RECOGNITION OF CHINESE CHARACTERS BY GROWING STROKE WINDOW

WEI-HSIEN WU[‡], KUO-CHIN FAN[†]
[†]Institute of Computer Science and Information Engineering,
National Central University, Chungli, Taiwan, 32054, R.O.C.
kcfan@ncuee.ncu.edu.tw

[‡]Department of Information Management, Kuang Wu Institute of Technology,
Pei-Tou, Taipei, Taiwan, 112, R.O.C.
wswu@ms8.pchome.com.tw

ABSTRACT

In this paper, we propose a novel Chinese character recognition method. In our approach, the 2-D input character is first converted to a 1-D stroke string. After stroke extraction, the stroke sequence is rearranged in which strokes with similar orientation are located consecutively. By special organizing, strokes that fail to be extracted correctly will be located adjacently in the rearranging stroke string. Some intra-character spatial relationships are recorded in an adjacency matrix. In the recognition stage, the adjacency matrix of input character and that of character in the database are compared. Since both the two adjacency matrices are built by its own character, the relative relationships between strokes will not be influenced by rotation and translation distortions. Finally, the similarity between input character and reference character is assessed by growing stroke window. Experimental results show that the proposed method is feasible. Besides, it can also resolve the problems inherited from the stroke extraction stage.

Keywords: Character recognition, Stroke extraction , Stroke string, Stroke window

1.INTRODUCTION

Chinese character recognition is admitted as a very difficult problem in character recognition due to (1) very large character set, (2) high complexity of Chinese characters, and (3) many similar character patterns. Many authors have taken the challenge and proposed various algorithms trying to get satisfactory result. According to the features used in the recognition of Chinese characters, there are two kinds of recognition mechanisms: one is based on statistic features[1],[2],[3], another is based on structure features[4],[5],[6],[7],[8].

Structure features include line segments, strokes, and fork points along with spatial relationships between them. For those recognition schemes that use structure features, some preprocessing processes are first applied to obtain the skeleton of character, and then extract the needed features. The result of feature extraction is the most important factor that influences the result of recognition. However, fork point is not a stable feature due to bad input image or the distortion generated by previous feature extraction process. The consequence is that the extracted strokes (or line segments) may be incorrect. These problems include merging two touching strokes together and splitting one stroke into two strokes. The former is very difficult to be detected and recovered just by some simple rules. The latter can be improved by some algorithms, though a perfect solution is still absent. Relaxation method[5],[6],[7] and dynamic programming technique[8] are commonly used to be the kernel of recognition mechanism based on structure features.

In this paper, we propose a novel Chinese character recognition method by modeling an off-line input character as an on-line input character. First, we rearrange the stroke order by the sequence with the horizontal strokes being the first, vertical strokes followed behind, right slanted strokes being the third, and left slanted strokes being the last. The rearrangement causes the strokes that own similar orientation appearing consecutively in the sequence. The stroke order is organized so that if two strokes of an input character are formed by splitting from one stroke, then the two strokes would be located adjacently in the rearrangement stroke sequence. Similarly, if one stroke is formed by merging two strokes, then the stroke will substitute the location for that of the two merging strokes in the rearranged stroke sequence (otherwise, they would not be merged together).

The Chinese character recognition problem now is similar to the attributed string matching problem[9]. However, there are some differences between the two problems. In the recognition of Chinese character, the relationships between strokes are very important factors to assess the similarity of two characters. That is, the operations conducted on the symbols (strokes) in the string should not just focus on the symbol (stroke) itself. In our works, the recognition process is proceeded by growing a "stroke window". While a new stroke (symbol) is to be assessed, the relationships between the stroke and other strokes in the stroke window are evaluated. By assessing the similarity of strokes in the stroke window, the relationships between them are well considered.

