IJDAR (2014) 17:91–104
DOI 10.1007/s10032-013-0208-1

ORIGINAL PAPER

An over-segmentation method for single-touching Chinese handwriting with learning-based filtering

Liang Xu · Fei Yin · Qiu-Feng Wang · Cheng-Lin Liu

Received: 26 January 2013 / Revised: 26 March 2013 / Accepted: 8 June 2013 / Published online: 22 June 2013

Abstract The segmentation of touching characters is still a challenging task, posing a bottleneck for offline Chinese handwriting recognition. In this paper, we propose an effective over-segmentation method with learning-based filtering using geometric features for single-touching Chinese handwriting. First, we detect candidate cuts by skeleton and contour analysis to guarantee a high recall rate of character separation. A filter is designed by supervised learning and used to prune implausible cuts to improve the precision. Since the segmentation rules and features are independent of the string length, the proposed method can deal with touching strings with more than two characters. The proposed method is evaluated on both the character segmentation task and the text line recognition task. The results on two large databases demonstrate the superiority of the proposed method in dealing with single-touching Chinese handwriting.

Keywords Single-touching strings · Chinese handwriting · Over-segmentation · Learning-based filtering · Geometric features

L. Xu (✉) · F. Yin · Q.-F. Wang · C.-L. Liu
National Laboratory of Pattern Recognition, Institute of Automation of Chinese Academy of Sciences, 95 Zhongguancun East Road, Beijing 100190, People's Republic of China
e-mail: lxu@nlpr.ia.ac.cn

F. Yin
e-mail: fyin@nlpr.ia.ac.cn

Q.-F. Wang
e-mail: wangqf@nlpr.ia.ac.cn

C.-L. Liu
e-mail: liucl@nlpr.ia.ac.cn

1 Introduction

To deal with the ambiguity of character segmentation, handwritten text recognition is usually accomplished by integrated segmentation and recognition [1]. An effective approach is to over-segment the text line (character string) into primitive segments each being a character or a part of character, combine adjacent primitive segments into candidate characters and evaluate candidate characters using a character recognizer and contexts. The over-segmentation of touching characters, which occur frequently in handwritten documents, is a challenging task and is crucial to the string recognition performance [2–6].

There have been many over-segmentation algorithms for dealing with touching characters, but most of them are based on heuristic rules [2,6–17]. This makes them hard to generalize from one application problem to another. Even if using problem-specific rules, there still remain many over-segmentation failures (under-segmentation errors) (cf. the experimental results in [3]). On one hand, we need to devise splitting algorithms to improve the recall rate of cut (separating line) detection. On the other hand, it is demanding to design a learning-based over-segmentation algorithm to improve its robustness [18], specifically, to better balance the recall rate and precision. The cut classification algorithm of Bayer and Kressel [19] can be viewed as a learning-based one, but classification on each column of string image is too computationally demanding. A more promising strategy is to over-generate candidate cuts using heuristic shape analysis and then filter out redundant cuts using a classifier [20]. In either cut detection or filtering, the considered geometric features or rules are hoped to be independent of the string length (number of characters in the string image) because the string length is unknown a priori. However, some previous methods have assumed known string length (e.g., character pair)

or use features depending on the string length (e.g., [21]). These methods are hard to generalize to touching strings of variable length.

In Chinese handwriting, character segmentation is difficult due to the large category set, complex character structure and writing style variability [3,22–24]. The segmentation of touching characters in Chinese and Japanese handwriting has drawn considerable attention but is still a bottleneck that hinders high performance in text line recognition [3]. To detect touching points with a high recall rate can guarantee that different characters are mostly separated in over-segmentation. However, a high recall rate is usually accompanied with many redundant cuts (false positives), which make the segmentation-recognition candidate path evaluation and search in string recognition complicated and may deteriorate the recognition performance. We thus have been exploring over-segmentation algorithms for achieving high detection rate of touching points while compressing false positives.

To enable fair evaluation of Chinese touching character segmentation algorithms, we recently released a touching character database called CASIA-HWDB-T [25], compiled from the Chinese handwriting database CASIA-HWDB [26]. The database reveals that single-touching strings (in which two adjacent characters are touched with one connected stroke) are the dominant type (more than 95 %) among all touching samples. The focus of this work is thus to deal with this majority type of single-touching. Figure 1 shows some examples of single-touching strings and multiple-touching ones. In our previous work [27], we developed a foreground skeleton-based segmentation algorithm which can detect most touching points with a moderate ratio of redundant cuts. This work extends the previous method and improves the handwritten text line recognition performance mainly in two aspects. First, we refine the cut detection steps to improve the recall rate with moderate computation. Second, we design a supervised learning-based filter for reducing redundant cuts and balance the recall rate and precision. The used geometric features and rules in cut detection and filtering are independent of string length and so are applicable to strings of variable length. Our experimental results on the touching character database CASIA-HWDB-T demonstrate the effectiveness of the proposed method and its benefit to text line recognition.

Fig. 1 Examples of **a** single-touching pair, **b** single-touching string with more than two characters, **c** multiple-touching pair, and **d** touching string with more than two characters including at least a multiple-touching pair [25]

The rest of this paper is organized as follows. Section 2 reviews related works in Chinese handwriting segmentation; Sect. 3 gives an overview of our over-segmentation method; Sect. 4 describes the algorithm for candidate cut generation; Sect. 5 describes the learning-based filter using geometric features; Sect. 6 describes the postprocessing step; Sect. 7 presents the experimental results and Sect. 8 concludes the paper.

2 Related works

Character segmentation in Chinese/Japanese handwriting for character string (text line) recognition has been studied in different application scenarios: bank check reading [9,12], mail address interpretation [2,8,13–16,21,28] and handwritten text recognition [10,11,17,27]. Some of the methods are aimed for segmentation-then-recognition, and some are aimed for integrated segmentation-recognition. The latter case is over-segmentation with the objective of separating characters with as less redundant cuts as possible. In either case, the splitting of touching characters is the main task, as the separation of non-touching characters can be easily achieved by connected component analysis. In the following, we briefly review some existing segmentation methods in Chinese/Japanese handwriting recognition.

Tseng and Chen [10] proposed a segmentation method based on stroke extraction. They extract eight types of strokes by run scanning in binary image. The highly horizontally overlapped strokes are heuristically merged into primitive segments, which are partitioned into characters by dynamic programming incorporating character-like geometric properties. By this method, touching characters can be separated in stroke extraction of different types, but the segmentation performance is influenced by the accuracy of stroke extraction. A similar stroke extraction-based method was presented by Wang et al. [16]. The segmentation method of Tseng and Lee [11] is recognition-based. They generate nonlinear separating lines (cuts) by probabilistic Viterbi search on equally sampled vertical positions of string image and filter out redundant cuts by heuristics such as overlapping ratio of adjacent cuts. Separating lines passing through foreground strokes can split touching characters. The final segmentation of characters is determined by shortest path search in a graph of candidate segmentation incorporating character recognition confidence and character-like geometric features.

Suwa [15] proposed a recognition-based touching character segmentation algorithm, using a graph to represent the skeleton of character image with each node denoting a corner or fork point and each edge denoting a stroke segment. For splitting touching characters, potential edge cuts are selected using empirical rules about the node type, the length and slant of edge. The optimal segmentation result is found on

the candidate character lattice with character recognition confidence. The recognition-based segmentation method of Yamaguchi et al. [13,14] similarly represents the foreground skeleton as a graph. They use valleys of projection profile and stroke crossing counts to prune redundant skeleton fork or corner points, and form candidate cuts from the remaining fork and corner points. Further, a heuristic based on the overlapping width of two adjacent candidate characters and the angle of virtual corner is used to filter out unnecessary cuts. Finally, touching pattern image is over-segmented at candidate separating lines. Graph-based segmentation well exploits the stroke structure of touching characters, but the proper selection of edge cuts is not trivial.

