knowledge base and this hypothesis will be verified. The recognition process either terminates with a success or stops with failure when all relevant hypotheses about the basic stroke substructure are examined.

3. Character coding and preclassification

The basic stroke substructures in a character can be used as the keys to identify the character. The strokes without any L-type or intersection relations are treated as a special group; they contain only a single segment. The preclassification method proposed in this paper is based on basic stroke substructures extracted from an input character and the spatial relations between the extracted stroke substructures.

3.1. 1-D string representation of a character for preclassification

If we use a graph to represent all the stroke substructures of a character (Lu et al., 1991), it would be too complex and not very reliable as far as the character preclassification is concerned. Instead, we use only part of stroke substructures to represent a character. In this way we can (i) use fewer spatial relations, (ii) ignore spatial relation distortions in the unused stroke substructures caused by writing variation, and (iii) use less storage space and spend less classification time.

To get the above benefits, we manually sort the 64 stroke substructures according to their structure complexity. We assign a larger numeric code to a stroke substructure that contains more strokes and has also a stronger spatial relationship. After sorting, we use only the first n (in our experiment, $n = 4$) stroke substructures for character representation.

Next, should we consider all the possible spatial relationships between the selected stroke substructures or only part of them? We find that the classification powers are almost the same irrespective of whether we use all or part of the spatial relations. So, we use part of the spatial relations for the benefit of less storage and lower computation time.

3.1.1. 1-D string coding scheme

Based on the analysis mentioned above, we present a 1-D string coding scheme for each Chinese character as follows. Assume a given character contains n basic stroke substructures. Then the 1-D string code of the character is given by

(a) $S_{\#}R_0S_1R_1S_2R_2S_3\cdots S_{n-1}R_{n-1}S_n$, if $n \geqslant 1$,

(b) $S_{\#}$, if $n = 0$.

Here $S_1, S_2, \ldots, S_n$ are the type names of the n basic stroke substructures that are identified from the

Table 3
Definition of the spatial relations used in the 1-D character coding

Spatial relation	String format	Code	Definition
R_0	R_0S_1	1	No stroke located at the top or to the right of substructure S_1
		2	There are some strokes located at the top of substructure S_1 and no stroke to the right of substructure S_1
		3	There are some strokes located to the right of substructure S_1 and no stroke at the top of substructure S_1
		4	There are both some strokes located at the top and to the right of substructure S_1
R_k	$S_kR_kS_{k+1}$	1	Substructure S_{k+1} is at the top of substructure S_k
		2	Substructure S_{k+1} is at the bottom of substructure S_k
		3	Substructure S_{k+1} is to the left of substructure S_k
		4	Substructure S_{k+1} is to the right of substructure S_k

character strokes. The sequence of $S_1, S_2, S_3, \ldots, S_n$ is obtained by sorting the basic stroke substructures according to the above-mentioned structure complexity measure. When two identical stroke substructures are found, they are ordered based on their relative position; the one at the top or right is arranged first. If the number of substructures n exceeds an upper bound, say, 4, then those $n-4$ substructures of less significance can be discarded. Next, $R_1, R_2, \ldots, R_{n-1}$ are the spatial relations between every two adjacent substructures; R_0 is the spatial location of the first substructure in the whole character. There are totally only four kinds of the spatial location R_0 and 4 kinds of the spatial relations between any two adjacent substructures, as defined in Table 3.

Finally, $S_\#$ stands for the number of the remaining strokes after the extraction of all basic stroke substructures from a character. Note that for some Chinese characters, there is no basic stroke substructure at all; instead, there are only disconnected strokes whose number is $S_\#$. In these cases, the string code is $S_\#$. In the other cases, after the extraction of all basic stroke substructures, a character may have some remaining strokes which are separate and have a single line segment. The number of these remaining single-segment strokes is given by $S_\#$ and this count is very reliable. The $S_\#$ information is useful to discriminate two characters when their substructure codes in terms of

$$R_0 S_1 R_1 S_2 R_2 S_3 \cdots S_{n-1} R_{n-1} S_n$$

are the same.

