北京大学学报(自然科学版)
Acta Scientiarum Naturalium Universitatis Pekinensis
doi: 10.13209/j.0479-8023.2016.025

A Benchmark For Stroke Extraction of Chinese Characters

CHEN Xudong, LIAN Zhouhui†, tang Yingmin, XIAO Jianguo

Institute of Computer Science and Technology, Peking University, Beijing 100871; † Correspondence author, E-mail: lianzhouhui@pku.edu.cn

Abstract This paper presents a benchmark, which includes a manually-constructed database and evaluation tools, to address this problem. Specifically, the database contains a number of images of Chinese characters represented in four commonly-used font styles and corresponding stroke images manually segmented from character images. Performance of a given stroke extraction method can be evaluated by calculating dissimilarities of the automatic segmentation results and the ground truth using two specially-designed metrics. Moreover, we also propose a new method based on Delaunay triangulation to effectively extract strokes from Chinese characters. Experimental results obtained by comparing three algorithms demonstrate that the benchmark works well for the evaluation of stroke extraction approaches and the proposed method performs considerably well in the application of stroke extraction for Chinese characters.

Key words benchmark; stroke extraction; Chinese characters

一种汉字笔画自动提取基准测试库

陈旭东　连宙辉†　唐英敏　肖建国

北京大学计算机科学技术研究所，北京 100871; † 通信作者, E-mail: lianzhouhui@pku.edu.cn

摘要 构建了一个提供评测工具的笔画基准测试库，其中包含一个人工搭建的笔画数据库，该数据库拥有 4 种字体的汉字图像以及对应的人工提取的笔画信息。通过比较算法自动提取的笔画结果和数据库中的标准笔画之间的差异，测试库可以评测出笔画自动提取算法的性能。还提出一种新的基于 Delaunay 三角剖分的方法，可以有效地从汉字图像中提取出笔画信息。在测试库中对现有的三种笔画提取方法进行比较，实验数据表明，所提出的笔画基准测试库能够对笔画提取算法给出有效的评测，并且新的算法在汉字笔画提取的性能中效率较高。

关键词 基准测试库；笔画提取；汉字
中图分类号 TP399

1 Introduction

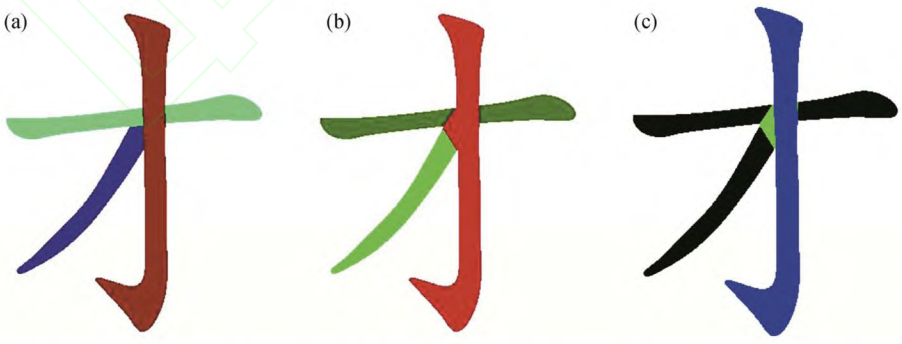
As the oldest characters in the world, Chinese characters are so widely used that we can see them everywhere in our daily lives. In this new digital information age, the demand for digital use of Chinese characters has become even larger. Many computer applications related to Chinese characters emerge at the right moment, for instance of text recognition, auto generation of Chinese font, digital ink, and so on. Stroke extraction plays an important role in these applications, but the problem of stroke extraction has not been well solved so far and how to acquire the accurate stroke extraction results has become a

国家自然科学基金(61202230, 61472015)、863 计划(2014AA015102)和北京市自然科学基金(4152022)资助
收稿日期: 2105–06–19; 修回日期: 2105–09–14; 网络出版日期:

bottleneck. As the key step of these techniques, stroke extraction has become a central issue and many researchers have focused on the stroke extraction algorithms.

Various algorithms have been developed to solve the problem of extracting strokes of Chinese characters, and they can be classified into two categories: unsupervised methods[1–4] and data-driven or model-based methods[5–7]. However, up to now, no stroke extraction benchmark has been developed in the literature, and thus it is hard to quantitatively compare the performance of different algorithms. In fact, most stroke extraction algorithms are evaluated on their own datasets with different styles of Chinese characters. As different font libraries have different shapes of the same stroke, results may change greatly when they are evaluated on different datasets. For instance, stroke extractions evaluated on Kaiti font library may be much better than on Songti font library. There have been two main problems. One problem is the lack of standard stroke database, without which "ground truth" lacks definition. Another problem is that no agreed-upon metrics have been proposed and algorithms suffer from how to effectively evaluate their results.