The method of Liu et al. [2] detects candidate cuts in potential touching patterns (which are connected components of large width or width-to-height ratio) by local contour analysis in single-stroke regions (horizonal positions with single vertical stroke crossing and limited run length). For detecting touching point, they match the local contour shape with seven empirically defined touching types, extended from those of Ikeda et al. [7]. After splitting based on touching type detection, a postprocessing step based on projection profile is applied to generate extra cuts on components which are still wide enough. This method has shown promise in Japanese mail address reading [2] and recently in Chinese handwritten text recognition [3]. The remaining under-segmentation errors are mainly due to the failure of touching point detection in single-stroke regions.

The segmentation method of Zhao et al. [21] uses the skeletons of both the foreground and the background of binary image. On potential touching patterns, they extract stroke segments from the foreground skeleton, detect candidate segmentation points from fork and corner points and form nonlinear separating lines incorporating both foreground and background skeleton information. They then verify the separating lines using a trained fuzzy decision tree classifier with geometric features (e.g., the width ratio of left and right parts). Some of the geometric features assume character pairs and so are not suitable for touching strings with more than two characters. On the other hand, the background thinning in this method consumes considerable computation and may generate many spurious branches to complicate separating line detection. The method of Liang and Shi [28] similarly detects separating lines from foreground and background skeletons, but verifies them using a mixture density probability model.

The previous works evaluated the performance of character segmentation in different metrics, some in the character string recognition performance (string-level or character-level accuracy), some in the percentage of correctly segmented characters and some from the viewpoint of segmentation point (cut) detection (recall rate, precision and F-measure). Some focus on touching characters or even touching character pairs only, and some count on character strings containing both touching and non-touching characters. Among the works, Suwa [15] reported a correct segmentation rate of 93.8 % on one hundred touching Japanese Kanji characters. Yamaguchi et al. [14] reported a correct segmentation rate of 83.0 % on 456 touching patterns from handwritten mail address images. Zhao et al. [21] evaluated their segmentation method on one thousand handwritten Chinese mail address strings, and as well, on 1,960 touching characters reported 52.1 % segmentation correct rate. Liu et al. [2] reported only string recognition rates on handwritten Japanese mail address images. Wang et al. [3] show that on the test text lines of database CASIA-HWDB, the over-segmentation method of [2] remains 4.46 % under-segmentation errors. These results are not directly comparable since these methods were evaluated on different datasets, and some methods utilized character recognizer while some not.

Along with Chinese character segmentation, there have been many segmentation methods proposed in handwritten digit segmentation [29–33] and other scripts. Some important separating techniques based on contour, stroke and background skeleton analysis were firstly proposed for handwritten digits and then accustomized to Chinese characters (e.g., [21,28]).

3 Overview of our over-segmentation method

Our over-segmentation method for handwritten Chinese text line recognition is aimed for separating characters with as less primitive segments as possible, and particularly for touching characters, finding the separating lines with high recall rate and precision. The block diagram of the method is shown in Fig. 2. By the method, the input character string image (in horizontal orientation) is first segmented into connected components (CCs). After merging highly horizontally overlapped CCs, we treat the resulted components with large width or width-to-height ratio as candidate touching patterns. To judge the width, the string image height, denoted as SI_{height}, is estimated as the mean of height of bounding box of every pair of consecutive components [2]. Then, a component is taken as candidate touching pattern if its width exceeds $\theta_1 \times SI_{\text{height}}$ or the width-to-height ratio exceeds θ_2. θ_1 and θ_2 are empirically set as 0.6 and 0.8, respectively, such that nearly all the touching characters are detected. Each candidate touching pattern is then processed in four steps: construction of candidate separating lines, verification of separating lines by heuristics and by learning-based filtering and postprocessing.

Candidate separating lines are generated based on the foreground skeleton of string image (after thinning) and contour information (before thinning). To save computation, we do not skeletonize the background area. On tracing the skeleton image both from the upper side and from the lower side,

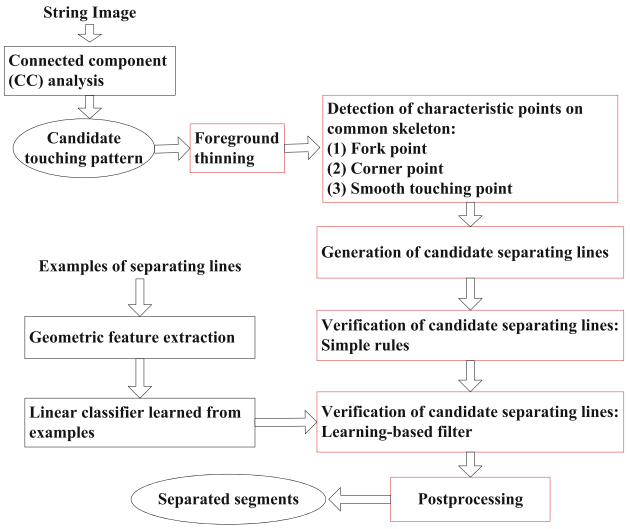


Fig. 2 Block diagram of our proposed over-segmentation method

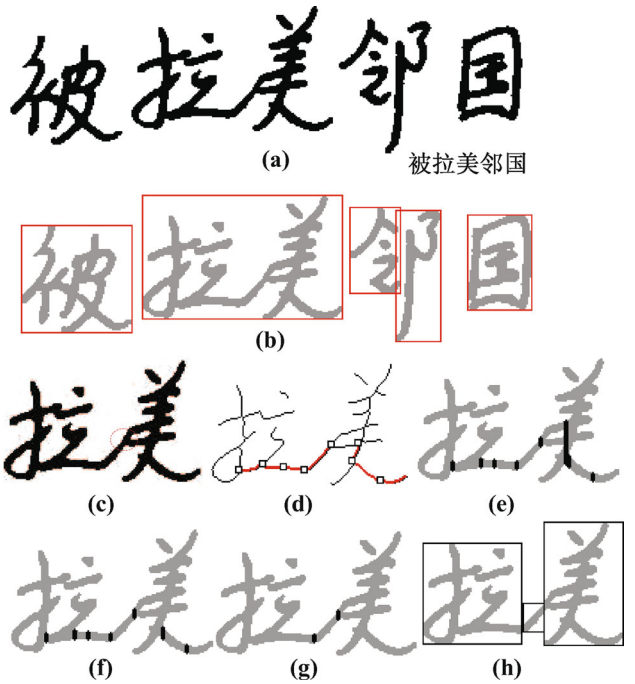


Fig. 3 Example of the over-segmentation steps. **a** Character string image; **b** components after connected component analysis; **c** a candidate touching pattern; **d** characteristic points detected on the common skeleton; **e** candidate separating lines generated from **d**; **f** verified separating lines after simple rules; **g** verified separating lines after learning-based filtering; **h** primitive segments after over-segmentation

the common skeleton (called common stroke) in both upper and lower traces is analyzed to detect characteristic points, including fork points, corner points and smooth touching points. Candidate separating lines are constructed on these characteristic points by connecting corresponding points in the upper and lower contours of the string image (see Sect. 4). Redundant separating lines are filtered out using some simple rules and a learning-based filter with geometric features (see Sect. 5). The postprocessing step is to separate the remaining components which are sufficiently wide to generate extra separating lines by projection analysis (see Sect. 6). After separating the touching patterns at the detected and verified separating lines, the resulted primitive segments are fed into a character string (text line) recognition system. Figure 3 gives an illustrative example of the over-segmentation steps.

4 Construction of candidate separating lines

Candidate separating lines are constructed in two steps: characteristic point detection and candidate separating line generation.

4.1 Characteristic point detection

In single-touching Chinese characters, the touching points are mostly lying on the common skeleton traced from upper side and from lower side. We hence detect candidate separating points only from the foreground skeleton, unlike some previous methods that skeletonize both foreground and background. A similar scheme analyzing common skeleton was adopted for separating touching digits [33].

