A Touching Character Database from Chinese Handwriting for Assessing Segmentation Algorithms

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
{*lxu,fyin,wangqf,liucl*}*@nlpr.ia.ac.cn*

Abstract

For assessing touching character segmentation algorithms, we present a database of touching characters collected from the Chinese handwriting database CASIA-HWDB, called CASIA-HWDB-T. It includes 56,469 two-character or multiple-character touching strings, among which 1,818 strings have multiple-touching characters. We also partition the touching strings into 50,157 all-Chinese strings, 2,788 all-digit ones, 328 all-letter ones, and 3,196 mixed-character ones. All the strings are annotated with the character classes, locations of touching points, and auxiliary values like string height and average stroke width. And last, we measure the segmentation performance of three existing algorithms on this database for reference.

1. Introduction

Handwritten Chinese text line recognition is receiving increasing attention in recent years [9, 11], partially owing to the availability of public annotated databases (HIT-MW [8] and CASIA-HWDB [6]). The performance of text line recognition depends on the algorithms of character segmentation, classification, and context modeling. Character segmentation is widely acknowledged to be a challenging problem and still remains un-solved.

Character segmentation methods can be generally grouped into dissection-based methods and recognition-based ones [1]. Dissection-based methods have obvious deficiencies in handwriting because handwritten characters have variable size, location, intra- and between-character gap, and there are many touching characters. Recognition-based methods generate candidate characters and verify using the classifier and contexts. They can be further categorized into implicit segmentation and explicit segmentation. The latter is also called heuristic over-segmentation, and has been prevalently adopted in Chinese/Japanese handwriting recognition.

Several explicit segmentation algorithms have been proposed in the last decade. According to the features used, these algorithms can be generally categorized into three types: foreground-based methods [2, 3, 10, 14, 5, 12, 13], background-based methods, and combined foreground-background methods [15, 4]. These methods usually have high segmentation accuracy when evaluated directly on the text line image. This is because that a text line image usually contains a large ratio of isolated and broken characters, while the number of touching characters is relatively small. The high accuracy may obscure the capacity difference of touching character segmentation, which is currently one of the core problems for segmentation.

Even though some authors have reported the results directly on touching strings, it is difficult to assess the segmentation algorithms because of the unavailability of a public touching string database. In the Chinese handwriting segmentation, different authors use different databases with different sizes and difficulty. And so the performance difference may be not very convincing. To the best of our knowledge, all the touching strings collected by themselves are not public. Thus, it is necessary to provide an annotated touching string database to assess the segmentation algorithms.

Our building of Chinese touching character database was motivated by the work of L.S. Oliveira et al [7] on providing a touching handwritten numeral string database, where the touching numeral pairs were synthesized from single handwritten digits and the touching point was assumed to be the synthesized place. Four state-of-the-art segmentation algorithms were assessed on this database. Both the correct segmentation rate

978-0-7695-4774-9/12 $26.00 © 2012 IEEE
DOI 10.1109/ICFHR.2012.173

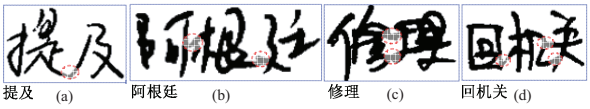


Figure 1. Examples of (a) single-touching pair, (b) single-touching string with more than two characters, (c) multiple-touching pair, and (d) touching string with more than two characters including at least a multiple-touching pair.

and computation cost for the postponed numeral string recognition were considered.

However, there are still some differences between the synthesized sample and the real sample in daily life. We decide to collect all the touching strings directly from the database CASIA-HWDB. These real touching samples may be better to reflect the practical situation than the synthesized samples.

In total, we collected 56,469 touching samples which contain two or more characters to form our touching string database, called CASIA-HWDB-T. Specifically, it includes 48,536 single-touching pairs, 6,115 single-touching strings with more than two characters and 1,818 multiple-touching pairs. Some examples are shown in Figure 1. In order to consider different specific usages, we also partition the touching strings into 50,157 all-Chinese strings, 2,788 all-digit ones, 328 all-letter ones, and 3,196 mixed-character ones. All the touching samples are annotated with the character classes, locations of touching points, and auxiliary values like string height and average stroke width. This touching string database will be public at our database webpage [1].

2. Database

We firstly introduce the scheme for annotation of this touching string database. Then we present the statistics of this database.