To take 園 as an example, it contains three basic stroke substructures $S_1 =$ ┼ (code = 3), $S_2 =$ ┐ (code = 2) and $S_3 =$ ┐ (code = 2). The number of remaining strokes is 9. The 1-D string code for 園 is therefore 9 R_0 3 R_1 2 R_2 2. Here, according to Table 3, the location of ┼ is 4, i.e., $R_0 = 4$; and S_2 (┐) is above S_1 (┼) and to the right of S_1, the "top-bottom" relation precedes the "left-right" relation, so $R_1 = 1$; S_3 is at the bottom of S_2, so $R_2 = 2$. The final 1-D string for 園 is 9 4 3 1 2 2 2.

3.2. Design consideration of the 1-D string code

$S_\#, S_1, S_2, \ldots, S_n$ in the 1-D string code are rather consistent under the writing variations. But the spatial information of $R_0, R_1, \ldots, R_{n-1}$ may not be all consistent. Generally speaking, the types of spatial relationships between two stroke substructures can be top-bottom, left-right and diagonal. The diagonal relation may confuse with the top-bottom and left-right relations, while the top-bottom and left-right relations are generally reliable. So, we do not use the diagonal relation in our 1-D string code. The problem now is how to deal with the diagonal relations in order to maintain the consistency of the 1-D preclassification code. We consider the problem separately in the two phases of the preclassification method: the learning phase and the testing phase. The 1-D string code is also referred to as the 1-D preclassification code when it is used for character preclassification.

The learning phase. For this phase, there are two possible ways to handle the diagonal relation. The first one is to create two versions of the 1-D preclassification code for the character to be stored in the reference data base: one is to replace the diagonal relation by the top-down relation and the other by the left-right relation. However, it is not only difficult to foresee all possible diagonal relations in each character, but it also causes the knowledge base to become too large to access efficiently.

The second approach is to replace each diagonal relation by the top-bottom relation or the left-right relation (in our experiment we use the top-bottom relation). So, each input character is coded by only one 1-D preclassification code. (See Table 4.)

The testing phase. Because there is only one 1-D preclassification code for each character stored in the database, we will generate all possible preclassification codes during the testing phase so that the "correct" one would not be missed. Moreover, the generation order should be properly designed such that the

Table 4
The reference classes obtained in the learning phase for the 5401 character set

Class size	No. of classes of given size	Class size	No. of classes of given size
1	3460	8	8
2	400	9	5
3	111	10	5
4	59	11	2
5	32	12	1
6	13	13	3
7	12	18	1

Table 5
The number of legal preclassification codes generated for 5940 test samples: (a) when multiple versions not included in the database, (b) when multiple versions included in database

(a) Number of legal preclassification codes generated	Number of test samples with the given codes	(b) Number of legal preclassification codes generated	Number of test samples with the given codes
0	286	0	54
1	4575	1	4651
2	897	2	1035
3	120	3	134
4	62	4	66
Total	5940	Total	5940

correct code for matching can be produced as soon as possible.

We use the generate-and-test strategy to find the correct preclassification codes. Because we have replaced all diagonal relations by the top-bottom relation in the learning phase, it is more likely to match to the right code stored in the database if we replace the diagonal relations by the top-bottom relations in the testing phase. So, the candidate preclassification codes are generated in the following order:

(1) Replace all diagonal relations by the top-bottom relations.

(2) Choose one of the diagonal relations in turn and replace it by the left-right relation and the remaining diagonal relations by the top-bottom relation.

(3) Choose two out of the diagonal relations in turn and replace them by the left-right relation and the remaining diagonal relations by the top-bottom relation, and so on. (Note there are generally only a few diagonal relations in the character.)

If a preclassification code cannot match to any code stored in the database, it would be an illegal one. Those matched to a legal code are classified to the class found. Now we shall explain that the misclassification probability caused by the accidental mismatch in our generate-and-test process is low. The reason is as follows. In the 1-D preclassification code, there is other information such as stroke substructure ID codes, top-bottom and left-right relation in addition to the diagonal relations. The chance that two legal codes differ in the positions of diagonal relations is low, in particular, when there are many basic stroke substructures. Most of the preclassification codes generated by the above generation method are illegal.