First, we skeletonize the binary image of touching pattern using a thinning algorithm [34]. If the touching pattern contains multiple CCs, we select the widest CC for foreground skeleton tracing. The pixels of skeleton are traced from left to right clockwise in the upper side and counter-clockwise in the lower side. The left end and right end are denoted as the start point (S) and the end point (E), respectively. On the common part of the upper and lower traces, the fork points (those have more than two eight-connected branches) are marked as characteristic points. The skeleton segment between two adjacent fork points is a common stroke.

In addition to fork points, a corner point on common skeleton can also be a touching point. We detect corner points from each common stroke in two steps: corner detection according to the local maximum of turning angles using the algorithm of Rosenfeld and Johnston [35] and polygonal approximation on each segment between two corner points using the algorithm of Ramer [36]. This scheme has been used in stroke extraction [37]. Both the corner points and the vertices of polygonal approximation are taken as characteristic points. Figure 4 shows an example of common skeleton and characteristic points.

In addition to the fork points and corner points on common skeleton, characters may touch on a smooth stroke without corners, such as the case in Fig. 5. We call this case as smooth touching, which is generally lying on a long common stroke

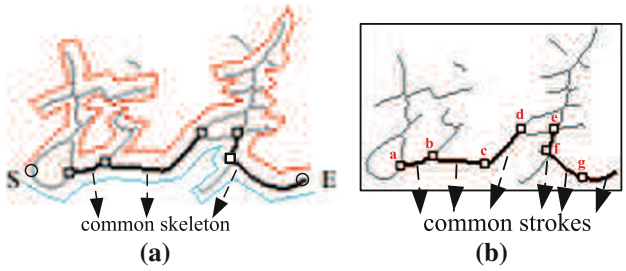


Fig. 4 **a** A skeleton image with common skeleton displayed in *bold* and common fork points marked by *squares*; **b** characteristic points contain five fork points (a, b, d, e, f) and two corner points (c, g)

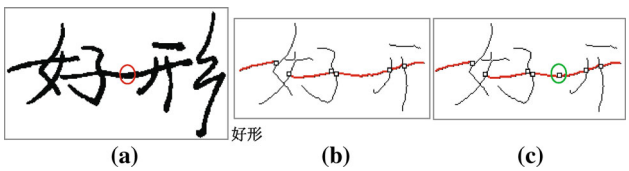
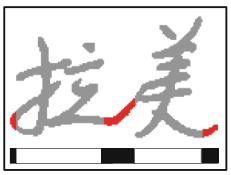


Fig. 5 **a** An example of smooth touching; **b** fork points on the common skeleton; and **c** smooth touching point detected (marked by a *green circle*) (color figure online)

with nearly horizontal direction. We detect smooth touching strokes according to three conditions as below:

1. The horizontal width of the common stroke is greater than $T_{\text{stroke-len}}$, where $T_{\text{stroke-len}}$ is empirically set as one-quarter of the string image height (SI_{height}).
2. Its direction is near horizontal, say, the angle with the x-axis is within $\pm 45°$.
3. It overlaps with at least a single-stroke region in the string image.

We denote the total overlapping width of a common stroke with all the single-stroke regions as *homo-length*. The single-stroke region is defined in [2] as a horizontal region with only one vertical stroke crossing. An example is shown in Fig. 6a. The *homo-length* is also useful for separating line generation as will be described later.



common stroke	*homo-length*
a -> b	0
b -> c	5
c -> d	16
e -> f	0
f -> g	0
g -> E	6

(a) (b)

Fig. 6 **a** Three single-stroke regions indicated by the *filled regions* in the bar below; **b** the *homo-length* values of the common strokes in Fig. 4b

4.2 Candidate separating line generation

Each characteristic point (fork, corner or smooth touching point) indicates a candidate cut for separating touching characters. From a characteristic point, a separating line can be formed by connecting one point from the upper contour and one from the lower contour of the touching pattern image. It is observed in Chinese handwriting that most touching characters can be separated by vertical lines. Thus, we consider only vertical separating lines for simple implementation.

First, the upper contour and lower contour are traced from left to right clockwise and counter-clockwise, respectively. To generate a separating line for each characteristic point, we need to find two suitable terminal points, one on the upper contour and one on the lower contour.

Since a corner point or a smooth touching point lies between fork points and is distant from other strokes, we can simply search a separating line with minimum weighted distance in the neighborhood. The weighted distance is defined as $d = d_1 + 0.4 \times d_2$, where d_1 is the vertical length of the separating line (distance between the paired points in upper and lower contours) and d_2 is the horizontal distance between the separating line and the characteristic point. The neighborhood is empirically confined to be within one-tenth of the string image height, horizontally from the characteristic point.

For a fork point, we need to decide on which common stroke (left or right side) to generate the separating line. The fork point is related to the common skeleton in two possible cases:

1. The fork point is a terminal of one common stroke on either left or right side, e.g., the fork a in Fig. 4b;
2. The fork point is shared by two common strokes on both left and right side, e.g., the fork b in Fig. 4b.

In the first case, it is clear that the separating line should be on the side of the single common stroke. For the second case, we design a simple and effective criterion for selecting a common stroke based on single-stroke region analysis. To do this, we divide the touching pattern image into single-stroke region(s) and multiple-stroke region(s), as shown in Fig. 6a. Then the *homo-length* value of each common stroke is calculated as the total overlapping width between the common stroke and all the single-stroke regions. Figure 6b shows the *homo-length* values of the common strokes in Fig. 4b. The selection criterion is based on the fact that most touching points lie on or near a single-stroke region, according to our investigation on a large database of touching characters (see Sect. 7.1). Thus, we select the common stroke with larger value of *homo-length* to generate the separating line. If the two adjacent common strokes have equal *homo-length*, we generate a separating line on both sides. Again, the separating

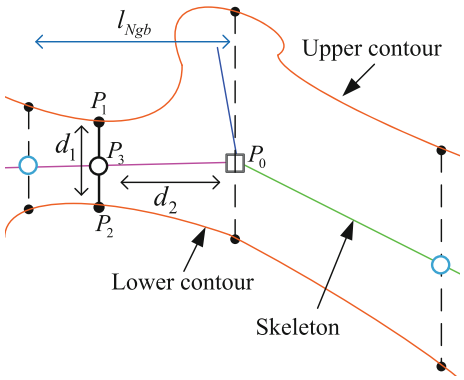


Fig. 7 Illustrative example: separation line ($\overline{P_1P_2}$) generation corresponding to a fork point (P_0), in a neighborhood with horizontal length (l_{Ngb})

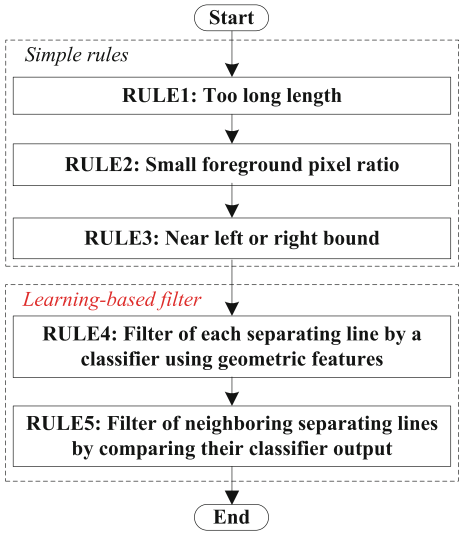


Fig. 8 Flowchart of candidate separating line filtering

line on a common stroke is on a pixel with minimum weighted distance $d = d_1 + 0.4 \times d_2$ from the fork point, in a neighborhood with horizontal length (l_{Ngb}) empirically set as one-tenth of the string image height. Figure 7 shows an illustrative example of separating line generation for a fork point.

