Comparison of Shape-based and Stroke-based Methods for Segmenting Handwritten Chinese Characters

Jian Yang, Han Zhang, Mark Dencler and Chao Lu
Department of Computer and Information Sciences
Towson University
8000 York Rd, Towson, Maryland, 21252, USA
Email: clu@towson.edu

Abstract

The segmentation of handwritten Chinese text into Chinese characters is an important preprocessing step to the offline Chinese character recognition. It is also a very difficult task due to so many Chinese characters and their handwritten structures can be very complex. Many researchers have developed various algorithms during the past decade [1-6]. In this paper, we compare two of the existing algorithms. The first one is spatial shape-based algorithm proposed in [5], which segments the character strings into radicals, not dealing with stroke identification. The second algorithm is stroke-based [2-4], which traces each stroke and draws the stroke-bounding box, then merges the boxes by a set of rules. Based on the algorithms presented in [2-5], we wrote C++ programs for time complexity, accuracy performance comparisons using different handwritten Chinese character texts. Our experimental result shows that the spatial shape-based algorithm [5] is faster and more accurate.

Keywords: *Segmentation, character recognition, computer vision, image processing.*

1. Introduction

Offline handwritten Chinese character recognition is much more difficult than machine printed character recognition. The main reasons are: a) there are about fifty thousand Chinese characters and about three thousand characters are used in daily life; b) the structures of various font styles can be complex; c) there are many analogical characters; and d) various people have different writing styles. How to extract the Chinese characters from the scanned Chinese text correctly and efficiently has been an interesting topic for researchers and practitioners. Shi and etc. [1] proposed an approach of decomposing characters into radicals. Tseng and etc. [2-4] suggested a stroke extraction method based on a combination of a simple feature point detection scheme and a novel stroke segment connecting method. In [6], another algorithm was proposed for offline handwritten Chinese character segmentation, each has its advantages and disadvantages.

In our previous paper [5], extending the radical-based algorithm in [1], a spatial shape-based algorithm was proposed to extract the squared-shape characters. We focused on the overlapping and over-splitting problems. The algorithm can be briefly described as: 1) line rotation; 2) a spatial-based method to segment the character strings into radicals; 3) several knowledge-based adjusting operations to solve the over-splitting and overlapping problems. In [2, 3], Tseng and etc. used stroke-based method to extract characters. They first scan the text image row by row to extract each stroke and form the bounding box for each stroke; second, stroke-bounding boxes and over-lapping relationships are used to develop the joint rules for a Chinese character, and then the knowledge-based merging and dynamic programming method are used to find the best way to segment the characters. Since each character usually has a number of strokes with a complex spatial relationships, it requires that handwritten characters are separated and in-line, which can be hardly true in real handwritten Chinese text. In this paper, we present results of many experiments with various Chinese handwritten texts by different people, C++ programs were developed based on the algorithm in [2-5] to test the two algorithms. The paper is organized as follows: the outline and programming considerations are described in section 2, experimental results and comparison are presented in section 3, and finally in section 4, a summary is given.

2. Simulated Outline of Both Algorithms

2.1. Spatial Shape-based Algorithm

The spatial shape-based algorithm includes three major parts: preprocessing, spatial shape-based character segmentation and knowledge-based adjusting.

2.1.1. Preprocessing

Most handwritten text, especially the text written on a blank paper, usually have slant. A line-rotation and segmentation algorithm is proposed, as the pre-processing stage, to separate the text into lines of Chinese character strings. The line-rotation algorithm can be stated by the following procedures:

Step1. Rotate the image with angle,

$$\alpha \in \{[-\pi/12, \pi/12], stepsize: \pi/180\}.$$

Step2. Calculate the sum of each row for each rotation,

$$s_j = \sum_{i=0}^{W-1} A(i,j), \quad j \in \{0,1,2...H-1\}.$$

Step3. Calculate the standard deviation σ,

$$s = \frac{1}{H}\sum_{j=0}^{H-1} S_j, \ \sigma_\alpha^2 = \frac{1}{H}\sum_{j=0}^{H-1}(S_j - S)^2.$$

Step4. Find the maximum σ with respect to angle $\alpha_{\max}$.

Step5. Rotate the original image by the angle $\alpha_{\max}$.

Where W is the width of the image, and H is the height of the image.

Then a line segmentation algorithm is used to separate the text into character string stripes. The line segmentation algorithm includes the following steps:

Step1. Calculate the sum of the pixel value for each row again after the rotation.

Step2. Find the average of the sum of the pixel value for each row.

Step3. Find the top and the bottom rows for each stripe of characters.

Step4. Remove the effect of noise in the result: If bottom-top<k, then remove this stripe.