While performing the recognition process, we define a stroke window to be a reference of the assessment of matching score. The stroke window is increased step by step until a bad matching is reached. Then, the stroke sequence is divided into two parts: one possesses good matching score and the other possesses bad matching score. The bad part proceeds the recognition process recursively. Finally, the characters in the database that own high matching score with the input character are selected to be the output result.

The rest of this paper is organized as follows. In section 2, the process for obtaining the rearranged stroke sequence is described. Section 3 introduces the recognition process. Experimental results are demonstrated in section 4 to verify the validity of our proposed approach. Finally, concluding remarks are given in section 5.

2. STROKE ORDER REARRANGEMENT

In our work, the strokes of input character are classified into four types: horizontal, vertical, 45-degree, and 135-degree strokes. Before presenting the details of the rearrangement of stroke order, we first assign one of the four stroke types to each stroke. If the orientation of a stroke is located in the interval of 340^o to 20^o, then the stroke is regarded as a horizontal stroke. If the orientation of a stroke is located in the interval of 70^o to 110^o, then the stroke is regarded as a vertical stroke. If the orientation of a stroke is located in the interval of 20^o to 70^o, then the stroke is regarded as a 45-degree stroke. If the orientation of a stroke is located in the interval of 110^o to 160^o, then the stroke is regarded as a 135-degree stroke. The strokes are reordered by the sequence of horizontal strokes being the first, vertical strokes being the second, then 45-degree strokes, and 135-degree strokes being the last.

The center point of each stroke is checked for the rearrangement of stroke order. For horizontal strokes, the sequence is rearranged from top to bottom and from left to right. While considering one stroke, those strokes that locate on a horizontal window below the object stroke are also considered to cover the variations caused by the stroke extraction process. These strokes, as well as the object stroke, are ordered from left to right. The final result will keep the spatial relationship among strokes. That is, if two strokes, say *a* and *b*, are located on the same horizontal level with (the center point of) stroke *a* which is located on the left side of (the center point of) stroke *b,* then stroke *a* is prior to stroke *b* in the rearranged stroke string. Similar processes are applied to vertical strokes except that it is reordered from left to right and from top to bottom.

For 45-degree strokes, a 45-degree window is used to cover the variations caused by the stroke extraction process. The stroke that is located on the right-up side of the window will be prior to those strokes that are located on the left-down side of the window in the rearranged stroke string. Similarly, for 135-degree strokes, a 135-degree window is used to cover the variations caused by the stroke extraction process. The stroke that is located on the left-up side of the window will be prior to those strokes that are located on the right-down side of the window in the rearranged stroke string. Shown in Figure 1 is an example illustrating the final stroke string generated by the rearrangement process. In Figure 1, horizontal strokes 1, 4, 6, 7 are first ordered from top to bottom, then vertical stroke 5 is ordered, finally, the 45-degree stroke 2 and 135-degree stroke 3 are reordered. Therefore, the final rearranged stroke string is "1467523".

As mentioned previously, the rearrangement of stroke sequence can keep the spatial relationship among strokes. Even if one stroke has been extracted as several substrokes by the distortions around fork points, the spatial relationship among these substrokes still remains unchanged. Another kind of distortion is caused by the touching of two strokes. For example, two horizontal strokes in the character "林" have been touched together before stroke extraction. It is very difficult to rectify the fault and extract the two strokes correctly. However, the extracted long stroke will substitute the position for that of the two touching strokes in the stroke string. In the recognition stage, the long stroke will be regarded as forming by the merging of two strokes comparing with the database character "林".

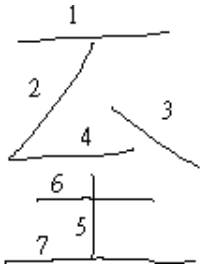

Figure 1. The rearranged 1-D stroke string is "1467523".