5 Verification of candidate separating lines

The generated candidate separating lines (cuts) contain genuine ones that separate characters at touching points, and redundant ones that lie within characters. The objective of cut verification or filtering is to prune redundant cuts while keeping genuine ones as much as possible. The filtering can be seen as a two-class classification problem, which is both imbalanced (there are much more redundant cuts than genuine ones) and cost-sensitive (rejecting a genuine cut is more costly than detecting a redundant one) [20].We perform cut filtering in two steps using simple rules in the first step and a learning-based filter (linear classifier) in the second step. The flowchart of filtering is depicted in Fig. 8. The combination of nonlinear rules and linear classifier provides good trade-off between classification performance and computational complexity.

We use the following notations for characterizing a candidate separating line ($Line_i$), formed by a pair of upper contour point (x_i, y_i^u) and lower contour point (x_i, y_i^l).

- $\pi_{\text{up}}(i)$: index of upper contour point of the ith separating line on the upper contour traced from the left end of touching pattern image;
- $\pi_{\text{lo}}(i)$: index of lower contour point of the ith separating line on the lower contour;
- L_i: length of the ith separating line in number of pixels, i.e., $y_i^l - y_i^u + 1$ for a vertical separating line;
- L_i^f: number of foreground pixels on the ith separating line.

5.1 Filtering by simple rules

We first apply three simple heuristic rules to remove the separating lines that are apparently redundant. Denoting the height of the touching pattern image as H, the rules are described below.

- RULE1: If the length of a separating line is too long, i.e., $L_i > T_1$, it is judged as a redundant one. The threshold T_1 is empirically set as four times of estimated stroke width.
- RULE2: If the foreground pixel ratio of a separating line is small, i.e., $L_i^f / L_i < T_2$, it is judged as a redundant one. T_2 is empirically set as 0.9.
- RULE3: If a separating line is very close to the left or right bound of the touching pattern image along the contour, i.e., $\pi_{\text{up}}(i) < T_3$ and $\pi_{\text{lo}}(i) < T_3$ or $\pi_{\text{up}}(i) > (N_{\text{up}} - T_3)$ and $\pi_{\text{lo}}(i) > (N_{\text{lo}} - T_3)$, it is judged as a redundant one. N_{up} and N_{lo} are the number of pixels in the upper contour and the lower contour, respectively. T_3 is empirically set as one-seventh of H.

The above thresholds are selected such that only those separating lines that are apparently redundant are removed. And the remaining separating lines are to be evaluated by the learning-based filter. Figure 3f shows an example of verified separating lines after filtering by three simple rules.

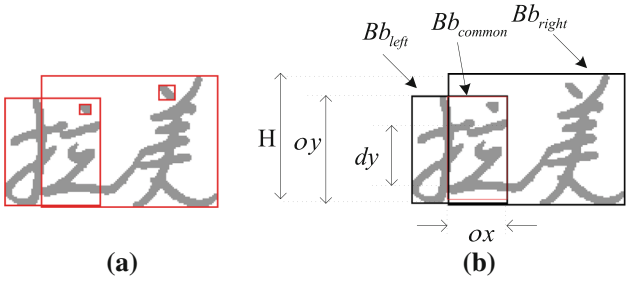


Fig. 9 Illustration of binary properties after separating a touching pattern at the leftmost separating line of Fig. 3f. **a** Extracted CCs; **b** left and right segments after splitting with *bounding boxes* $\mathrm{Bb}_{\mathrm{left}}$ and $\mathrm{Bb}_{\mathrm{right}}$, and their overlapping area with common *bounding box* $\mathrm{Bb}_{\mathrm{common}}$

5.2 Learning-based filtering

The classification performance of learning-based filtering largely depends on the extracted features as well as the training data. While the training data is specified in the experimental section, the geometric features are described below.

5.2.1 Geometric features of candidate separating line

From each candidate separating line, we extract nine geometric features which are independent of the string length (number of characters in the string image). The features can be grouped into two categories depending on whether they characterize an individual separating line (unary features) or the adjacent image segments after splitting at the separating line (binary features). Figure 9 gives an example showing some binary geometric features.

The four unary features ($f_1 - f_4$) and five binary features ($f_5 - f_9$) are listed in Table 1, where the column "Norm" indicates that the feature is normalized by the image height H, the column "Ref" gives the references that the feature is adapted from. We present the feature f_4 based on the observation that a certain type of touching is likely to take place in a stable vertical position. The feature f_7 refers to Fig. 9, where the foreground pixel in the left segment nearest to the center of the separating line is on the upper part of the right bound. When the left and right segments are overlapped, a small dy implies a low likelihood that the candidate separating line should be a genuine one. This feature was presented in [14] for being complementary to the horizontal overlapping width ox. When the left and right segments are not overlapped, we set $ox = 0$ and $\mathrm{d}y = 0$.

5.2.2 Learning-based filter

The verification of candidate separating lines is a two-class classification problem: classify to positive class ω_1 (genuine cut) or negative class ω_0 (redundant cut). Using a learning-based filter has two benefits: optimize the combination of features via supervised learning, flexible balance of the recall rate and precision of cut detection via adjusting only one threshold. We use a linear classifier to combine the features, and specifically, have tested two linear classifiers: linear discriminant function (LDF) [39] and linear support vector machine (SVM) [40]. We have also tested nonlinear SVM but do not observe significant performance gain. On a feature vector $\mathbf{x}$, the output of two-class linear classifier is

$$f(\mathbf{x}) = \mathbf{w}^T\mathbf{x} + b, \tag{1}$$

where $\mathbf{w}$, b are the vector of linear weights and the bias of the classifier, respectively. The weights and bias are estimated in different ways, i.e., assuming equal-covariance Gaussians by LDF and minimizing margin-regularized hinge loss by SVM. For decision, $f(\mathbf{x}) > 0$ indicates that the input sample (candidate cut) is positive, and $f(\mathbf{x}) \leq 0$ indicates that the candidate cut is negative (redundant). The classifier output is often transformed to probability by the sigmoidal function:

$$\mathrm{Prob}(\omega_1|\mathbf{x}) = \frac{1}{1 + \exp\left[-\left(\alpha f(\mathbf{x}) + \beta\right)\right]}, \tag{2}$$

where α and β can be simply set as 1 and 0, respectively. By this transformation, we can select a threshold for the probability from (0, 1).

Based on the classifier output, we can make a decision on an individual candidate cut or on two neighboring cuts. The decision on an individual cut is below.

- RULE4: If $\mathrm{Prob}(\omega_1|\mathbf{x}) > P_{\mathrm{thr}}$, $\mathbf{x} \in \omega_1$; otherwise, $\mathbf{x} \in \omega_0$.

$P_{\mathrm{thr}} \in (0, 1)$ can be adjusted to investigate the trade-off between the recall rate and precision of genuine cut detection. A default threshold is $P_{\mathrm{thr}} = 0.5$. Figure 10b shows an example of removing two redundant separating lines by RULE4.

If two separating lines are close to each other, it is very likely that only one of them is genuine. So, we have a decision based on two neighboring candidate cuts:

- RULE5: If two separating lines are neighboring, the one with lower probability is judged as in ω_0.

The neighboring condition for two adjacent separating lines ($Line_i$ and $Line_j$) is $dc_{ij} < T_{dc}$, where dc_{ij} denotes the contour distance defined as $\max\{|\pi_{\mathrm{up}}(i) - \pi_{\mathrm{up}}(j)|, |\pi_{\mathrm{lo}}(i) - \pi_{\mathrm{lo}}(j)|\}$. T_{dc} is empirically set as one-third of H. Figure 10c shows an example of removing four redundant separating lines by RULE5.