2.1 Annotation of database

Before the introduction to the annotation of our touching string database, we briefly describe the source handwriting database CASIA-HWDB [6]. It has been collected and annotated by the authors' group recently. It contains about 5,090 paragraphs of handwritten texts and 1.35 million segmented character samples. It is involved with over 2,600 commonly used character classes (e.g. Chinese, digit, English letter, punctuation mark). Each segmented character sample is stored as gray-scale image in a DGR file, with records such as its class label, position, the bitmap, and so on. The gray-scale image has its background pixels uniformly set as 255 and can be easily converted to the binary image. Each DGR file is made up of all the character samples in a handwritten text page, with the sequential order of text lines. Figure 2 shows an example of a text line image with its character-level annotation in this source database.

With the help of character-level annotation in the source database, we can extract all the touching characters automatically using a simple scheme as follows. We check the foreground pixels of two consecutive character parts in the annotated text line image; If there exists any foreground pixel touching, then these two characters belong to a touching string. Meanwhile, according to the eight-connected checking of touching foreground pixels, we can get the number and location of touching points between these two character samples. All the detected touching strings together with their touching points' location information are saved in a ground truth file.

According to the number of characters, a touching string can be generally classified as a touching pair and a touching string with more than two characters. According to the number of touching points between two characters, a touching pair can be classified as single-touching and multiple-touching pair. Figure 1 shows some examples of four types of touching strings. For single-touching pair with all Chinese characters, Liu et al [5] further defines five different types as shown in Figure 3. And we have analyzed the ratio of five types on randomly sampled 1,000 single-touching Chinese pairs and found that all these five types occupies about ninety-eight percent of all the examined samples.

We have investigated all the above detected touching strings and found that most of them belong to the type of single-touching pair. And most of remaining strings belong to other two types of single-touching string with more than two characters and multiple-touching pair. Thus we decide to build our touching string database (CASIA-HWDB-T) by the above three representative types. Figure 4 presents some examples of each type available in our database.

In the following, we will introduce the ground truth file's format for each touching string. As aforementioned, we have already get the information about the character class label and touching points' locations of the string automatically. Moreover, in order to facilitate the comparison of different segmentation algorithms,

[1] http://www.nlpr.ia.ac.cn/databases/handwriting/Home.html

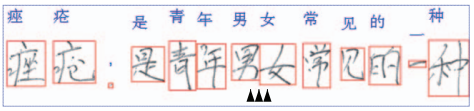


Figure 2. Example of a ground-truthed text line image in CASIA-HWDB (triangles point to a touching pair).

Type	Touching Stroke Relation	Rate (%)	Examples
1	⊢ ⊣	27.3	
2	⌄ ⌃	34.4	
3	⋌ ⋋	20.4	
4	— —	12.9	
5	✕	3.2	

Figure 3. Touching types of single-touching Chinese character pair [5].

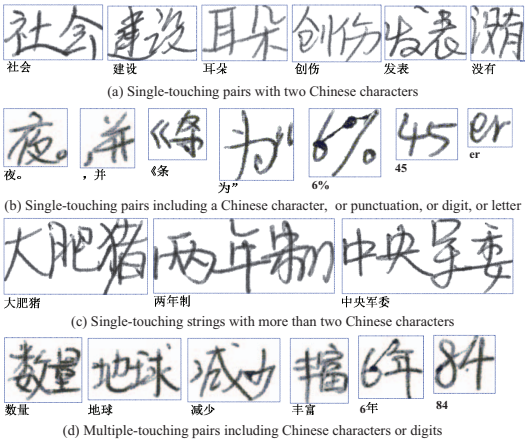


Figure 4. Examples of touching string samples extracted from the database (with character class labels at the left bottom of each image).

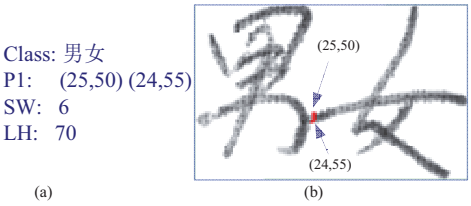


Figure 5. Example of (a) the ground-truth information, (b) the touching point (in red) and its two terminals for a single-touching pair from Figure 2.