3. RECOGNITION

After the strokes embedded in a character have been extracted, an adjacency matrix *A* corresponding to each character in the database is built. Each element, $A_n(i,j)$, $n=1,2,3$, in the adjacency matrix represents the relationship between stroke *i* and stroke *j*, where $A_1(i,j)$ represents the distance relationship between the center points of strokes *i* and *j*, $A_2(i,j) = \cos^{-1}(i.j/|i||j|)$ represents the angle relationship between strokes *i* and *j*, and $A_3(i,j) = ||i|-|j||/MAX(|i|,|j|)$ represents the ratio of length relationship between strokes *i* and *j*.

While recognizing an input character, an adjacency matrix *B* corresponding to the character is also built. Assume that the input character possesses *n* strokes, then all characters in the database that own *n-4* to *n+4* strokes are chosen to be the candidates. For each candidate, a matching score is calculated. If the matching score is larger than a threshold, then the candidate is put into a matching list. After all candidates have been checked, the matching list is sorted descendingly. Finally, output the matching list as the recognition result.

3.1 The Matching Process

Before discussing the details of the matching algorithm, we first define "Score of Stroke Window" (SOW). Let the rearranged stroke string of an input character be $IS_1,IS_2,\ldots,IS_i,\ldots,IS_j,\ldots IS_e$ and that of candidate character be $CS_1,CS_2,\ldots,CS_m,\ldots,CS_n,\ldots\ CS_e$. SOW(i,j,m,n) is defined as the matching score between $IS_i,\ldots,IS_j$ and $CS_m,\ldots,CS_n$(where $j - i = n - m$). While the size of Stroke Window is equal to 1(i.e. $i = j$ *and* $m = n$), SOW(i,i,m,m) is the matching score between stroke *i* of the input character and stroke *m* of the candidate character. In this case, SOW is defined as:

$$\mathrm{SOW}(i,i,m,m) = (e^{-\mathrm{DIST}(i,m)/\mathrm{thd1}} \times e^{-\mathrm{AGL}(i,m)/\mathrm{thd2}})^{1/2},$$

where DIST(i,m) is defined as the distance of the center points of stroke *i* and stroke *m*, AGL(i,m) is defined as the angle between the two strokes.

If the size of stroke window is larger than 1, SOW is formulated as:

$$\mathrm{SOW}(i,j,m,n) = (\prod_{k=i}^{j} S_k(i,j,m,n))^{(1/(j-i+1))},$$

where $S_k(i,j,m,n)$ is the total matching score of stroke *k* in the interval of strokes *i* and *j*, which is formulated as:

$$S_k(i,j,m,n) = (\prod_{p=i,p\neq k}^{j} S_{kp})^{(1/(j-i))},$$

where S_{kp} is formulated as:

$$S_{kp} = (e^{-|A_1(k,p)-B_1(l,q)|/thd1} \times e^{-|A_2(k,p)-B_2(l,q)|/thd2} \times e^{-|A_3(k,p)-B3(l,q)|/thd3})^{(1/3)},$$

where thd1, thd2, and thd3 are the thresholds, whereas $l = m+(k-i)$, and $q = m+(p-i)$. That is, the relative location between strokes *i* and *k* is equal to that between strokes *m* and *l*, and the relative location between strokes *i* and *p* is equal to that between strokes *m* and *q*. Note that S_{kp} is the measurement of the difference of three relationships between input strokes *k* and *p* and that between candidate strokes *l* and *q*. Figure 2 illustrates the calculation of S_k with the stroke window size equaling 3. Assume that the adjacency matrices *A* and *B* that hold the three spatial relationships are available. Then, calculate $S_{k(k+1)}$ and $S_{k(k+2)}$, respectively. Finally, we can obtain S_k which is the product of $S_{k(k+1)}$ and $S_{k(k+2)}$. Note that *A(k,p)* and *B(l,q)* hold relative relationships between the strokes of input character and candidate character, respectively; and S_{kp} is the difference measurement between the input character and candidate character. Therefore, the matching methodology possesses high tolerance to translation variation and, in some range, rotation variation. The detail of the matching methodology will be presented in the following contexts.