Table 1 Geometric features used in learning-based filtering

No.	Feature description	Norm	Ref.
f_1	Vertical length of separating line	Y	[21]
f_2	Vertical projection value at the x position of cut	Y	[28]
f_3	Vertical crossing count at the x position of cut		[19]
f_4	y Coordinate of the center of separating line	Y	
f_5	Horizontal overlapping width (ox) of two bounding boxes of the left and right segments	Y	[13,30]
f_6	Vertical overlapping distance (oy) of two bounding boxes of the left and right segments	Y	[30]
f_7	Vertical distance (dy) between the center of separating line and the nearest foreground pixel on right bound of $\mathrm{Bb}_{\mathrm{left}}$ when the center is on left bound of $\mathrm{Bb}_{\mathrm{common}}$, or the nearest foreground pixel on left bound of $\mathrm{Bb}_{\mathrm{right}}$ when the center is on right bound of $\mathrm{Bb}_{\mathrm{common}}$	Y	[30]
f_8	Foreground pixel ratio in the common bounding box		[38]
f_9	Square root of the area of the common bounding box	Y	[38]

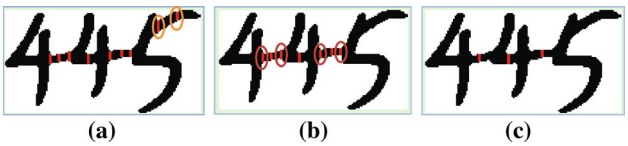


Fig. 10 An example of learning-based filtering. **a** Candidate separating lines before filtering, **b** separating lines after filtering by RULE4 with $P_{\mathrm{thr}} = 0.5$; **c** separating lines after filtering by RULE5

6 Postprocessing

The postprocessing step is adapted from the forced splitting rule of Liu et al. [2]. To make this paper self-contained, we outline here the technique and show an example. Though carefully designed, the above separating line detection and verification techniques may fail to split some touching characters which have unusual touching shape. The forced splitting rule utilizes the segmentation ability of projection profile: a minimum of vertical projection is likely to correspond to a touching point. The method of [2] combines the projection profile weighted with vertical crossing count and the horizontal distance from the center of touching pattern into a characteristic function and locates the x position of minimum characteristic function as a splitting point. The separating line at the point is simply the vertical line at the x position. Figure 11a shows a touching pattern that is failed to split by skeleton-based separating line detection. Figure 11b shows that it is correctly separated by forced splitting in postprocessing.

7 Experimental results

We evaluated the over-segmentation performance of the proposed method and its effect on string recognition on the touching character database CASIA-HWDB-T [25].

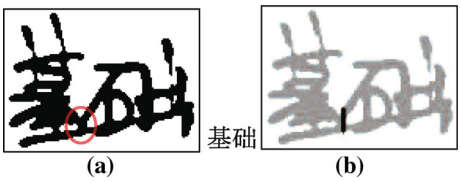


Fig. 11 **a** A touching pattern image (the touching region shown in a *red circle*); **b** a correct separating line generated in postprocessing step (color figure online)

7.1 Databases and comparison methods

The Chinese handwriting database CASIA-HWDB [26] contains three datasets of isolated characters (DB1.0–1.2) and three datasets of handwritten texts (DB2.0–2.2), produced by 1,020 writers. There are 5,091 pages of handwritten texts which contain about 1.35 million character samples. The database CASIA-HWDB-T contains touching character images collected from CASIA-HWDB. One subset in the database, CASIA-HWDB-ST, contains 54,651 single-touching strings, including 48,536 single-touching pairs and 6,115 single-touching strings with more than two characters. The database provides the ground-truth information of the touching point's location, character class, estimated stroke width (SW) and estimated text line height (string height).

We further partition the single-touching subset into three sets for training, validation and testing, respectively. The training set contains 9,904 single-touching pairs from the training set of DB2.1. All the candidate separating lines generated by our method on the training set are used to train the linear classifier. A candidate separating line is treated as a positive sample if the chessboard distance between its center and a labeled touching point is smaller than a threshold ($2 \times$ SW), otherwise a negative sample. The training set has 55,067 candidate separating lines, including 8,665 positive samples and 46,402 negative samples. The validation set contains 1,905 single-touching pairs from the test set of DB2.1. The test set is made up of the remaining 36,727

single-touching pairs and all the single-touching strings with more than two characters. In implementing the learning-based filter, we used the LibSVM package [41] for training the linear SVM classifier.

We investigated the locations of touching points in all single-touching pairs from CASIA-HWDB-ST, and observed that 76 % of touching points lie in one of the single-stroke regions (cf. Fig. 6a). The percentage increases to 91 % if we allow a touching point to lie near one of the single-stroke regions within a small horizontal threshold (SW). This indicates that it is reasonable to use *homo-length* to assist the generation of candidate separating lines in Sect. 4.2.

To evaluate the effect of the proposed over-segmentation method on string recognition, we conducted experiments of string recognition on the single-touching strings in the test set of CASIA-HWDB-ST, using the text line recognition system of [3] only with a character classifier on the primitive segments produced by over-segmentation. We did not use a language model since a touching string normally contains only two or three characters. The second dataset is made up of 1,542 handwritten text lines with nearly 43,000 characters, selected from the test set of DB2.0. Each text line selected has at least one single-touching string. For this dataset, the string recognition system uses both a character classifier and a tri-gram language model on the primitive segments.

We compare the segmentation performance of our proposed method with other three segmentation methods in the literature:

1. The over-segmentation of Liu et al. [2], which has applied successfully to handwritten Japanese mail address recognition and also handwritten Chinese text line recognition;
2. The method of Zhao et al. [21];
3. Our previous method based on heuristics [27];

We implemented the method of [21] as the source codes from the authors are not available. We did not use the fuzzy decision tree classifier for filtering because the filter was designed for touching character pairs while we consider touching strings of variable length.

We implemented the algorithms in C++, and experimented on a personal computer with Intel Core i5-2400 CPU 3.1 GHz, 4 GB of RAM and Windows7 professional 64bit. The algorithms of Liu et al. [2], Zhao et al. [21], our previous method [27] and the current proposed method, spend about 0.1, 4, 2 and 1.6 ms on a single-touching string image of 70 × 128 pixels, respectively.

7.2 Results on character segmentation task

For evaluating the over-segmentation performance without character recognition, we use the chessboard distance between the center of the detected separating line and that of the labeled separating line (touching point) to judge whether the detected separating line is correct or not. We decide a correct separation when the chessboard distance is less than a threshold (2 × SW empirically). If there are multiple detected separating lines within a distance threshold from a labeled touching point, only the nearest one is considered correct. The performance of over-segmentation is measured by the recall rate R and the precision P, defined below:

$$R = \frac{\#\text{correct detected separating lines}}{\#\text{labeled separating lines}} \times 100\,\%, \tag{3}$$

$$P = \frac{\#\text{correct detected separating lines}}{\#\text{detected separating lines}} \times 100\,\%. \tag{4}$$

R is also called as correct segmentation rate in [4,13,14].

Table 2 Over-segmentation performance on the test set of CASIA-HWDB-ST

Algorithm	R (%)	P (%)
Liu et al. [2]	71.7	69.0
Zhao et al. [21]	78.5	31.7
Our previous method [27]	**88.6**	47.4
Our proposed method (SVM)	74.8	**69.6**

Bold values indicate the best results obtained in each column under the current parameter setting

The results of the proposed method and three comparison methods on the test set of CASIA-HWDB-ST are shown in Table 2, where the learning-based filter in the proposed method takes a threshold $P_{\text{thr}} = 0.76$ so as to well balance the recall rate and precision. We can see that the proposed method can outperform the method of [2] in both recall rate and precision. Compared with the method of [21] (without verification by fuzzy decision tree classifier), the proposed method yields a little lower recall rate but much higher precision. Our previous method in [27] outperforms the one of the [21] in both recall rate and precision. To compare the proposed method with the one in [27], we turn to view the precision–recall curve by varying the decision threshold P_{thr} in Fig. 12. We can see that at certain threshold, the proposed method can yield both higher recall rate and higher precision.

Figure 12 shows the precision–recall curves of two classifiers in filtering (LDF and linear SVM) and their variations without simple rules (RULE1–RULE3) or neighboring pruning rule RULE5. It is seen that the two classifiers perform comparably with variable thresholds, and with proper thresholds, both outperform the previous methods of Liu et al. [2], Zhao et al. [21] and Xu et al. [27]. The learning-based filter has a further advantage that the trade-off between recall rate and precision can be flexibly tuned according to the needs of different applications.