2.1.2. Spatial Shape-based Character Segmentation

After the text is separated into stripes of character strings, each stripe is segmented into characters using spatial shape-based method given by the following steps:

Step1. Select one stripe of character strings and find the value of the top and bottom. Calculate the sum of every column of this stripe.

Step2. Calculate the average of the sum of every column of this stripe.

Step3. Find the possible left and right of every character.

Step4: Find the next character and redo the steps 1~3 until the last string is done.

2.1.3. Knowledge-based Adjusting

The previous procedure produces a lot of slender boxes. A knowledge based adjusting algorithm is used to merge these boxes. We calculate the average width of the character radicals first, then find all the boxes of this kind by testing them one by one, to see if there is another slender box near this one, and decide if they belong to the same character. The rule is: if the distance between the two boxes is less than one-fifth of the characters' average width and the two boxes' merging result is less than the double of average width, then the two boxes belong to the same character and merge them.

After this operation, the following possible errors may exist: one box includes one and half characters and the next box has the other half of the second character. Two characters are included in the same box.

To correct the above two possible errors, the following steps were proposed:

Step1. Find all boxes with width more than one and half of the average width.

Step2. Separate those boxes at their best segmentation point (smallest sum of every column of this stripe).

Step3. Test the width of the two new boxes and decide if each is only part of a character.

Step4. If we find that one of the boxes seems to be half of the character, then we test the other box next to it, and to see if it includes half of a character and decide if we should merge them.

An example is given in Figures 1~4 to illustrate all the procedures given above for the spatial shape-based algorithm.

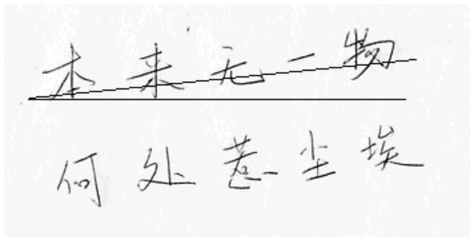

Figure 1. **Slanted characters**

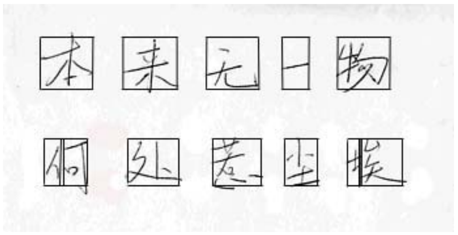

Figure 2. **After pre-processing**

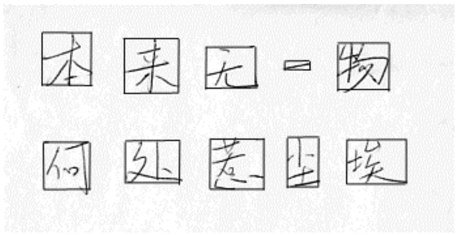

Figure 3. **After segmenting**

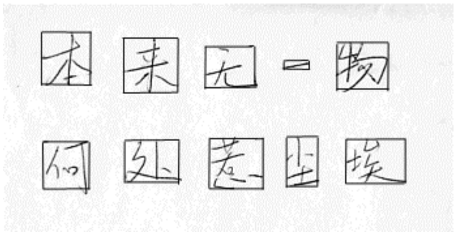

Figure 4. **After adjusting**

2.2 Stroke-based Algorithm

The stroke-based algorithm also has three major parts: extract each stroke and find its bounding box, merge these boxes into a character and then the knowledge-based adjusting.

2.2.1. Stroke Extraction

Step1. Trace the dark stripes from top to bottom, row by row. If the stripe has not yet been traced and it is not noise, record this starting point, the location and width of this stripe. Continue to trace this stroke in the effective area (one pixel wider on both sides than the dark area of the previous row) of the next row. Dynamically adjust the effective area of each row to find all possible continuous pixels to the dark strips in the effective area.

Step2. If the widths of neighbor strips become narrower, then it is the end of a stroke. Otherwise, it is the crossing area.

Step3. If there is more than one stripe in the effective area of the next row, choose the one has similar position and similar width of the current stripe.

Step4. If the stroke extracts noise, then change it to the blank background.

Step5. Record the stroke extracted and a stroke-bounding box is defined as the smallest rectangle that contains this stroke. If the sum of pixels of this stroke is less than a constant (determined by experience), then it is noise. Record the coordinate value of the stroke bounding box into an array and assign a number to the pixels of this stroke based on its type. We gave a specific value for each type of stroke in our program.

Step6. Redo the steps 1-5 after the current stroke is extracted. The starting point of each loop is the previous starting point. We do this until we process all the dark strips.

2.2.2 Merge Stroke-bounding Boxes

Step1. With the information of the stroke types assigned (horizontal, vertical, up-left-slanting, up-right-slanting) and the stroke lengths (long stroke, short stroke), we can classify the strokes into eight types. According to the coordinates of their top-left points, all stroke-bounding boxes are sorted by the X-coordinates in non-decreasing order. If the X-coordinates become the same, then the sorting process changes to sort the Y-coordinates, also in ascending order.