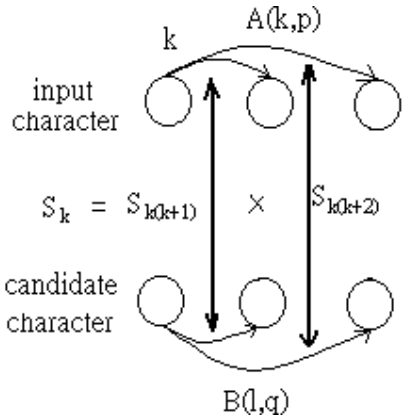

Figure 2. Illustration of the calculation of S_k.

Assume that the candidate character possesses α (or β) strokes, and the input character possesses β (or α) strokes, $\alpha \geq \beta$. Beginning with the first stroke in the input character, the matching process increases the size of stroke window step by step until the last added stroke, say stroke *k*, possesses a bad matching score with other strokes in the stroke window. That is, $S_k(1,k,1,k)$ is less than a threshold. If the size of stroke window continuously increases to β, it means that there are so many similarities between the input character and the candidate character. The candidate is put into the matching list. On the other hand, if the matching process is halted by a bad matching stroke *k*, then the input stroke *k* is regarded as the beginning of bad matching. Basically, we consider that the candidate exactly matches with input character. Therefore, we assume that there are some problems appearing in stroke *k*. The problem may be resulted from the disorder of stroke *k* in the rearranged stroke sequence, or the distortions in the stroke extraction stage. The latter may be caused by a redundant stroke, a lost stroke, merging two strokes into one stroke, or splitting one stroke into two strokes.

3.2 Dealing With The Distortions

There are five kinds of distortions may occur in the extracted stroke. In this section we discuss the details of each distortion and the solution of the proposed recognition method.

For demonstration convenience, in Figures 3~4, the gray circles are defined to represent strokes that possess high matching score, and the black circles represent strokes that have some problems in the matching process. The white circles represent strokes that should be matched recursively. Besides, for discussion convenience, stroke *k* means the *k-th* ($0 \leq k \leq$ j-i) stroke in the stroke window.

Case 1: *The problem resulted from the disorder of a stroke*

For the problem resulted from the disorder of a stroke (say stroke *k*), we try to change the location of stroke *k* with the stroke that is posterior to stroke *k* in the stroke string, say stroke *l*. If the new score of $S_k(1,k,1,k)$ is larger than a threshold, then stroke *l* is moved and inserted prior to stroke *k*. Now we have two groups of strokes: one possesses a good matching score, and the other possesses a bad matching score. The bad group will proceed the matching process recursively. Finally, the returned matching score is the product of the matching score of good group and that of bad group.

Case 2: *The distortions caused by a redundant stroke*

For the distortions caused by a redundant stroke, the *k-th* stroke is the beginner of bad matching. Since stroke *k* is redundant, the matching process will evaluate SOW(i+k+1,j,m+k,n) (see Figure 3(a)) recursively. For this kind of distortion, the final matching score is the product of the score of good part and that of bad part.

Case 3: *The distortions caused by a lost stroke*

For the distortions caused by a lost stroke, the *k-th* stroke is the beginner of bad matching. Similarly, since stroke *k* is next to the lost stroke (if the lost stroke exists), the matching process will evaluate SOW(i+k,j,m+k+1,n) (see Figure 3(b)). The final matching score is the product of the score of good part and that of bad part.