We investigated the impacts of two levels of filtering (i.e., simple rules and learning-based filter). On one hand, we plot the performance curve of our proposed method (SVM) with-

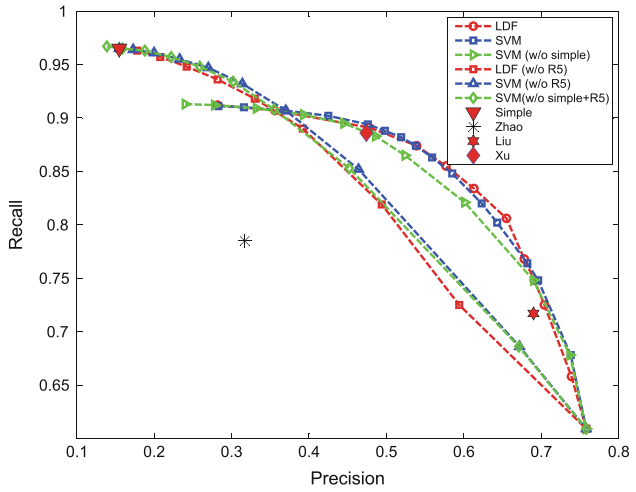


Fig. 12 Precision–recall curves of the proposed over-segmentation method with LDF/SVM filtering, SVM filtering without simple rules (w/o Simple), compared with filtering with only simple rules (Simple), LDF/SVM filtering without RULE5 (w/o R5), SVM filtering without simple rules and RULE5 (w/o Simple+R5), previous methods of Liu et al. [2], Zhao et al. [21] and Xu et al. [27]. The latter three methods and "Simple" have only single points of precision–recall trade-off. The default settings of LDF/SVM filtering use all the rules RULE1–RULE5. All the methods have postprocessing

out filtering by simple rules, denoted as "SVM (w/o Simple)" in Fig. 12. Compared with the proposed method (SVM), it is shown that simple rules help improve the performance slightly, and thus, are complementary to the learning-based filter. On the other hand, we give the performance of our proposed method without the learning-based filter (i.e., filtering with only simple rules), denoted as "Simple" in Fig. 12. It is shown that the three simple rules can only filter out a small portion of obvious redundant cuts while keep most of the genuine ones. In contrast, the learning-based filter can filter out more redundant cuts while sacrifice some genuine ones to improve the precision. In learning-based filtering, it is inevitable to prune some genuine cuts because some of them are very similar to redundant cuts in local shape.

In Fig. 12, we also give the precision–recall curves of LDF/SVM filtering without neighboring pruning rule RULE5 (w/o R5). Compared to the default settings of LDF/SVM filtering with RULE5, it is shown that RULE5 decreases the recall rate abruptly when the precision is low. When increasing the precision, however, RULE5 gives better precision–recall trade-off than filtering without RULE5. This justifies the effectiveness of RULE5 in pruning redundant cuts.

In order to judge the effectiveness of each stage of our proposed method (SVM with $P_{\text{thr}} = 0.76$), we show the intermediate results in Table 3. After the step of candidate cuts generation (before RULE1), most of touching points are detected (high recall rate) with many redundant candidate cuts (low precision). After filtering by simple rules

Table 3 Over-segmentation performance of each filtering stage of our proposed method (SVM with $P_{\text{thr}} = 0.76$) on the test set of CASIA-HWDB-ST

Filtering step	R (%)	P (%)
Before RULE1	**96.5**	13.9
After RULE1	96.4	14.7
After RULE2	96.4	14.8
After RULE3	96.3	15.5
After RULE4	55.6	56.0
After RULE5	53.3	**75.1**

Postprocessing was not used in this experiment
Bold values indicate the best results obtained in each column under the current parameter setting

RULE1, RULE2 and RULE3, some obvious redundant cuts are removed (a small increase in precision), with most genuine ones kept. After SVM filtering (RULE4), most redundant cuts are removed (precision increased 40.5 %) while many genuine ones are also wrongly filtered out (recall rate decreased 40.7 %). The filtering step of RULE5 further improves the precision by 19.1 % with the recall rate decreased only 2.3 %. Finally, by re-generating some cuts in postprocessing step, we obtained the result reported in Table 2, with recall rate increased 21.5 % and precision decreased 5.5 %.

For the learning-based filter, we also evaluated the effects of different geometric features. We adopt a wrapper method [42] with LDF to select feature subsets sequentially (sequential forward search on training data by five-fold cross validation). As a result, the order of features being selected is $\{f_8, f_9, f_6, f_5, f_4, f_7, f_2, f_1, f_3\}$. The precision–recall curves using nine subsets comprising one to nine ordered features are shown in Fig. 13. It is shown that classi-

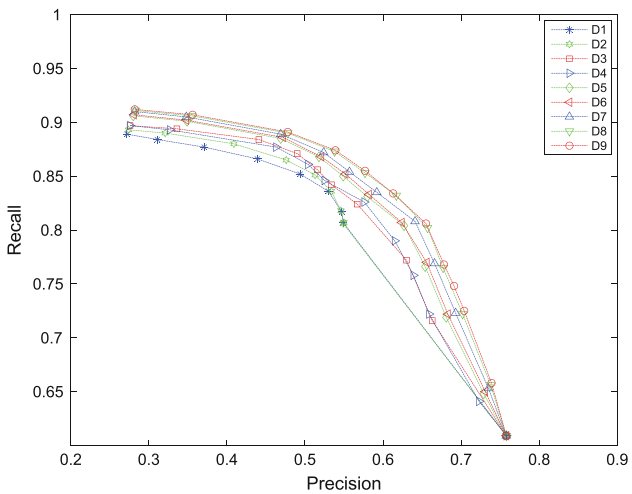


Fig. 13 Precision–recall curves of LDF filtering using ordered feature subsets with one to nine features denoted as D1, D2,…, D9. The curve of D9 is the same as that of LDF in Fig. 12

fication with all the nine features gives the best performance. The feature f_3 is last selected implying least effective but still helps improve the performance slightly. This result indicates that all the nine geometric features are effective for filtering candidate separating lines. The convergence of all the nine curves to a unique point ($R = 60.9\,\%$, $P = 75.8\,\%$) is due to the effect of postprocessing that re-generates many separating lines even if the learning-based filter prunes all the candidate cuts at threshold $P_{\text{thr}} = 1$. Our later results in Sect. 7.3 show that a moderate threshold P_{thr} for higher recall rate than postprocessing only is beneficial for text line recognition.

7.2.1 Segmentation error analysis

Figure 14 shows some examples of separating lines generated by the proposed method (SVM with $P_{\text{thr}} = 0.5$) and three previous methods. We can see that the proposed method can correctly over-segment all the three touching images while the method of Liu et al. [2] fails mainly for the failed matching of empirically defined touching types. This contributes to an increase in recall rate. Compared with the methods of [21, 27], the proposed method produces less redundant separating lines. This contributes to the improvement of precision.

