Touching Character Separation in Chinese Handwriting Using Visibility-Based Foreground Analysis

Liang Xu, Fei Yin, Qiu-Feng Wang, Cheng-Lin Liu
National Laboratory of Pattern Recognition(NLPR)
Institute of Automation, Chinese Academy of Sciences
95 Zhongguancun East Road, Beijing 100190, P.R. China
Email: {lxu,fyin,wangqf,liucl}@nlpr.ia.ac.cn

***Abstract*—In offline handwritten text recognition, the separation of touching characters remains a challenge due to the variability of touching structures. This paper proposes a new touching character separation method for Chinese handwriting based on skeleton analysis and contour analysis incorporating the visibility of separating points. Separating points are detected from strokes that are common in both upper and lower skeleton tracing, and the profile visibility of strokes and separating points is analyzed to adjust and verify separating points. Our experiments on two large handwriting databases demonstrate the effectiveness of the proposed method.**

***Keywords*-touching character separation; skeleton analysis; contour analysis; visibility analysis; segmentation**

I. Introduction

In handwritten text recognition, character segmentation is a difficult problem due to the variability of character size and between-character gap. Offline handwriting recognition also encounters touching characters. To overcome the ambiguity of character segmentation, the integrated segmentation-recognition (ISR) strategy is often adopted: the character string is over-segmented into primitive segments with the hope that each segment composes a character or a part of characters, and candidate characters formed by concatenating primitive segments are then verified by character recognition incorporating contexts. The success of ISR, however, relies on the separation of touching characters as well as some other factors. The separation of touching characters is a challenge due to the variability of touching structures, especially in Chinese handwriting which has complicated character structures.

The problem of touching character separation is not solved though many efforts have been done. According to the features used in segmentation, the methods proposed so far can be categorized into three groups: foreground-based methods (including projection analysis [1], contour analysis [2], [3] and skeleton analysis [4], [5], [6], [7]), background-based methods, and combined foreground and background methods [8], [9], [10]. These methods have shown effects in different situations but have their respective insufficiency. The vertical projection-based method [1] cannot deal with touching characters that are overlapping horizontally, e.g., the pattern in Fig. 7(d). The local contour analysis method in [2] is based on single-stroke area analysis, and is also sensitive to character overlapping. The contour DTW method in [3] can detect most true touching points but yield too many redundant separating points. The methods based on foreground and background thinning [9], [10] encounter many spurious skeleton branches for Chinese characters. Skeleton analysis facilitates separating point detection because it provides direct clues of strokes and fork points. The method in [4] detects separating points from horizontal ligatures, but does not consider the slanted and overlapped strokes. The methods in [5], [6] filter skeleton feature points using projection analysis, which is sensitive to character overlapping. The graph-based method in [7] can handle various touching types including multiple connection, but was not evaluated on a large dataset.

In this paper, we propose a foreground analysis-based method for separating touching characters in Chinese handwriting. Our aim is to detect most of true touching points while generates as less redundant separating points as possible. In order to locate the touching point in overlapping characters, we extend the analysis of single-stroke area in [2] to visible foreground area, e.g., Fig. 6(b) and Fig. 6(c). We measure the visibility of strokes based on the top and bottom profiles of the touching pattern, and incorporate the visibility into the detection of candidate separating points from skeleton strokes and verify the separating points. Visibility analysis guarantees that both non-overlapping and overlapping touching points can be detected, and can filter out many redundant separating points. We evaluated the performance of the proposed method on two large databases of Chinese handwriting. Compared to a representative previous method in [2], the proposed method yields significantly higher detection rate of touching points and meanwhile the percentage of redundant separating points is moderate.

II. Overview of the separation algorithm

The flowchart of our proposed method is shown in Fig. 1. It consists of three stages: preprocessing, touching pattern detection, and touching pattern separation.