To investigate the feasibility of the above code generation process, we collect two statistics on (i) the number of possible legal class candidates for each test character sample (the smaller the better) and (ii) the order of the correct preclassification code (the one stored in the database) found in the sequence of generated legal preclassification codes (the sooner the better). Table 5 indicates that most of the test character samples generated only 1 to 2 possible preclassification codes. Table 6 shows that more than

Table 6
The preclassification results for 5940 test samples: (a) when multiple versions not included in the database, (b) when multiple versions included in the database

(a) n	The accumulative number of correctly preclassified characters using the first n generated 1-D codes	(b) n	The accumulative number of correctly preclassified characters using the first n generated 1-D codes
1	5424	1	5787
2	5512	2	5852
3	5530	3	5870
4	5532	4	5872

91% of the test character samples would be correctly classified when using only the firstly generated preclassification code (more detail in Section 4). In other word, the preclassification code generation sequence mentioned above is desired.

4. Experimental results

The implementation of the preclassification method consists of two phases: learning and testing. In the learning phase one handwritten sample for each of the 5401 Chinese character set is inputted. Then all possible basic stroke substructures embedded in the character are extracted. At the end, we collect manually all the possible basic stroke substructures and construct a knowledge base to describe the stroke organization of these substructures. Then we can use this knowledge base to design an automatic recognition process for identifying each of the substructures. Please refer to the method developed by us (Cheng et al., 1994). On the second pass of the same samples, the system identifies the types