In this paper, we build a benchmark for quantitative evaluation of stroke extraction algorithms. In order to acquire standard strokes of Chinese characters, we designed an interactive tool to manually extract strokes with great efficiency. Different from 3D mesh segmentation[8], Chinese characters have defined strokes in sequence. We use a GB2312 Kaiti font library as the reference information for it is the most popular among all font libraries, which has independent stroke information with corresponding order. Thus, strokes of various font libraries could be correctly extracted in corresponding order guided by the Kaiti stroke information. We treat those manually extracted strokes as the ground truth results for comparison. Besides, it is fair and convincing for algorithms based on data-driven methods to use the Kaiti font library as the unique reference information to generate stroke extractions in the benchmark. We experiment on three stroke extraction algorithms to investigate the practical use of the benchmark, including naive K-NN method, Lian's method[5] together with our method inspired by the work in Ref. [4]. Based on a standard stroke database containing characters of 4 font libraries (FS, HYKT, HT, LS), we compute metrics that measure how well the strokes generated by algorithms match those manually extracted ones (Fig. 1). Benefited from the metrics, algorithms will be given quantitative evaluations and comparisons with each other, and thus



(a) shows the standard stroke set of the Chinese character *Cai* in HYKT font style; (b) gives stroke extraction results using our method introduced in Section 3; (c) compares the standard *vertical hook* stroke in (a) and the computer-generated one in (b) by measuring their similarity (region in blue color represents correctly extracted part and region in green color identifies the wrongly extracted part.)

Fig. 1 Evaluating stroke extractions in the benchmark

they can improve themselves for better performance in the future.

2 Benchmark Design

Our main contribution in this paper is the design of a benchmark for stroke extraction of Chinese characters. In this section, we focus on how to build such a benchmark. Several problems have to be solved, including “how to acquire standard strokes?”, “which font libraries and how many characters should a standard stroke dataset contain?” and “how to effectively measure the similarity between computer-generated strokes and standard strokes?”. We give discussions of those problems in the following subsections.

2.1 An Interactive stroke extraction tool

As mentioned above, each computer-generated stroke is evaluated by comparing it to a standard stroke. Here comes the problem of how to acquire such standard strokes. Given a binary image of a Chinese character, since each stroke has its definite shape that has already been designed by artists, we can manually extract them if we know how to write the character using an image editing tool such as photoshop. However, the tool we choose to use for stroke extraction should be easy to learn and be fast enough to use (a couple of seconds per stroke), otherwise it will take years to finish extracting thousands of strokes using inconvenient tools like photoshop. It should also be capable to guide people which stroke to extract next in case the user does not know the correct order of the strokes.

To our best knowledge, no database of standard strokes has been constructed so far in the literature. In order to build such a database, we develop an interactive tool, with which each stroke is correctly and accurately extracted. First, a user selects a font library (i.e., FS library) whose strokes are to be extracted. Then the user selects points along stroke boundaries by clicking the left mouse button (Fig. 2(a)). When the right mouse button is clicked, the selected points are converted into a poly and region with black pixels in the poly is cut out from the original image. As a result, a binary image of the corresponding stroke is generated (Fig. 2(b)), together with a text recording the vertices of its bounding box. In this way, we can use those stroke images to re-draw the original character according to their bounding box vertices. When the current stroke has been extracted, our tool guides the user which stroke to extract next by marking the stroke of the reference character in a GB2312 Kaiti font library (Fig. 2(c)), in which the work of stroke extraction for every Chinese character has been done manually by a company called Founder Group. We directly use their data as reference information. Zoom functions are available in our tool so that the user can segment the singular regions precisely. Besides, we also provide drawback and revise functions to accelerate the extraction procedure.

2.2 Standard stroke database

There are various font libraries of Chinese characters and it is not realistic to construct a database

(a) Selecting points by clicks; (b) After extracting one stroke; (c) Reference character indicating stroke order

Fig. 2 Using the stroke extraction tool

containing all of them. Besides, strokes of some font libraries are so cursive that even a human cannot distinguish one stroke from another. Thus it is necessary to select several typical font libraries the strokes of which can be well extracted using our tool described above.