In preprocessing, we first smooth the string image to remove noises on the contour. Then we extract the connected

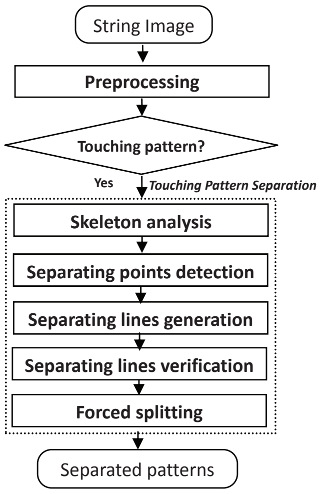
String Image
Preprocessing
Touching pattern?
Yes
Touching Pattern Separation
Skeleton analysis
Separating points detection
Separating lines generation
Separating lines verification
Forced splitting
Separated patterns

Figure 1. Flowchart of proposed separation method

components from the smoothed image. And we merge the connected components which are highly overlapped in horizontal direction. Average stroke width (SW) and string image height (IH) are estimated as done in [2].

In the touching pattern detection, we select the component with large width or width-to-height ratio as a possible touching pattern.

- For each component with width cw and height ch, it is identified as a candidate touching pattern, if $cw > \theta_1 \times IH$ or $cw/ch > \theta_2$, where θ_1 and θ_2 are empirically set as 0.6 and 0.8.

The possible touching pattern(s) will undergo the process of touching pattern separation.

The touching pattern separation process is made up of five steps as shown in dotted rectangle in Fig. 1. We firstly extract the candidate strokes by tracing the upper and lower skeleton. Secondly, we detect separation points on candidate strokes, with stroke visibility analysis. Thirdly, we form possible separating lines from the separating points, by the contour analysis of the touching pattern. Fourthly, separating lines are verified by their visibility difference and some other constraints. Finally, we apply forced splitting to generate some separating lines, which may be complementary when the touching point is not on the candidate strokes. All the remaining separating lines are used to separate the touching pattern.

Since the single-touching character pattern is the dominant case in practical documents, we focus on the method to separate it in this paper.

III. Details of the separation algorithm

In this section, we only describe the details of the first four main steps in the touching pattern separation process, because forced splitting is the same as the method in [2].

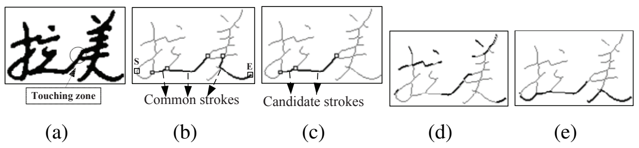

Figure 2. Skeleton analysis. (a) A touching pattern, (b) its skeleton image, with common fork points (marked by a square) and common strokes (in bold black), (c) candidate strokes, (d) the top profile of the skeleton image (in bold black), (e) the bottom profile.

A. Skeleton analysis

We define the common part between two adjacent fork points on the skeleton as a common stroke shown as Fig. 2(b). And in practice, we find that most of the separation points locate at the common stroke. In order to extract the common strokes, we firstly get the skeleton of the possible touching pattern using the thinning algorithm in [11]. The leftmost point of the skeleton is selected as the start point (S), and the rightmost point is selected as the end point (E), as shown in Fig. 2(b). Then we trace the skeleton from S to E in two directions: clockwise (in upper contour) and counterclockwise (in lower contour). Common fork points are marked during tracing, which have more than two 8-connected branches. A common stroke consists of the common points between two adjacent fork points (or S or E). All common strokes found in this step are kept for further processing.

Though most of common strokes mentioned above may contain the separation points, there are still some redundant ones, which can be identified with the help of stroke visibility defined below.

- **Stroke visibility**: If there are points on the top profile, we call the stroke *top-visible*; If there are points on the bottom profile, we call the stroke *bottom-visible*; If the stroke is both top-visible and bottom-visible, we call the stroke *top-bottom-visible*.

Fig. 2(d) and Fig. 2(e) show an example of the top and bottom profile of the skeleton image, respectively. And we decide a redundant common stroke by the following two rules:

- RULE1: if a stroke is not top-visible and bottom-visible, it is redundant.
- RULE2: if there are more than one common strokes, and a stroke begins with S or ends with E, and with a short top-bottom-visible part, the stroke is redundant, as shown in Fig. 2(c).

After removing all the identified redundant common strokes, we keep the remaining ones as candidate strokes to detect separation points.