Case 4: *The distortions caused by the merging of two strokes into one stroke*

For the distortions caused by the merging of two strokes into one stroke, the *k-th* and the *(k+1)-th* strokes of candidate character are assumed to be merged together and form stroke *k* of the input character. Therefore, the stroke string is divided into two groups: one possesses a good matching score and the other possesses a bad matching score. Since stroke *k* of the input character possesses similar orientation to that of strokes *k* and *k+1* of the candidate character (otherwise they would not be merged), and the center points of the two stroke *k* (stroke *k* of input character and stroke *k* of candidate character) are close, SOW(i,i+k,m,m+k) may be large enough to be regarded as a good match. In such case, stroke *k+1* will be the beginner of bad matching. On the other hand, SOW(i,i+k,m,m+k) may be less than the testing threshold. Thus, stroke *k* is the beginner of bad matching. Since the rearrangement of stroke order will keep the spatial relationship between strokes, the merged two strokes would be located adjacently in the stroke string. Referring to Figure 4(a), for the case that stroke *k* is the beginner of bad matching, the good matching group contains stroke *i* to stroke *i+k-1*. To calculate the matching score of bad matching group, strokes beginning with stroke *k+1* of input character are matched recursively with strokes beginning with stroke *k+2* of candidate character. Similarly, for the case that stroke *k+1* is the beginner of bad matching (now pointer *k* is located on stroke *k+1*, see Figure 4(b)), the good matching group contains stroke *i* to stroke *i+k-2*. Strokes beginning with stroke *k* of input character are matched recursively with strokes beginning with stroke *k+1* of candidate character. The larger one of the two matching score is chosen to be the output of this kind of distortion. Shown in Figure 4, the light gray circles are the generated strokes by the merging distortion.

Case 5: *The distortions caused by splitting one stroke into two strokes*

For the distortions caused by splitting one stroke into two strokes, the processes are similar to that of case 4.

The matching process will proceed recursively until all strokes are completely matched. Whenever there are problems that halt stroke window increasing, the output will be the largest one of the matching score of the five types of problems as mentioned above. However, the processes of dealing with the last two kinds of distortions are actually included in the processes of dealing with lost stroke and the redundant stroke. Consider Figure 4, the strokes that cause distortions are skipped, and a new stroke window matching process that contains the remaining strokes is proceeded thereafter. The process is actually equal to skipping one stroke in the case of redundant stroke(Figure 3(a)), and then in the following recursive steps, skip twice of the lost stroke(Figure 3(b)). Therefore, in the implementation of the matching process, only the first three cases are evaluated, and this greatly improves the recognition efficiency.

4. EXPERIMENTAL RESULTS

In this section, experimental results are demonstrated to verify the validity of the proposed approach. The algorithm is implemented using C language. For efficiency consideration, we limit the recursive depth to degree 6.

The proposed recognition method has been conducted on 1500 frequently used printed Chinese characters. The database contains the standard strokes string of each character extracted from Hei form characters sizing in 120 by 120, and the input is the stroke string extracted[10] from true type character. For the character that possesses over 15 strokes, we divide the stroke string into 4 parts (according to 4 types of strokes) to get higher efficiency and to reduce the influence of distortion. The final recognition result is the combined matching score of the four parts. If one part gets a matching score that is unsatisfactory due to too many distortions, the other three parts will dilute the bad effect. The recognition rate is 92%, and the recognition speed is about 0.55 s/c (run at an IBM® compatible PC with AMD K6-II 350 CPU).

We also test the proposed method with an article as shown in Figure 5. The input characters are scanned with gray-scale 300 dpi resolution, and preprocessed and transformed into binary image. The stroke strings of reference characters are generated[10] from the associative Hei-font characters. The input includes two images: one contains printed characters and the other contains handwritten characters. The recognition rates are about 97.2% and 93.1%, respectively.

5. CONCLUSIONS

In this paper, a novel method is proposed to recognize Chinese characters. In our approach, the stroke order of a character is rearranged. The rearrangement process will keep the spatial sequence between the same kind of strokes. Then three spatial relationships (distance between strokes, angle between strokes, and ratio of length of two strokes) are evaluated. In the three spatial relationships, the distance between the center points of two strokes is the only relationship that is sensitive to the size difference in two characters. Rotation and translation distortions would not alter the absolute value of the three spatial relationships. In our approach, the recognition process measures the similarity between two characters by growing stroke window. For the sake of efficiency, the rearranged stroke string is divided into four parts, with each part corresponding to one type of stroke. The recognition also gets the benefit from the division of stroke string due to the fact that the bad matching caused by distortion of one part would not affect the other parts.