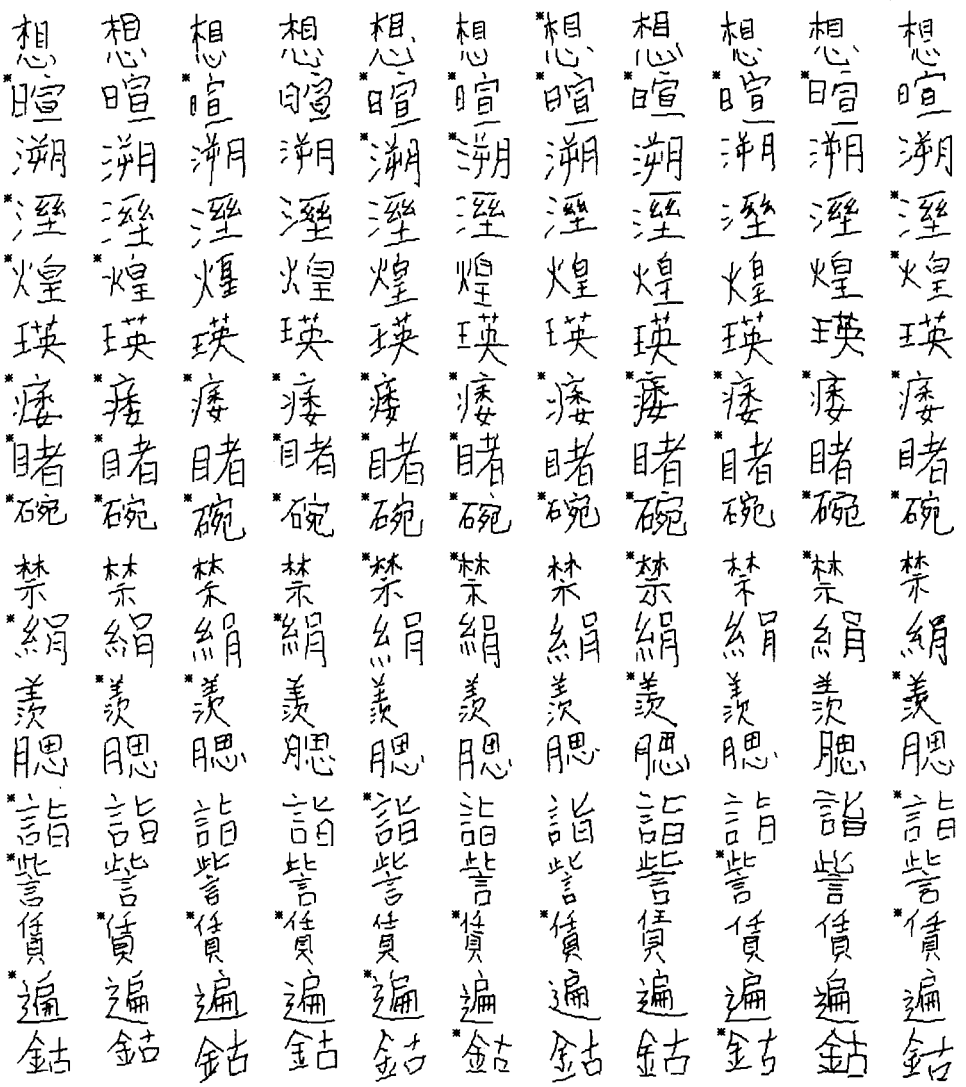

Fig. 3. Some test samples used in the testing phase of our experiments. The characters with * are those correctly classified by our method, but not by other methods using the T-connection relation.

Table 7
Comparisons of different methods according to the stroke information used

	T-connection	Stroke slope	⌜, ⌟ connection	⌝, ⌞ connection	Intersection	Global stroke information	Use of spatial relation
Cheng and Wang (1993)	Y	Y	Y	Y	Y	N	Y
Lu et al. (1991)	Y	Y	Y	Y	Y	Y	Y
Ours	N	N	N	Y	Y	Y	Y

of basic stroke substructures and sort them according to the structure complexity, then finds the spatial relations between the pairs of adjacent substructures. Finally, the 1-D numeric code of the character is constructed. The code is compared with the reference data base built so far. If a hit is found, the current character is grouped into the found class; if no hit is found, a new class is inserted into the reference data base. The reference classes of the 5401 Chinese character set obtained in the learning phase are given in Table 4. It indicates that the 64 substructures shown in Table 1 are able to group the 5401 characters into 4112 classes.

In the testing phase, 540 characters are selected uniformly from the 5401 character set, one out of every ten characters. These characters represent a typical spectrum of character stroke patterns whose complexity ranges from simple to complex. Then 11 samples of each of the 540 characters, with a subtotal of 5940 samples, are collected to be the test samples. Some test samples are shown in Fig. 3. For each test sample, the 1-D numeric character code is constructed. The character code is checked against the reference data base to see if it finds a hit in the data base. Table 6(a) illustrates the preclassification result of this experiment. There are 5424 samples which are correctly preclassified using only the firstly generated 1-D preclassification code. After we check

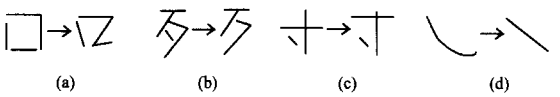

Fig. 4. Some test samples that cause the preclassification errors.

all legal class candidates for each sample, which are at most 4 candidates in our case, there are totally 5532 characters which could be finally preclassified correctly. Table 6(a) shows there are 408 test samples which cannot be correctly preclassified. These 408 test samples can be broken down to two categories: 286 of them find no match and, therefore, are rejected (see Table 5(a)) and 122 of them find an incorrect match and, thus, are misclassified (the misclassification rate is about 2%). The above experiments are implemented on an IBM PC 486-33 and the preclassification time takes only 0.064 second per character on average (the stroke extraction time is excluded).

The misclassified character samples are due to

(1) a character with two or more possible writing versions (Fig. 4(a)),

(2) some short strokes are missing (Fig. 4(b)),

(3) the intersection and/or L-type relations are missing (Fig. 4(c)), or

(4) a stroke type error occurs (Fig. 4(d)).

The problem (1) mentioned above could be solved by including multiple versions in the database. Ta-

Table 8
Classification statistics obtained by various methods

	Character set size	No. of classes obtained	Max. class size	Expected class size
Jeng et al. (1987)	5384	320	340	> 36.27
Li and Zhou (1986)	753	397	–	–
Wang (1976)	7334	2144	38	~ 7.5
Cheng and Wang (1993)	5401	2144	84	9.16
Ours	5401	4112	18	2.13

bles 5(b) and 6(b) illustrate that the accumulative correct preclassification rate using the first 4 candidates is 98.85% (5872 test samples) after using multiple versions. On the other hand, we can allow a small change in the value of $S_{\#}$ by including more preclassification codes to preclassify an unknown character in order to tolerate some unimportant missing strokes. The ambiguity problem of the intersection and L-type relations in Figs. 4(c) and 4(d) may be solved by considering all possible combinations of the ambiguous relations.