B. Separating points detection

We detect possible separating points from the candidate strokes detected as above. According to [5], a touching point (which we aim to separate) is usually a turning point or fork point on the skeleton. First, we extract the turning points on a stroke by the corner detection algorithm [12] and polygonal approximation algorithm [13]. All the turning points are kept as candidate separating points.

We form candidate separating points by analyzing the fork points on candidate strokes. Though it is fairly well to separate at a fork point, there will be errors in the following two occasions:

- When a touching point is on a ligature, it is usually impossible to extract a turning point or fork point on the touching zone, as shown in Fig. 3(a) and Fig. 3(b).
- When a fork point is shared by two top-bottom-visible candidate strokes, it is likely to form a separating point on each stroke, without the recognition information, as shown in Fig. 3(c) and Fig. 3(d)

As a result, we may form a possible separation point by adjusting a fork point with some heuristics. We adjust a fork point on a stroke only if the following three conditions are all satisfied:

- Visibility-difference: if a stroke is top (or bottom)-visible and its fork point is not on the top (or bottom) profile, we may move the fork point to a nearby visible point.
- Long-candidate-stroke: the horizontal distance of a candidate stroke should not be very short.
- Short-moving: the horizontal distance where a fork point move should not be too long.

Fig. 4(b) shows an example of an adjusted fork point on a candidate stroke. A separating point is formed by visual adjustment of the right fork point. The left fork point fails to move for it doesn't satisfy the short-moving condition. In a word, for a fork point on a stroke, we keep either the adjusted fork point or unadjusted fork point as a candidate separating point. Fig. 4(c) shows an example of two unadjusted fork points, with all candidate separating points in Fig. 4(d).

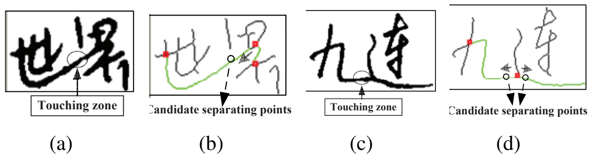

Figure 3. Two motivations for visibility adjustment. (a) A touching pattern with a ligature, (b) a candidate separating point from visual moving of a fork point (marked by a red square), (c) a touching pattern, (d) two candidate separating points from visual moving of a fork point.

C. Separating lines generation

We generate possible separating lines basing on the separating points mentioned above. Since a separating line is formed by connecting two boundary points (i.e. upper and lower terminal), it is necessary to find two matched terminals for a separating point by contour analysis on a touching pattern. We first trace the outer contour of the touching pattern image, similar with previous skeleton tracing. After

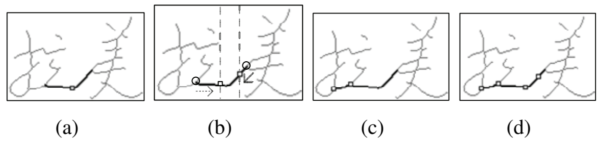

Figure 4. Separating point detection. (a) A turning point (marked by a square) on a candidate stroke (in bold black), (b) an adjusted fork point. Only the right fork point is adjusted successfully according to the stroke visibility. (c) Two unadjusted fork points, (d) candidate separating points.

tracing, we get the upper contour and lower contour, together with their corner points detected by two algorithms [12], [13], as shown in Fig. 5(b) and Fig. 5(c). Then we try to form a corresponding separation line for each separation point, with the contour information.

There are three kinds of separating points: turning point, adjusted fork point and unadjusted fork point. For a turning point or an adjusted fork point, we just form a vertical separating line. Basing on the feature point, we search two boundary points on the touching pattern, both upward and downward. Fig. 5(a) shows an example.

For a unadjusted fork point, we attempt to match a upper or a lower contour corner point on the touching pattern. We apply the following constraint to the matching process.

- The coordinates (x,y) of a corner point on upper-contour or lower-contour to be matched must satisfy $x_{fork} - x_{bound} < x < x_{fork} + x_{bound}$ and $y_{fork} - y_{bound} < y < y_{fork} + y_{bound}$ where x_{fork} and y_{fork} are the coordinates of a unadjusted fork point, and x_{bound} and y_{bound} are the parameters restricting the searching region.