REFERENCES

[1] G. L. Cash and M. Hatamian, “Optical Character Recognition by The Method of Moments,” Comput. Vision Graphics Image Processing, Vol. 39, 1987, pp.291-310.

[2] J. S. Huang and M. L. Chung, “Separating Similar Complex Chinese Characters by Walsh Transform,” Pattern Recognition, Vol. 20, 1987, pp. 425-528.

[3] R. D. Romero, D. S. Touretzky, and R. H. Thibadeau, “Optical Chinese Character Recognition Using Probabilistic Neural Networks,” Pattern Recognition, Vol. 30, 1997, pp. 1279-1292.

[4] F. H. Cheng, W.-H. Hsu, and M.-Y. Chen, “Recognition of Handwritten Chinese Characters by Modified Hough transform techniques,” IEEE Trans. Patt. Anal. Machine Intell., Vol. 11, 1989, pp. 429-439.

[5] L. H. Chen and J. R. Lieh, “Handwritten Character Recognition Using a 2-layer Random Graph by Relaxation matching,” Comput. Vision Graphics Image Processing, Vol. 23, 1990, pp.1189-1205.

[6] B. Wang, K. C. Fan and J. S. Huang, “Recognition of handwritten Chinese Character by modified relaxation methods,” Image and Vision Computing, Vol. 12, 1994, pp. 509-522.

[7] F. H. Cheng, W. H. Hsu and M. C. Kuo, “Recognition of Handprinted Chinese Characters Via Stroke Relaxation,” Pattern Recognition, Vol. 26,

1993, pp. 579-593.
[8] H. J. Lee and B. Chen, “Recognition of Handwritten Chinese Characters Via Short Line Segments,” Pattern Recognition, Vol. 25, 1992, pp. 543-552.
[9] W. H. Tsai and S. S. Yu, “Attributed String Matching with Merging for Shape Recognition,” IEEE Trans. PAMI, Vol. PAMI-7, No. 4, 1985, pp. 453-462.
[10] K. C. Fan and W. H. Wu, “A Run Length Coding Based Approach To Stroke Extraction Of Chinese Characters,” Pattern Recognition, Vol. 33, No. 11, 2000, pp. 1881-1895.

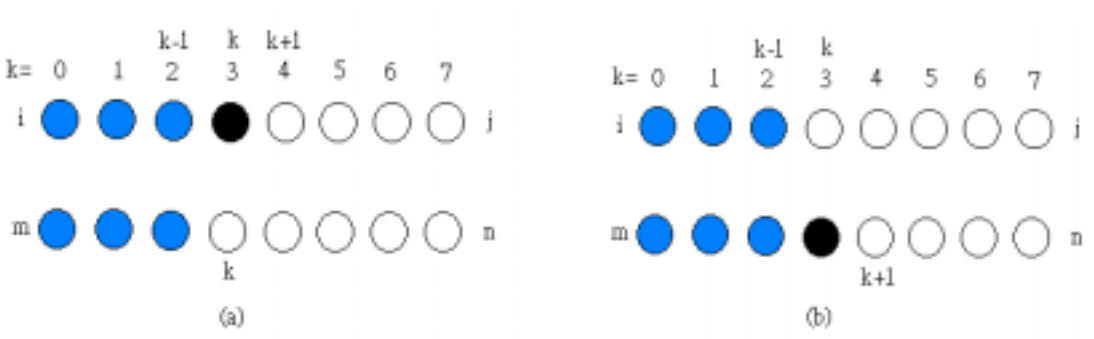

Figure 3. (a) Illustration of the matching relationship of the distortion caused by redundant stroke k where the black circle represents the redundant stroke. (b) Illustration of matching relationship of the distortion caused by a lost stroke where the black circle represents the unmatched stroke.