Up to now, we have chosen 4 font libraries to build the standard stroke database. They are FS, LS, HT and HYKT respectively, which are the most commonly used font libraries in daily lives. There are numerous Chinese characters while the number of their stroke categories is limited. In this case, we select 639 characters as a sample set in each font library, guaranteeing all types of strokes included. The next step is to acquire standard strokes of those 639 characters in each font library by using our stroke extraction tool. Although each stroke can be extracted quickly (about 10 seconds per stroke on average), it is still a huge work because many Chinese characters are very complicated. It takes about a week to finish the job of manually extracting 639 characters in each font library, and we have spent a month building the proposed database, consisting of 4 font libraries. The stroke database could be easily expanded with the help of our segmentation tool to contain more font libraries, which makes evaluation results more comprehensive.

2.3 Evaluation metrics

Since we have a standard stroke database, a set of metrics are required to evaluate how well computer-generated strokes match the standard strokes. Several metrics have been proposed and used in prior work to evaluate image segmentations, and we adapt two of them. The first metric, Hamming Distance, is region-based and measures the consistency of stroke interiors. To measure how close stroke boundaries are to one another, we use Cut Discrepancy as a boundary-based method.

2.3.1 Hamming distance

The first metric, Hamming Distance, measures the overall region-based difference between two strokes. Given a Chinese character C, pixel (C) is defined as the set of all interior pixels in the image of C. Assuming there are totally n strokes in C, then (S_1, S_2, …, S_n) is the set of all standard strokes of C in our benchmark database as the ground truth, and (T_1, T_2, …, T_n) is the set of computer-generated strokes of C to be evaluated. Thus the Hamming Distance is defined as

$$\mathrm{HD(C)}=\sum_{i=1}^{n}\frac{\mathrm{pixel}(S_i)\cup\mathrm{pixel}(T_i)-\mathrm{pixel}(S_i)\cap\mathrm{pixel}(T_i)}{\mathrm{pixel}(C)}, \tag{1}$$

where "||x|| is the size of set x (e.g., ||pixel(C)|| is the total number of interior pixels in C). For a single stroke T_k generated by a computer, we can also calculate its precision by comparing it with the corresponding standard stroke S_k as follows:

$$\mathrm{precision}(T_k)=\frac{\mathrm{pixel}(S_k)\cap\mathrm{pixel}(T_k)}{\mathrm{pixel}(S_k)\cup\mathrm{pixel}(T_k)}. \tag{2}$$

The main advantage of Hamming Distance is that the metric gives a meaningful and quantitative evaluation by figuring out how much the computer-generated strokes and the standard strokes overlap. However, it cannot evaluate the shape correspondences of strokes. In common sense, strokes with smooth boundaries should get higher scores while Hamming Distance is not able to guarantee that. Therefore, a boundary-based method is also required.

2.3.2 Cut discrepancy

To measure the distances between boundaries of strokes, the second metric, Cut Discrepancy, sums the distances from points along the boundaries of computer-generated strokes to the closest points in the standard stroke boundaries, and vice-versa.

Assume that S_k is a standard stroke in our benchmark, T_k is a computer-generated one, and C_{S_k} and C_{T_k} are boundaries of S_k and T_k respectively. We use $d_G(p_1, p_2)$ to measure the Euclidean distance between two points on a two-dimension plane. Then the geodesic distance from a point $p_1 \in C_{S_k}$ to a set of points in the boundary C_{T_k} is defined as follows:

$$d_G(p_1, C_{T_k})=\min\{d_G(p_1, p_2), \forall p_2 \in C_{T_k}\}. \tag{3}$$

The Directional Cut Discrepancy, $\mathrm{DCD}(C_{S_k} \Rightarrow C_{T_k})$, of C_{S_k} with respect to C_{T_k} is

defined as the mean of the distribution of $d_G(p_1, C_{T_k})$ for all points $p_1 \in C_{S_k}$:

$$\mathrm{DCD}(C_{S_k} \Rightarrow C_{T_k}) = \mathrm{mean}\{d_G(p_1, C_{T_k}), \forall p_1 \in C_{S_k}\}. \quad (4)$$

Now we can give the definition of Cut Discrepancy, $\mathrm{CD}(C_{S_k}, C_{T_k})$, to be the mean of the directional functions in both directions, divided by a parameter(avgRadius) as follows:

$$\mathrm{CD}(C_{S_k}, C_{T_k}) = \frac{\mathrm{DCD}(C_{S_k} \Rightarrow C_{T_k}) + \mathrm{DCD}(C_{T_k} \Rightarrow C_{S_k})}{\mathrm{avgRadius}}. \quad (5)$$

where avgRadius in the equation is the average Euclidean distance from boundary points to the centroid of the stroke to avoid effects due to scale. After calculating the metric for each stroke, the Cut Discrepancy for a Chinese character C with n strokes is defined to be the average of them as

$$\mathrm{CD}(C) = \frac{\sum_n^{i=1} \mathrm{CD}(C_{S_i}, C_{T_i})}{n}. \quad (6)$$