Step2. Merge two boxes using relative positions of their up-left point and bottom-right point. First, only merge those boxes corresponding to short strokes. Second, merge those boxes corresponding to the slanting-strokes. Third, if two boxes are overlapped with each other more than half of the width of either box and the width of the merged box is less than *K* (determined by experience), then they will be merged. Fourth, similar to the third step except that the overlapped area is one third of the width of either box. A series of bounding-boxes of Chinese characters are created.

2.2.3 Knowledge-based Adjustment

After a series of bounding-boxes of characters are created, a knowledge-based adjustment is conducted by using dynamic programming method to make certain that each connected bounding-boxes is a whole character. We get the dissimilarity of current boxes with reference boxes whose width is about thirty percent of the width of the current box. Through the comparison, we try to remove this dissimilarity.

In Figures 5~7, an example is given to illustrate the stroke-based algorithm.

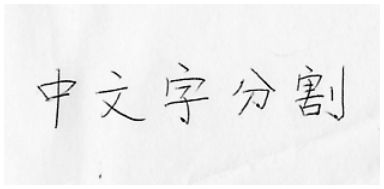
Figure 5. **Original characters**

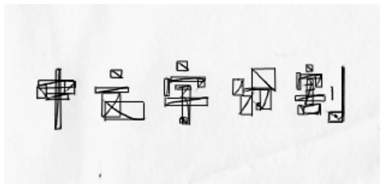
Figure 6. **After stroke extraction**

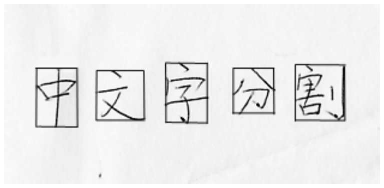
Figure 7. **After merging**

3. Experimental Results and Comparison

All the experiments are conducted on a PC, and all programs are written in C++ for performance. Our experimental results show that both algorithms performed similarly in some cases while the performance can be significantly different in other cases. Here we discuss a few situations where their performance is similar or different.

a). Both algorithms have integrated the preprocessing step to eliminate the noise. The spatial shape-based algorithm, in the pre-processing stage, determines weather it is noise by calculating the height of the character. While the stroke-based algorithm determines whether a stroke is noise by calculating its sum of pixels. Under certain conditions, when it is a single long strip having many pixels, it cannot be identified as noise to be removed. An example is given in Figures 8~10 to illustrate the situation.

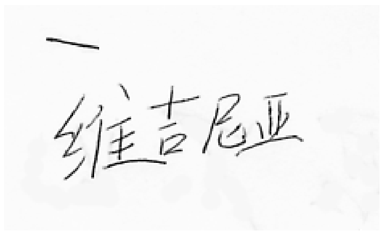

Figure 8. **Original image**

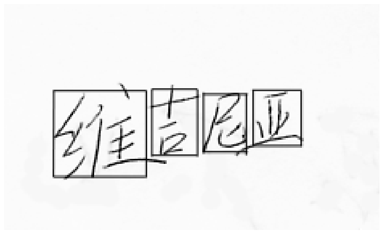

Figure 9. **Segmentation result using the spatial algorithm**

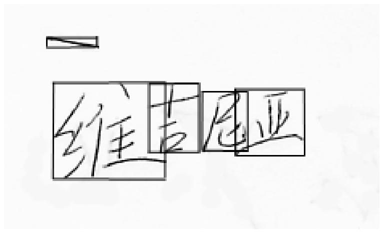

Figure 10. **Segmentation result using the stroke algorithm**

b). For both algorithms, there are several parameters need to be determined manually by experiments, experience, character fonts, different written styles and gray-level of the image. Self-adaptable algorithms need to be developed in the future.

c). Computing speed: we ran both programs with a large number of scanned Chinese texts as input images, our experimental results show that the spatial shape-based algorithm is faster than the stroke-based algorithm. The time complexity for the spatial shape-based algorithm is O (n^2) (double loops) while for the stroke-based algorithm the time complexity is O (n^3) (triple loops), where *n* is the total number of pixels in the scanned texts.

d). Both algorithms can deal with over-splitting and overlapping problems well in their adjustment process. An example is given in Figures 9 and 10. The spatial-based algorithm set top and bottom lines first to enclose each character stripe, it cannot get the whole Chinese character when a character is way too high than its neighbors. That means if a character is much higher than the other characters in the same stripe, part of its information will be lost.

e). Handwritten Chinese characters can be artistically written. A number of curved strokes in a character can make the handwritten Chinese full of diversification, which increases the difficulty in characters' segmentation and recognition. For the spatial-based algorithm the characters with curved strokes can be successfully segmented, since we do not extract the characters based on its structure and strokes, while the stroke-based algorithm cannot recognize the curved strokes. An example is given in Figure 11 with Chinese characters "□", "□", where the spatial-based algorithm is used.