Comparisons of our method with other methods are shown in Tables 7 and 8. Table 7 indicates that more writing freedom is allowed in our method. Table 8 indicates the classification results of various methods. Generally speaking, our method is better.

5. Conclusions

We have presented a preclassification method that produces a satisfactory preclassification result. The proposed method uses the consistent features in handwriting to define the reliable stroke substructures instead of using inconsistent features such as the T-type connection and stroke length, etc. In our method, only when determining certain types of stroke substructures, we used the quantized stroke slope. Therefore, our method allows mild handwriting variations in the stroke sequence, stroke slope and length, curve shape of the stroke, and T-type connection between two strokes. The experimental results showed that the correct preclassification rates obtained were rather high. In the future we shall improve our method by solving the ambiguity problem of the stroke intersection and L-type relations encountered in the applications. Also, we shall consider the automatic construction of the knowledge bases used in the recognition method.

References

Chen, Y.S. and W.H. Hsu (1989). A systematic approach for designing 2-subcycle and pseudo 1-subcycle parallel thinning algorithm. *Pattern Recognition* 22 (3), 267–282.

Cheng, H.D. and J.F. Wang (1993). Preclassification for handwritten Chinese character recognition by a peripheral shape coding method. *Pattern Recognition* 26 (5), 711–719.

Cheng, R.H., C.W. Lee and Z. Chen (1994). Recognition of radicals in handwritten Chinese characters by means of problem reduction and knowledge guidance. Submitted to *Internat. J. Pattern Recognition Artificial Intelligence*.

Chu, Y.K. and C.Y. Suen (1986). An alternate smoothing and stripping algorithm for thinning digital binary patterns. *Signal Processing* 11, 207–222.

Gan, K.W. and K.T. Lua (1992). Chinese character classification using an adaptive resonance network. *Pattern Recognition* 25 (8), 877–882.

Hilderbrand, T.H. and W. Liu (1993). Optical recognition of handwritten Chinese characters: advances since 1980. *Pattern Recognition* 26 (2), 205–225.

Jeng, B.S., E. Hsieh, G.H. Chang and C.H. Huang (1987). Classify Chinese character by stroke and center methods. *Proc. 1987 Telecommunication Symp.*, 545–547.

Knoll, A.K. (1969). Experiments with characteristic loci for recognition of handprinted characters. *IEEE Trans. Comput.* 18, 366–373.

Li, B. and S. Zhao (1986). A new approach to recognition of both handwritten and multi-font printed Chinese characters. *Proc. Internat. Conf. on Pattern Recognition*, 641–643.

Lin, T.Z. and K.C. Fan (1994). Coarse classification of on-line Chinese characters via structure feature-based method. *Pattern Recognition* 27 (10), 1365–1377.

Lu, S.W., Y. Ren and C.Y. Suen (1991). Hierarchical attributed graph representation and recognition of handwritten Chinese characters. *Pattern Recognition* 24 (7), 617–632.

Maeda, K., Y. Kurosawa, H. Asada, K. Sakai and S. Watanabe (1982). Handprinted Kanji recognition by pattern matching method. *Proc. Internat. Conf. on Pattern Recognition*, 789–792.

Nouboud, F. and R. Plamondon (1991). A structural approach to on-line character recognition: system design and applications. *Internat. J. Pattern Recognition Artificial Intelligence* 5 (1 & 2), 311–334.

Ogawa, H. and K. Taniguchi (1982), Thinning and stroke segmentation for handwritten Chinese character recognition. *Pattern Recognition* 15 (4), 299–308.

Oka, R. (1982). Handwritten Chinese–Japanese characters recognition by using cellular feature. *Proc. Internat. Conf. on Pattern Recognition*, 783–785.

Umeda, M. (1982). Recognition of multi-font printed Chinese character. *Proc. Internat. Conf. on Pattern Recognition*, 763–769.

Wang, Y.W. (1976). *Four-Corner Method.* Commercial Press, Taipei, Taiwan.

Zhang, S., B. Taconet and A. Faure (1990). A complexity measure based algorithm for multifont Chinese character recognition. *Proc. Internat. Conf. on Pattern Recognition*, 573–577.