If found, the nearest upper corner point is selected as the upper terminal. Similarly, the nearest lower corner point is selected as the lower terminal. If only one side of corner point is matched, we form a vertical separating line from the matched corner point. If neither side of corner points are matched, we form a vertical separating line basing on the fork point. Fig. 5(d) shows an example.

Finally, we may remove some unsuitable separating lines whose upper (or lower) terminal is not on the upper (or lower) contour. All the remaining separating lines are kept for further verification.

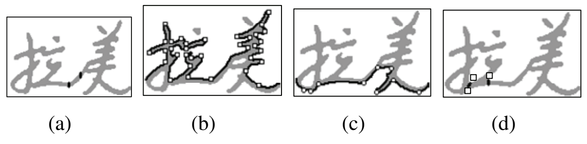

Figure 5. Separating line generation. (a) Two vertical separating lines corresponding to a turning point and an adjusted fork point, in Fig. 4(a) and Fig. 4(b), (b) the upper contour (in black) and its corner points (marked by a square), (c) the lower contour and its corner points, (d) two separating lines corresponding to two unadjusted fork points in Fig. 4(c).

D. Separating lines verification

We first apply the following four empirical validation rules to remove some redundant separating lines from above step: length, foreground pixel ratio, crossing long vertical stroke, and length difference of neighboring lines, as done in [3].

In order to further remove redundant separating lines, we give a definition of the separating-line-visibility below.

- **Separating-line-visibility**: If the upper terminal is on the top profile, we call a separating line *top-visible*; If the lower terminal is on the bottom profile, we call a separating line *bottom-visible*; If a separating line is both top-visible and bottom-visible, we call a separating line *top-bottom-visible*.

Fig. 6(b) and Fig. 6(c) show an example of the top and bottom profile of a touching pattern, respectively.

In order to compare the visibility difference, we compute the visible-distance of a separating line as following:

$$\text{top-vis-dist} = \begin{cases} |dx_1^t| & L_1^t < L_2^t; \\ |dx_2^t| & \text{otherwise.} \end{cases}$$

$$\text{botm-vis-dist} = \begin{cases} |dx_1^b| & L_1^b < L_2^b; \\ |dx_2^b| & \text{otherwise.} \end{cases}$$

$$\textbf{visible-distance} = max\{\text{top-vis-dist}, \text{botm-vis-dist}\} \quad (1)$$

where dx_1^t and dx_2^t are the horizontal distance between the upper terminal and the nearest top-visible contour point on upper contour, both in left and right direction, L_1^t and L_2^t are the number of contour points traced in two directions. dx_1^b, dx_2^b, L_1^b, and L_2^b are defined similarly for bottom-visible on lower contour. If no top-visible or bottom-visible contour points are found in one direction, a very large number is assigned to dx and L.

After we compute the visible-distance value of all candidate separating lines by (1), we check their redundancies. We first find the separating line with the minimum value of visible-distance. Then we calculate the difference of visible-distance value of all other separating lines, with respect to the minimum value. If the value difference of a separating line is large, the separating line is redundant and removed. Finally, we preserve the remaining separating lines to separate the touching pattern. Fig. 6(d) shows an example.

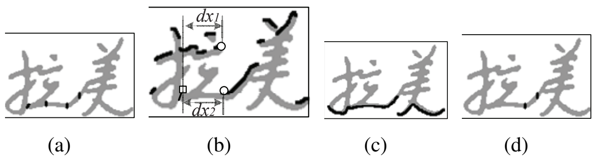

Figure 6. Separating lines verification. (a) Candidate separating lines, (b) the top profile(in bold black) and a separating line's dx_1^t and dx_2^t, (c) the bottom profile, (d) separating lines after verification

IV. Experimental results

We first introduce the experiment database and two other compared methods. Then we describe our performance evaluation. Finally, we discuss the reasons for some typical errors of our method.