The over-segmentation errors (failure of touching point detection) can be categorized into three types. The first type is the failure of candidate touching pattern detection, i.e., touching pattern is not processed by the splitting procedure because of small width and width-to-height ratio. Figure 12 shows that the highest recall rate is about 92 %, while among the remaining 8 % segmentation errors, about half (4 %) are

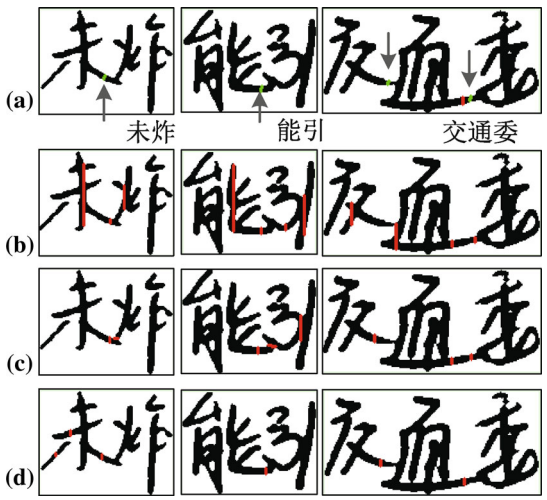


Fig. 14 **a** Labeled touching points (*marked with arrows*) and the detected separating lines (in *red*) by the method of Liu et al. [2]; **b** separating lines detected by the method of Zhao et al. [21]; **c** separating lines detected by the method of Xu et al. [27]; **d** separating lines detected by our proposed method (color figure online)

Fig. 15 Three types of over-segmentation errors (*marked with an arrow*). **a** Failure of touching pattern detection; **b** misclassification by filter; **c** lower probability compared with neighboring redundant cut

due to the failure of touching pattern detection. The second type results from the misclassification of a genuine separating line as redundant one by the filter when $P_{\text{thr}} = 0.5$. The third type is caused by the fault of RULE5 based on comparing the probabilities of neighboring candidate cuts: the probability of a genuine cut may be smaller than a neighboring redundant one. Figure 15 shows examples of three types.

7.3 Results on text line recognition task

Using a text line recognition system with character classifier and linguistic context, we evaluated the effects of over-segmentation in terms of the string recognition performance on the test sets of touching strings in CASIA-HWDB-ST and text line images containing touching characters in CASIA-HWDB2.0. On segmenting the string image into primitive segments, consecutive segments are combined into candidate character patterns subject to the maximum number of segments and constraint of character width. The candidate characters form a candidate segmentation lattice (an example shown in Fig. 16), where each path is evaluated by fusing character classification scores and linguistic model, and the optimal path gives the final result of character segmentation and recognition. The string recognizer uses a modified quadratic discriminant function (MQDF) classifier [43] for character classification and character tri-gram language model, as in [3].

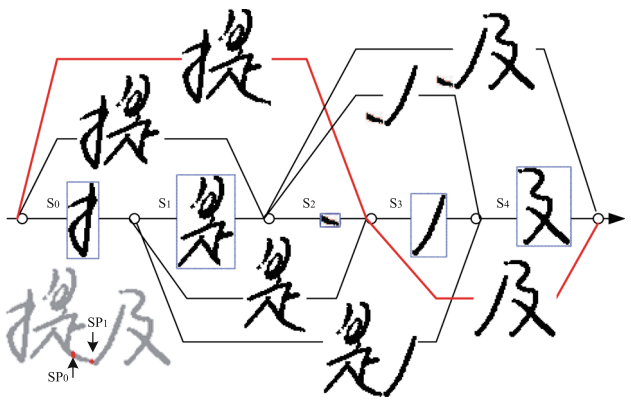


Fig. 16 Example of candidate segmentation lattice for a single-touching pair [25]

Table 4 String recognition performance on the test set of single-touching string database CASIA-HWDB-ST

Algorithm	AR (%)	CR (%)
Liu et al. [2]	52.3	63.0
Zhao et al. [21]	30.1	62.1
Our previous method [27]	37.5	**69.9**
Our proposed method	**54.5**	66.7
Postprocessing only	49.4	57.4

Bold values indicate the best results obtained in each column under the current parameter setting

Table 5 String recognition performance on 1,542 text lines containing touching strings from CASIA-HWDB2.0

Algorithm	AR (%)	CR (%)
Liu et al. [2]	89.4	90.0
Zhao et al. [21]	88.7	90.5
Our previous method [27]	90.1	91.6
Our proposed method	**91.0**	**92.2**
Postprocessing only	88.5	89.2

Bold values indicate the best results obtained in each column under the current parameter setting

The string recognition performance is measured by two character-level criteria: accuracy rate (AR) and correct rate (CR) as in [3,24]. CR represents the percentage of characters correctly recognized while AR further considers the number of incorrect characters inserted due to over-segmentation.

In string recognition, we compared our proposed over-segmentation method with the ones of Liu et al. [2], Zhao et al. [21] and our previous method [27]. Since the post-processing step of Liu et al. [2] alone gives fairly high precision and recall rate as shown in Fig. 13, we also evaluated this step for over-segmentation in string recognition. The string recognition results on single-touching strings are shown in Table 4. It is seen that our proposed method yields the best performance in terms of AR compared with the previous over-segmentation methods. Though our previous method [27] has higher recall rate of touching point detection, the number of over-segmented primitive segments complicates the path evaluation in candidate segmentation lattice and finally deteriorates the string recognition performance, particularly, the low AR indicates high percentage of over-segmentation. The proposed method can flexibly adjust the trade-off between the recall rate and precision of touching point detection, and at certain threshold P_{thr}, it can yield both higher AR and CR in string recognition than the other methods. The previous method of Liu et al. [2] performs fairly well for its balanced recall rate and precision. Using the postprocessing step only for over-segmentation, the obtained string recognition performance is inferior to that of the method of Liu et al. [2], because the low recall rate prevents some characters from being correctly segmented.

The recognition results on 1,542 text lines containing touching strings from CASIA-HWDB2.0 are shown in Table 5. For our proposed method, we set the default threshold $P_{\text{thr}} = 0.5$ which allows splitting of most touching points with a moderate ratio of over-segmentation. In this case of realistic text line recognition, the string recognizer uses linguistic model (character tri-gram) as well as the character classifier. We can see in Table 5 that our proposed method performs best (highest AR and CR) among the comparison methods in string recognition on realistic text line images. Since linguistic context was used on these long text lines (on average, over 20 characters per line), the correct rates AR and CR are much higher than those on short touching strings (Table 4). And since the realistic text lines have a low proportion of touching characters, the difference of overall performance between different over-segmentation methods is small. In this case, using the postprocessing step only for over-segmentation again yields inferior string recognition performance to the method of Liu et al. [2]. This indicates that the trade-off between the recall rate and precision of over-segmentation is important for string recognition.

8 Conclusion

We proposed an effective over-segmentation method with learning-based filtering for splitting Chinese handwritten characters of single-touching, which is the dominant touching type in Chinese handwriting. After detecting touching points and generating candidate separating lines (cuts) with a high recall rate, we prune redundant cuts using some simple rules and a linear classifier using geometric features extracted from candidate cut and the pair of separated image segments. The geometric features are independent of the string length (number of characters) such that the proposed method is applicable to touching strings of variable length. Our experiments on a database of single-touching strings show that the proposed method is able to yield good trade-off between the recall rate and precision of cut detection. The extracted nine geometric features are shown to perform well in filtering with a linear classifier. Experiments of character string recognition on short touching strings and on long text lines containing touching strings show that the proposed over-segmentation method leads to improved string recognition performance. The splitting of the remaining under-segmentation errors on single-touching strings and the separation of multiple-touching strings are to be addressed in our future work.

Acknowledgments This work was supported by the National Natural Science Foundation of China (NSFC) grants 60933010 and 61175021. The authors thank Prof. Horst Bunke, Dr. TongHua Su, Yan-Wei Wang and Xu-Yao Zhang for helpful discussions.