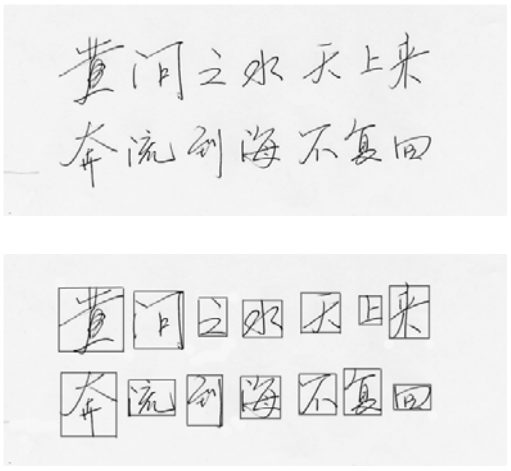

Figure 11. **Curved and continuous strokes recognized by the spatial-based algorithm**

f). The spatial-based algorithm can be applied to all type of squared-shape character segmentation, such as: Korean, Japanese and etc. Korean language has circular strokes. An example is given in Figure 12 shows that the spatial-based algorithm can segment Korean language characters well. Since the stroke-based algorithm classifies the strokes into four types: horizontal, vertical, up-left-slanting and up-right-slanting, it is hard to deal with the characters with circular type of strokes.

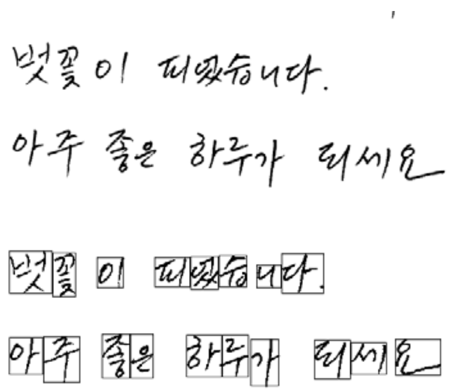

Figure 12. **Segmentation of Korean characters by the spatial-based algorithm**

Summarizing the above performance discussion, we give the performance comparison of both algorithms in Table 1.

	Spatial-based Algorithm	Stroke-based Algorithm
Time Complexity	O (n^2)	O (n^3)
Accuracy	Various based on handwriting	Various based on handwriting
Noise removal	Good for long single string removal	Good for pepper noise removal
Parameter Adjustment	Manual	Manual
Curve-shaped Character	Yes.	No.
Application Area	Chinese, Korean, Japanese characters	Chinese characters

Table 1. **Overall comparison of both algorithms**

4. Summary

A comprehensive comparison of both the spatial-based algorithm and the stroke-based algorithm for handwritten Chinese character segmentation is given in this paper. Our experimental results show that the spatial-based algorithm is more efficient in terms of computing speed, while the stroke-based algorithm can segment handwritten Chinese characters when they are written with various heights. When handwritten Chinese characters possess artistic style with circular curves and continuous strokes, the spatial-based can separate those characters while stroke-based algorithm cannot, which in addition it shows possible application to segment other square-shaped characters, such as Korean and Japanese. More work needs to be done with automatic parameter adjustment in the programming consideration to make the algorithm less dependant and more robust. The next step before we can proceed to character recognition is to adjust the segmented bounding boxes to the same size for each character.

References

[1] Daming Shi, Robert I. Damper, Steve R. Gunn "An Approach to Off-Line Handwritten Chinese Character Recognition Based on Hierarchical Radical Decomposition", Journal of Quantitative Linguistics, Volume 10, Number 1 / April 2003

[2] L.Y. Tseng, C.T. Chuang, "An Efficient Knowledge-Based Stroke Extraction Method for Multi-font Chinese Characters", Pattern Recognition Vol. 25., 1997

[3] L.Y. Tseng, R.C. Chen, "A new method for segmenting handwritten Chinese Characters", Proceedings of the 4th International Conference on Document Analysis and Recognition (ICDAR'97).

[4] Feng Lin, Xiaoou Tang "Off-Line Handwritten Chinese Character Stroke Extraction", ICPR'02 Volume 3, August 2002

[5] Han Zhang, Chao Lu, "An Algorithm for Segmenting Handwritten Squared-Shape Characters", SNPD 2004, July, Beijing, China.

[6] S.Y. Zhao, Z.R. Chi, P.F. Shi and Q. Wang, "Handwritten Chinese character segmentation using a two-stage approach", Proceedings of the 6th International Conference on Document Analysis and Recognition, (ICDAR'01).