A. Databases and compared methods

We used two public Chinese handwriting databases to assess the separation algorithms. One is HIT-MW [14] database, and the other is CASIA-HWDB2.1 [15] database. Both two databases have been annotated manually recently. In the experiment, we used 6,543 single-touching character pairs from HIT-MW database, and extracted 10,351 single-touching character pairs from CASIA-HWDB2.1 database.

We compare our proposed method in this paper with other two methods: (1) One is our previous work which only uses contour analysis [3]. (2) The other one is the segmentation method which was applied in the Japanese handwritten mail address recognition [2].

B. Performance evaluation

We evaluate a separation using the distance (d) from the obtained separating point to the annotated touching point. And we decide a correct separation when d is less than a threshold d_{th}, which is empirically set as $2 \times SW$. The overall performance of a separation method is measured by the recall rate R and the precision rate P, defined below.

$$R = \frac{N_c}{N_t} \times 100\%; P = \frac{N_c}{N_s} \times 100\% \quad (2)$$

where N_c, N_t, and N_s are the number of correct separating lines, ground-truth separating points and all detected separating lines, respectively.

Experimental results on HIT-MW and CASIA-HWDB2.1 database are listed in Table I and II, respectively. In the table, "*verify*" denotes that we apply all the rules to remove possible redundant common strokes and separating lines. "*verf_slv*" denotes that we apply only one rule of the separating-line-visibility verification. The results show that comparing with [2], our approach improves R (53.7%→77.2%, 64.0%→84.1%), with a moderate P. Besides, comparing with [3], without any verification, our approach improves R (89.0%→90.6%, 89.5%→92.4%). Moreover, the separating-line-visibility verification improves P of our approach (25.1%→35.2%, 29.4%→42.9%) and [3] (17.3%→29.9%, 18.8%→32.8%). The improvement of P will decrease the number of primitive segments and thus reduce the searching cost in the ISR strategy, which helps to improve the performance of string recognition. Fig. 7 shows some examples of separation results on three touching character strings.

C. Error analysis

There are mainly two types of separation error detected. The first type is caused by wrongly removing of a common stroke by two rules in the skeleton analysis, as shown in Fig. 8(a). The second error type results from the case where a touching point is wrongly removed by the separating-line-visibility verification, as shown in Fig. 8(b).

Table I
PERFORMANCE COMPARISON ON HIT-MW

Method	R (%)	P (%)	N_c	N_s
Liu et al. [2]	53.7	52.0	3,512	6,748
Previous method [3]	89.0	17.3	5,822	33,729
Proposed method	77.2	35.2	5,053	14,358
Proposed (no *verify*)	90.6	13.6	5,931	43,750
Proposed (no *verf_slv*)	83.7	25.1	5,476	21,851
Previous [3] + *verf_slv*	78.6	29.9	5,145	17,209

Table II
PERFORMANCE COMPARISON ON CASIA−HWDB2.1

Method	R (%)	P (%)	N_c	N_s
Liu et al. [2]	64.0	66.2	6,290	10,014
Previous method [3]	89.5	18.8	9,260	49,350
Proposed method	84.1	42.9	8,704	20,293
Proposed (no *verify*)	92.4	14.5	9,567	66,143
Proposed (no *verf_slv*)	86.4	29.4	8,943	30,398
Previous [3] + *verf_slv*	83.3	32.8	8,618	26,294

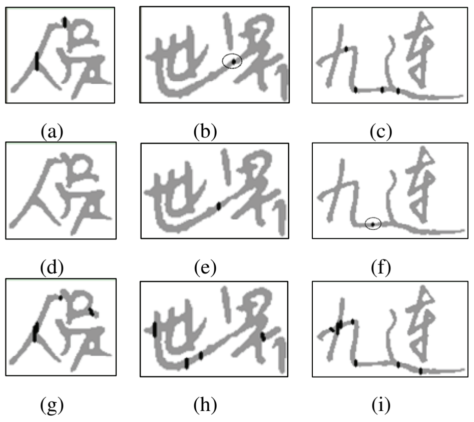

Figure 7. Examples of separation results for comparison. (a-c) Separating lines by our proposed method(in black), (d-f) separating lines by the method [2], (g-i) separating lines by our previous method [3].