The Cut Discrepancy metric has the advantage of providing a simple, intuitive measure of how well the stroke boundaries align. The disadvantage is that it lacks definition when extracted strokes have no boundaries at all (i.e., a stroke may be wrongly extracted with no black pixel in it). We present results for both Cut Discrepancy metric and Hamming Distance metric to be more comprehensive because the algorithm may have quite different performances according to different metrics.

3 Stroke Extraction Algorithms

Various algorithms have been proposed to solve the problem of stroke extraction, and we select three of them for evaluation in order to investigate the utility of the proposed benchmark.

3.1 Naive K-NN method

We apply the k-Nearest Neighbors algorithm (K-NN) as a naive method to extract strokes. The idea is quite simple. Generally, given a certain character of n strokes, interior pixels are classified into n groups and each group is regarded as a stroke. Firstly, the corresponding reference character in Kaiti font library is scaled to the same size of the target character which is to be extracted. Then the points in the overlap region of the two character images are treated as the training examples in the target character. Those training points are labeled to n (n is the number of strokes in the reference character) classes according to which stroke it belongs to in the reference character. After training, the left black pixel points are put into an unlabeled vector. Each test point in the vector is classified by assigning the label which is the most frequent among the k training samples nearest to it. Here k is a user-defined constant, and we simply use Euclidean distance as the distance metric. Finally, both test points and training points with the same label compose a complete stroke.

3.2 Lian's Method

Given a Chinese character (e.g., in the HT style), both the skeleton of itself and the skeleton of the corresponding reference character are extracted in the first step using classic thinning algorithm. Afterwards, the Coherent Point Drift (CPD)[9] algorithm is applied for registering the point set randomly sampled from the character's skeleton to the skeleton template of the corresponding reference character. The next step is to segment the skeleton of the character into skeletons of corresponding strokes based on the non-rigid registration result. Then, points on the contour of the character are assigned to the closest points on the skeleton. In the last step, the given Chinese character is decomposed into several strokes accurately by completing and smoothing those segmented contours.

3.3 Our method

In this paper, we improve the stroke extraction algorithm based on Delaunay triangulation[4].

3.3.1 Pre-processing and triangulation

We first extract the edge of the character image using Canny operator. A corner detector based on global and local curvature properties[10] is applied to get corner points on the edge. Then we trace pixels in the contour in clockwise direction and get closed contours. Each closed contour is segmented by corner points. By uniform sampling between each pair of

neighbor corners, points on the contour are evenly sampled. Afterwards, the sampling points on the contour are converted into triangular mesh using Constrained Delaunay Triangulation (CDT)[11], which is used to deal with points enclosed by a polygonal boundary so that triangles outside the boundary can be avoided. The main advantage of using CDT is that the singular region detection will be more effective and efficient. The number of triangles is proportional to the sampling rate. More sampling points resulting in more triangles give a better description of the character shape, but consume more time and cause less efficiency instead.

3.3.2 Singular region identification

The interior of the character boundary is completely filled with triangles after CDT process. We only need to focus on the junction triangles, which can be distinguished by the number of internal edges. Junction triangles have three internal edges while others have two or less. A character poly could be decomposed into stroke-like components given the information of singular regions, which will be generated by stroke crossing or touching with other strokes. A singular region is also known as an ambiguous zone because the original information about the continuity and shape description of the stroke has been missing or ambiguous. We find the observation that most junction triangles are located in the singular regions of the characters and we can use them to identify singular regions.

Some triangles that are not located in singular regions may be wrongly regarded as junction triangles. They are defined as spurious junction triangles (Fig. 3(a)). We use skeleton intersection points for eliminating spurious junction triangles, which helps identify singular regions and improves efficiency. Specifically, the matrix of distance between the central point of each junction triangle and all intersection points in the skeleton is calculated. For junction triangle J_i, assuming the minimum distance between its central point and intersection points is d_i, if $d_i>w$ where w is a constant value, then J_i is regarded as a spurious junction triangle and we simply drop it from the junction triangle set. Compared with the method of using the PBOD curve[12] information demonstrated in Ref. [4], using skeleton intersection points is much more efficient because calculating the number of the crests in the PBOD curve is so time consuming.