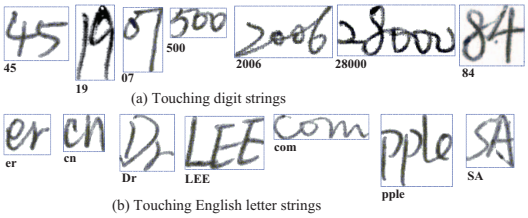
we save the value of two parameters estimated from the whole text line image: string height (LH) and average stroke width (SW). Both parameters are often used in a segmentation algorithm to normalize different writing styles and pen thickness. As a result, the ground truth file of a touching string includes character labels, location of touching points, and auxiliary values (LH and SW). Figure 5 gives an example of the ground truth file for a single-touching pair.

We manually remove all the mislabeled touching strings by a checking tool, since some annotations in the source database of text lines may be incorrect. Also, some touching strings with a character label out of the pre-defined character set are removed. All the remaining touching strings are used to form our database (CASIA-HWDB-T), which includes three subsets: single-touching pair (HWDB-ST-P), single-touching string with more than two characters (HWDB-ST-M), and multiple-touching pair (HWDB-MT).

2.2 Statistics of database

We investigate the number of touching strings, touching points and characters in this database plus its above three subsets, as depicted in Table 1. We can see that this database contains a large number of touching strings sufficient for segmentation experiment. On average, each string have 1.2 touching points and 2.1 characters. It is in accordance with our observation that most touching strings belong to the type of single-touching pair. For HWDB-ST-M, the number of characters is about three times than touching strings. So most of single-touching strings with more than two characters actually have only three characters. For HWDB-MT, the number of touching points is about two times than touching strings. Thus, most of multiple-touching pairs actually have only two touching points (i.e. double-touching pair).

We also partition the touching strings into 50,157 all-Chinese strings, 2,788 all-digit ones, 328 all-letter ones, and 3,196 mixed-character ones, as depicted in Table 2. All-Chinese strings can be met in Chinese mail address recognition and business form processing. And all-digit



(a) Touching digit strings

(b) Touching English letter strings

Figure 6. Examples of touching string samples including all digits or English letters extracted in the database.

strings and all-letter ones can be met in bank check recognition. And it is suitable to design individual segmentation algorithms on these all-Chinese strings and all-digits ones, for their large numbers. Figure 6 shows some examples of all-digits and all-letters strings, while Figure 4a and Figure 4b show some examples of all-Chinese ones and mixed-character ones, respectively.

Table 1. Statistics of CASIA-HWDB-T according to the subsets of single-touching pair, single-touching string with more than two characters and multiple-touching pair.

Dataset	#String	#Touch Point	#Character
HWDB-ST-P	48,536	48,536	97,072
HWDB-ST-M	6,115	13,367	19,482
HWDB-MT	1,818	3,756	3,636
Total	56,469	65,659	120,190

Table 2. Statistics of CASIA-HWDB-T according to Chinese, alphanumerical character's partition.

Dataset	#String	#Touch Point	#Character
allChinese	50,157	57,749	106,090
allDigits	2,788	3,601	6,320
allLetters	328	440	765
other	3,196	3,869	7,015

3. Segmentation algorithms

We have measured the performance of three existing over-segmentation algorithms on this database for reference.

(1) The first algorithm is proposed by Liu et al [5]. It is based on local contour analysis to generate candidate separating points according to five touching types as in Figure 3. Then a complementary operation relying on vertical projection and stroke crossing number is invoked to produce some extra separating points. This algorithm has been successfully applied to handwritten Japanese address recognition.

(2) The second algorithm, proposed by Xu et al [12], takes into account the information of the whole contour, of which upper and lower part are matched by Dynamic Time Warping (DTW). All the contour corner points together with their matched contour points are used to form the candidate separating lines. Then four heuristic rules are applied to remove some redundant separating lines.

(3) Differently from the previous two algorithms, the third one takes into account the foreground skeleton analysis, together with the contour analysis. It is proposed by Xu et al [13] recently. Candidate separating points are detected on the skeleton. Then candidate separating lines are formed according to contour information. Four rules as in the second algorithm, plus another heuristic utilizing the profile visibility metric of the separating line, are used to remove some redundant separating lines.

4. Experiments

For each over-segmentation algorithm, we should at first identify the touching pattern in the string image. After extracting connected components and merging highly overlapped ones, we treat components with large width (greater than 0.8 $\times LH$) or width-to-height ratio (greater than 1.3) as candidate touching patterns. Then we apply an over-segmentation algorithm to generate candidate separating points (or lines), where we cut to form a list of primitive segments. Algorithm #1, #2 and #3, implemented in C++, take about 0.3, 3 and 4.6 ms to run on a 83 $\times$ 123 resolution image (single-touching character pair) with a personal computer (Intel Core2 CPU 3GHz), respectively.