V. CONCLUSION

We presented in this paper a new method for touching character separation in Chinese handwriting. The main feature of the method is the combination of skeleton analysis and contour analysis, as well as the visibility analysis of separating points for filtering out redundant separating points efficiently. Our preliminary experiments on two large databases demonstrated that the proposed method can locate most of between-character boundaries, with a moderate percentage of redundant separating points. In the future, we will extend the separation algorithm to characters of multiple touching.

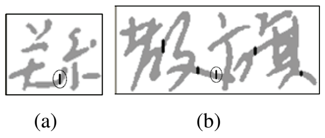

Figure 8. Examples of separation errors detected. (a-b) Separating lines by our proposed method (in bold black), and the ground-truth (in thin black within an ellipse).

ACKNOWLEDGMENT

This work is supported by the National Natural Science Foundation of China (NSFC) under grants no.60825301 and 60933010. The authors thank Dr. Tonghua Su for helpful discussions and providing the database.

REFERENCES

[1] H. Ikeda, Y. Ogawa, M. Koga, H. Nishimura, H. Sako, H. Fujisawa. A recognition method for touching Japanese handwritten characters, *Proc. 5th ICDAR*, 1999, pp.641-644.

[2] C.-L. Liu, M. Koga, H. Fujisawa. Lexicon-driven segmentation and recognition of handwritten character strings for Japanese address reading, *IEEE Trans. Pattern Anal. Mach. Intell.*, 24(11): 1425-1437, 2002.

[3] L. Xu, F. Yin, C.-L. Liu. Touching character splitting of Chinese handwriting using contour analysis and DTW, *Proc. 2010 Chinese Conference on Pattern Recognition(CCPR)*, 2010, pp.814-818.

[4] J. Gao, X. Ding, Y. Wu. A segmentation algorithm for handwritten Chinese character strings, *Proc. 5th ICDAR*, 1999, pp.633-636.

[5] T. Yamaguchi, T. Yoshikawa, T. Shinogi, S. Tsuruoka, M. Teramoto. A segmentation method for touching Japanese handwritten characters based on connecting condition of line, *Proc. 6th ICDAR*, 2001, pp.837-841.

[6] T. Yamaguchi, S. Tsuruoka, T. Yoshikawa, T. Shinogi, E. Makimoto, H. Ogata, M. Shridhar. A segmentation system for touching handwritten Japanese characters, *Proc. 8th IWFHR*, 2002, pp.407-412.

[7] M. Suwa. Segmentation of touching handwritten Japanese characters using the graph theory method, *Proc. SPIE*, Vol. 4307, 2001, pp.280-289.

[8] Y.-K. Chen, J.-F. Wang. Segmentation of single- or multiple-touching handwritten numeral string using background and foreground analysis, *IEEE Trans. Pattern Anal. Mach. Intell.*, 22(11): 1304-1317, 2000.

[9] S. Zhao, Z. Chi, P. Shi, H. Yan. Two-stage segmentation of unconstrained handwritten Chinese characters. *Pattern Recognition*, 36(1): 145-156, 2003.

[10] Z. Liang, P. Shi. A metasynthetic approach for segmenting handwritten Chinese character strings, *Pattern Recognition Letters*, 26(10): 1498-1511, 2005.

[11] S. Suzuki, K. Abe. Binary picture thinning by an iterative parallel two-subcycle operation, *Pattern Recognition*, 10(3): 297-307, 1987.

[12] A. Rosenfeld, E. Johnston. Angle detection on digital curves, *IEEE Trans. Computers*, 22: 875-878, 1976.

[13] U. Ramer. An iterative procedure for the polygonal approximation of plane closed curves, *Comput. Graphics Image Process*, 1: 244-256, 1972.

[14] T. Su, T. Zhang, D. Guan. Corpus-based HIT-MW database for offline recognition of general-purpose Chinese handwritten text, *Int. J. Document Analysis and Recognition*, 10(1): 27-38, 2007.

[15] C.-L. Liu, F. Yin, D.-H. Wang, Q.-F. Wang. CASIA online and offline Chinese handwriting databases, *to submit to ICDAR2011*.