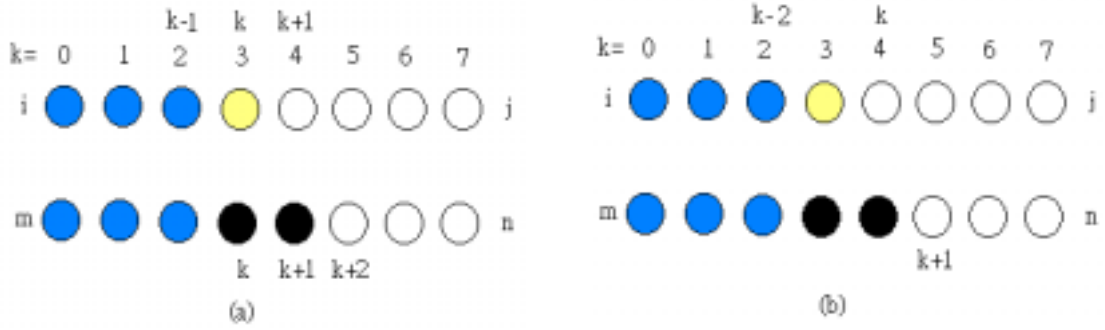

Figure 4. Illustration of two possible matching relationships of the distortion caused by merging two strokes into one stroke. The black circles represent the two strokes that are merged together.

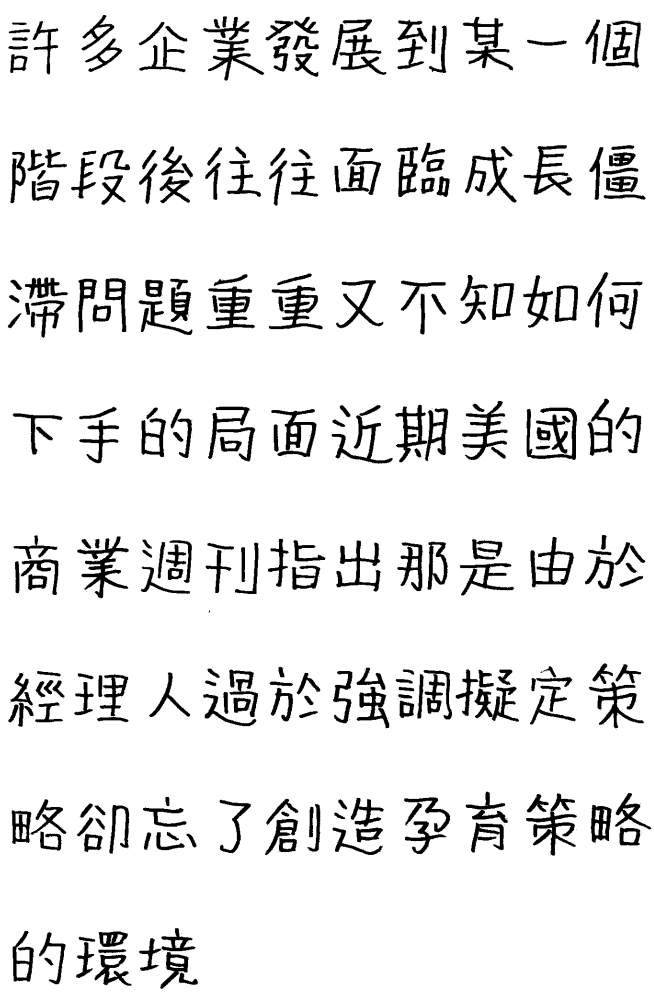

Figure 5. Handwritten testing characters (scanned image).