We evaluate the previous over-segmentation algorithms directly according to the distance (d) between a candidate separating point and a touching point. And we consider a correct segmentation when d is less than a threshold d_{th}, which is set as $2 \times SW$ empirically. The overall performance of over-segmentation is measured by the recall rate R and the precision rate P, as following.

$$R = \frac{\#\text{ of correct separating points}}{\#\text{ of touching points}} \times 100\%$$
$$P = \frac{\#\text{ of correct separating points}}{\#\text{ of candidate separating points}} \times 100\%$$

R is called correct segmentation rate and used to assess the segmentation algorithms for touching numerals in [7].

The result on the whole touching string database is shown in Table 3, while the results on each subset is given in Table 4. If we only consider the recall rate R as [7], then algorithm #2 performs best. And algorithm #3 has similar result. For the results on the subset, we can see that the performance on the multiple-touching pairs is much lower than other two subsets of single-touching strings. On contrary, algorithm #1 performs best on the multiple-touching case. It is because that algorithm #2 and #3 are mainly designed for single-touching case.

Table 3. Performance comparison on the whole database HWDB-DB-T.

Algorithm	R (%)	P (%)
1	67.3	68.5
2	83.1	19.4
3	80.9	46.7

Table 4. Performance comparison (%) on the three subsets: HWDB-ST-P (ST-P), HWDB-ST-M (ST-M) and HWDB-MT (MT).

	ST-P		ST-M		MT	
Algorithm	R	P	R	P	R	P
1	68.8	67.7	67.2	70.9	48.6	73.4
2	85.5	18.3	91.4	25.6	22.0	14.3
3	82.6	44.9	86.4	54.2	39.5	47.2

However, besides high recall rate R, it is necessary to have at least a moderate precision rate P. A lot of candidate separating points can generate a lot of primitive segments, which may form a lot of hypothesis patterns. And all the patterns should be evaluated by a classifier, leading to expensive computation cost. Moreover, a lot of hypothesis patterns can cause many confusing recognition results, with the currently unperfect classifier for the large character set. Figure 7 depicts an example of an image segmented by algorithm #3 and its corresponding segmentation graph. We can see that the algorithm generates two candidate separating points (SP_0 and SP_1) and five primitive segments (S_0 to S_4),

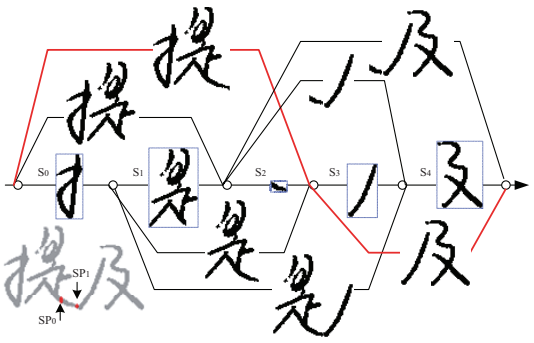


Figure 7. Example of a segmentation graph for a single-touching pair.

three of which (S_1, S_2 and S_3) are produced by two separating points. Assuming that a hypothesis pattern should contain at most three consecutive primitive segments, we form totally twelve hypothesis patterns and must invoke the classifier twelve times to get a score for each pattern. And a path with the maximal score is found in the graph to represent the final segmentation and recognition result.

Table 5 reports the average number of primitive segments generated by the three over-segmentation algorithms and the average number of classifier calls on this database. We can see that the number of classifier calls is polynomial ($O(n^2)$) to the number of primitive segments.

In order to evaluate the over-segmentation algorithms indirectly, we apply the string recognition system only with character classifier as in [11]. And we measure the recognition performance by accuracy rate (AR) and correct rate (CR). CR represents the percentage of characters correctly recognized while AR further considers the number of incorrect characters inserted and may have minus value. Table 6 and Table 7 show the string recognition result on the database and its three subsets, respectively. We can see that algorithm #2 has the worst performance of accuracy rate AR although it has the highest recall rate R. It is because that algorithm #2 has very low precision rate P, leading to a lot of confusing patterns to the classifier and many insertion errors (i.e. low AR). In the text line recognition, the string recognition performance can be improved significantly by incorporating contexts (e.g. language model) to reduce confusions and insertion errors [11].

In summary, we argue that the objective of over-segmentation is to have a high recall rate R, with at least a moderate precision rate P, which means an acceptable computation cost and a small number of recognition confusions. In this view, algorithm #3 would be

the best choice while algorithm #1 is also a good one.