References

1. Casey, R.G., Lecolinet, E.: A survey of methods and strategies in character segmentation. IEEE Trans. Pattern Anal. Mach. Intell. **18**(7), 690–706 (1996)
2. Liu, C.-L., Koga, M., Fujisawa, H.: 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. Wang, Q.-F., Yin, F., Liu, C.-L.: Handwritten Chinese text recognition by integrating multiple contexts. IEEE Trans. Pattern Anal. Mach. Intell. **34**(8), 1469–1481 (2012)
4. Ribas, F.C., Oliveira, L.S., Britto, A.S., Jr., Sabourin, R.: Handwitten digit segmentation: a comparative study. Int. J. Doc. Anal. Recognit. (published online) (2013)
5. Alginahi, Y.M.: A survey on Arabic character segmentation. Int. J. Doc. Anal. Recognit. (published online) (2013)
6. Lee, H., Verman, B.: Binary segmentation algorithm for English cursive handwriting recognition. Pattern Recognit. **45**(4), 1306–1317 (2012)
7. Ikeda, H., Ogawa, Y., Koga, M., Nishimura, H., Sako, H., Fujisawa, H.: A recognition method for touching Japanese handwritten characters. In: Proceedings of 5th International Conference on Document Analysis and Recognition, pp. 641–644 (1999)
8. Han, Z., Liu, C.-P., Yin, X.-C.: A two-stage handwritten character segmentation approach in mail address recognition. In: Proceedings of 8th International Conference on Document Analysis and Recognition, pp. 111–115 (2005)
9. Yu, M.L., Kwok, P.C.K., Leung, C.H., Tse, K.W.: Segmentation and recognition of Chinese bank check amounts. Int. J. Doc. Anal. Recognit. **3**(4), 207–217 (2001)
10. Tseng, L.Y., Chen, R.C.: Segmenting handwritten Chinese characters based on heuristic merging of stroke bounding boxes and dynamic programming. Pattern Recognit. Lett. **19**(10), 963–973 (1998)
11. Tseng, Y.-H., Lee, H.-J.: Recognition-based handwritten Chinese character segmentation using a probabilistic Viterbi algorithm. Pattern Recognit. Lett. **20**(8), 791–806 (1999)
12. Gao, J., Ding, X., Wu, Y.: A segmentation algorithm for handwritten Chinese character strings. In: Proceedings of 5th International Conference on Document Analysis and Recognition, pp. 633–636 (1999)
13. Yamaguchi, T., Yoshikawa, T., Shinogi, T., Tsuruoka, S., Teramoto, M.: A segmentation method for touching Japanese handwritten characters based on connecting condition of line. In: Proceedings of 6th International Conference on Document Analysis and Recognition, pp. 837–841 (2001)
14. Yamaguchi, T., Tsuruoka, S., Yoshikawa, T., Shinogi, T., Makimoto, E., Ogata, H., Shridhar, M.: A segmentation system for touching handwritten Japanese characters. In: Proceedings of 8th International Workshop on Frontiers in Handwriting Recognition, pp. 407–412 (2002)
15. Suwa, M.: Segmentation of touching handwritten Japanese characters using the graph theory method. In: Proceedings of 8th International Conference on Document Recognition and Retrieval, pp. 280–289 (2001)
16. Wang, R., Ding, X., Liu, C.: Handwritten Chinese address segmentation and recognition based on merging strokes. Qinghua Daxue Xuebao/J. Tsinghua Univ. **44**(4), 498–502 (2004) (in Chinese)
17. Li, N.-X., Gao, X., Jin, L.-W.: Curved segmentation path generation for unconstrained handwritten Chinese text lines. In: Proceedings of the IEEE Asia Pacific Conference on Circuits and Systems, pp. 501–505 (2008)
18. Bunke, H.: Recognition of cursive Roman handwriting-past, present and future. In: Proceedings of 7th International Conference on Document Analysis and Recognition, pp. 448–459 (2003)
19. Bayer, T., Kressel, U.: Cut classification for segmentation. In: Proceedings of 2nd International Conference on Document Analysis and Recognition, pp. 565–568 (1993)
20. Vellasques, E., Oliveira, L.S., Britto Jr, A.S., Koerich, A.L., Sabourin, R.: Filtering segmentation cuts for digit string recognition. Pattern Recognit. **41**(10), 3044–3053 (2008)
21. Zhao, S., Chi, Z., Shi, P., Yan, H.: Two-stage segmentation of unconstrained handwritten Chinese characters. Pattern Recognit. **36**(1), 145–156 (2003)
22. Suen, C.Y., Mori, S., Kim, S.-H., Leung, C.H.: Analysis and recognition of Asian scripts-the state of the art. In: Proceedings of 7th International Conference on Document Analysis and Recognition, pp. 866–878 (2003)
23. Srihari, S., Yang, X., Ball, G.: Offline Chinese handwriting recognition: an assessment of current technology. Frontiers Comput. Sci. China **1**(2), 137–155 (2007)
24. Su, T., Zhang, T., Guan, D., Huang, H.: Off-line recognition of realistic Chinese handwriting using segmentation-free strategy. Pattern Recognit. **42**(1), 167–182 (2009)
25. Xu, L., Yin, F., Wang, Q.-F., Liu, C.-L.: A touching character database from Chinese handwriting for assessing segmentation algorithms. In: Proceedings of 12th International Conference on Frontiers in Handwriting Recognition, pp. 89–94 (2012)
26. Liu, C.-L., Yin, F., Wang, D.-H., Wang, Q.-F.: CASIA online and offline Chinese handwriting databases. In: Proceedings of 11th International Conference on Document Analysis and Recognition, pp. 37–41 (2011)
27. Xu, L., Yin, F., Wang, Q.-F., Liu, C.-L.: Touching character separation in Chinese handwriting using visibility-based foreground analysis. In: Proceedings of 11th International Conference on Document Analysis and Recognition, pp. 859–863 (2011)
28. Liang, Z., Shi, P.: A metasynthetic approach for segmenting handwritten Chinese character strings. Pattern Recognit. Lett. **26**(10), 1498–1511 (2005)
29. Strathy, N.W., Suen, C.Y., Kryzyzak, A.: Segmentation of handwritten digits using contour features. In: Proceedings of 2nd International Conference on Document Analysis and Recognition, pp. 577–580 (1993)
30. Ha, T.M., Zimmermann, M., Bunke, H.: Off-line handwritten numeral string recognition by combining segmentation-based and segmentation-free methods. Pattern Recognit. **31**(3), 257–272 (1998)
31. Chen, Y.-K., Wang, J.-F.: 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)
32. Oliveira, L.S., Lethelier, E., Bortolozzi, F., Sabourin, R.: A new segmentation approach for handwritten digits. In: Proceedings of 15th International Conference on, Pattern Recognition, pp. 2323–2326 (2000)
33. Sadri, J., Suen, C.Y., Bui, T.D.: Automatic segmentation of unconstrained handwritten numeral strings. In: Proceedings of 9th International Workshop on Frontiers in Handwriting Recognition, pp. 317–322 (2004)
34. Suzuki, S., Abe, K.: Binary picture thinning by an iterative parallel two-subcycle operation. Pattern Recognit. **10**(3), 297–307 (1987)
35. Rosenfeld, A., Johnston, E.: Angle detection on digital curves. IEEE Trans. Comput. **22**, 875–878 (1976)
36. Ramer, U.: An iterative procedure for the polygonal approximation of plane closed curves. Comput. Graph. Image Process **1**, 244–256 (1972)

37. Liu, C.-L., Kim, I.-J., Kim, J.H.: Model-based stroke extraction and matching for handwritten Chinese character recognition. Pattern Recognit. **34**(12), 2339–2352 (2001)
38. Yin, F., Wang, Q.-F., Liu, C.-L.: Integrating geometric context for text alignment of handwritten Chinese documents. In: Proceedings of 11th International Conference on Frontiers in Handwriting Recognition, pp. 7–12 (2010)
39. Duda, R.O., Hart, P.E., Stork, D.G.: Pattern Classification, chap. 2. Wiley, New York (2001)
40. Vapnik, V.N.: The Nature of Statistical Learning Theory. Springer, New York (1995)
41. Chang, C.-C., Lin, C.-J.: LIBSVM: a library for support vector machines. ACM Trans. Intell. Syst. Technol. **2**(3), 1–27 (2011). Software available at http://www.csie.ntu.edu.tw/cjlin/libsvm
42. Guyon, I., Elisseeff, A.: An introduction to variable and feature selection. J. Mach. Learn. Res. **3**, 1157–1182 (2003)
43. Kimura, F., Takashina, K., Tsuruoka, S., Miyake, Y.: Modified quadratic discriminant functions and the application to Chinese character recognition. IEEE Trans. Pattern Anal. Mach. Intell. **9**(1), 149–153 (1987)