Table 5. The average number of primitive segments generated by the over-segmentation algorithms and corresponding classifier calls.

	#Primitive segments		#Classifier calls	
Algorithm	Mean	Std. Dev.	Mean	Std. Dev.
1	2.6	1.0	5.2	3.6
2	4.8	2.0	15.3	10.3
3	3.5	1.5	9.0	6.8

Table 6. String recognition performance comparison on the whole database.

Algorithm	$AR\,(\%)$	$CR\,(\%)$
1	50.1	61.0
2	6.4	61.8
3	35.3	66.2

Table 7. String recognition performance comparison (%) on the above three subsets.

	ST-P		ST-M		MT	
Algorithm	AR	CR	AR	CR	AR	CR
1	51.3	63.0	46.8	54.2	35.8	45.0
2	5.9	62.8	13.8	65.1	-19.0	17.1
3	35.2	67.2	39.2	67.2	18.8	35.9

5. Conclusion and future works

We briefly introduce a touching string database from Chinese handwriting. Our aim is to facilitate the comparison of various segmentation algorithms. Meanwhile, the database will be public on the web site, hopefully helping the research on the segmentation of touching strings. And ultimately we hope this database can improve the performance of handwritten Chinese text line recognition. In the future, we will add the experimental results of other typical segmentation algorithms in Chinese handwriting. Also, we will do further statistical analysis about the touching points on this database.

Acknowledgment

This work was supported by the National Basic Research Program of China (973 Program) Grant 2012CB316302, the Strategic Priority Research Program of the CAS (Grant XDA06030300), the National Natural Science Foundation of China (NSFC) Grants 60825301 and 60933010. The authors thank Dr. Tonghua Su and Dahan Wang for helpful discussions.

References

[1] R. Casey and E. Lecolinet. A survey of methods and strategies in character segmentation. *IEEE Trans. Pattern Anal. Mach. Intell.*, 18(7):690–706, 1996.

[2] J. Gao, X. Ding, and Y. Wu. A segmentation algorithm for handwritten chinese character strings. In *Proc. 5th ICDAR*, pages 633–636, 1999.

[3] H. Ikeda, Y. Ogawa, M. Koga, H. Nishimura, H. Sako, and H. Fujisawa. A recognition method for touching japanese handwritten characters. In *Proc. 5th ICDAR*, pages 641–644, 1999.

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

[5] C.-L. Liu, M. Koga, and 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.

[6] C.-L. Liu, F. Yin, D.-H. Wang, and Q.-F. Wang. Casia online and offline chinese handwriting databases. In *Proc. 11th ICDAR*, pages 37–41, 2011.

[7] L. S. Oliveira, A. de Souza Britto Jr., and R. Sabourin. A synthetic database to assess segmentation algorithms. In *Proc. 8th ICDAR*, pages 207–211, 2005.

[8] T. Su, T. Zhang, and D. Guan. Corpus-based hit-mw database for offline recognition of general-purpose chinese handwritten text. *IJDAR*, 10(1):27–38, 2007.

[9] T. Su, T. Zhang, D. Guan, and H. Huang. Off-line recognition of realistic chinese handwriting using segmentation-free strategy. *Pattern Recognition*, 42(1):167–182, 2009.

[10] M. Suwa. Segmentation of touching handwritten japanese characters using the graph theory method. In *Proc. 8th DRR*, pages 280–289, 2001.

[11] Q.-F. Wang, F. Yin, and C.-L. Liu. Handwritten chinese text recognition by integrating multiple contexts. *IEEE Trans. Pattern Anal. Mach. Intell.*, 2012, in press.

[12] L. Xu, F. Yin, and C.-L. Liu. Touching character splitting of chinese handwriting using contour analysis and dtw. In *Proc. 2010 Chinese Conference on Pattern Recognition(CCPR)*, pages 814–818, 2010.

[13] L. Xu, F. Yin, Q.-F. Wang, and C.-L. Liu. Touching character separation in chinese handwriting using visibility-based foreground analysis. In *Proc. 11th ICDAR*, pages 859–863, 2011.

[14] T. Yamaguchi, S. Tsuruoka, T. Yoshikawa, T. Shinogi, E. Makimoto, H. Ogata, and M. Shridhar. A segmentation system for touching handwritten japanese characters. In *Proc. 8th IWFHR*, pages 407–412, 2002.

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