Singular regions are then represented by merging the true junction triangles. Assume J_1 and J_2 are two junction triangles, the minimum distance between vertexes of J_1 and J_2 is d_v , and the distance between two central points c_1 and c_2 of J_1 and J_2 is $d_c = \| \overline{c_1 c_2} \|$. If J_1 and J_2 satisfy either of the two rules ($d_v<T_v$ or $d_c<T_c$) they are to be merged.

3.3.3 Stroke extraction

Intuitively, each stroke can be represented by the connection of stroke segments and singular regions. But how to compose a complete stroke is a challenging task. To address this problem, Wang et al.[4] analyzes the sub-stroke continuity and search all the simple paths to find complete strokes. However, the main disadvantage of this unsupervised method is that a complicate stroke may be wrongly treated as the combination of several simple strokes.

In this paper, we give a data-driven solution which is much more effective to the problem. We definitely draw the conclusion that each stroke segment belongs to a unique stroke while a singular region could be shared with several strokes. Thereby, we deal with stroke segments and singular regions separately. Specifically, applying the same point registration algorithm demonstrated in Ref. [5], we get skeletons of corresponding strokes referencing Kaiti font library. Then, a metric will be calculated measuring the similarity between skeletons of stroke segments and complete strokes. For each stroke segment, it belongs to the stroke whose skeleton has the most similarity. The key to the problem is how to measure the similarity. Assuming s is the skeleton of a stroke segment, K is a complete stroke segment, and i and j are points in s and K respectively, the similarity is calculated as the following equation:

T_v and T_c are two thresholds which can be set as

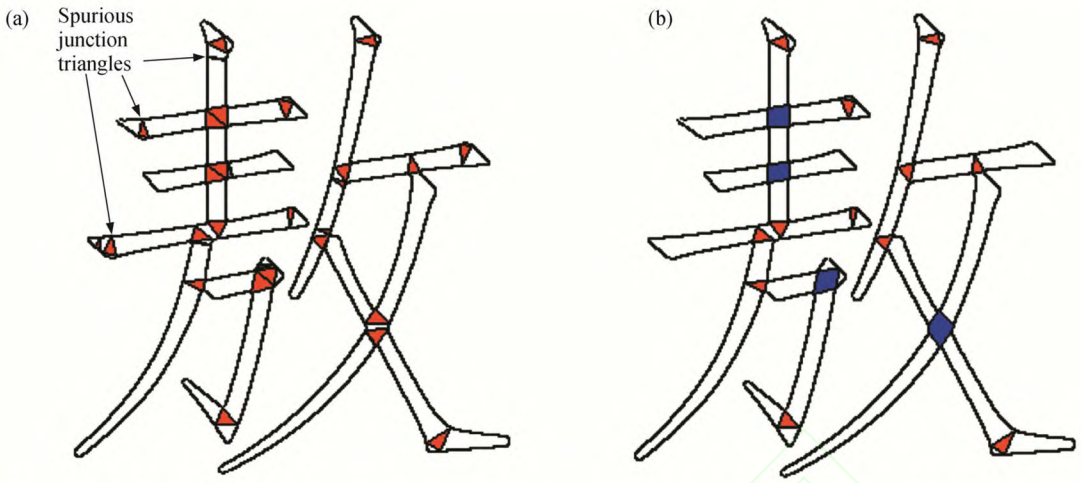


Fig. 3　All junction triangles including spurious ones in the triangular mesh of Chinese character *Ao* (a) and singular regions of the character by merging (in blue color) after eliminating spurious junction triangles (b)

input parameters. A singular region is identified when all corresponding junction triangles are merged into a convex hull. If a junction triangle has not been merged, then itself represents a singular region (Fig. 3(b)).

$$\text{Similarity}(s \to K) = \sum_{i \in s} \min\{d_G(i, j), \forall j \in K\}, \quad (7)$$

Where $d_G(i, j)$ identifies the Euclidean distance between point i and point j. For a certain stroke segment s, $K_{\min}$ in character C with the minimum value of Similarity ($s \to K$) is defined as the right stroke skeleton that s belongs to (Fig. 4(a)). When every stroke segment has been assigned, we put together all stroke segments belonging to the same stroke to represent the majority part of it, which we define as a "demo-stroke" (Fig. 4(b)). In the final step, singular regions need to be filled to compose a complete stroke. It is observed that each stroke should be a complete connected component, and for a demo-stroke, a singular region need to be filled only if it connects two stroke segments or more. When all singular regions have been filled, a complete stroke is correctly extracted (Fig. 4(c)).

4　Experimental Results

In this section, three algorithms mentioned above are evaluated on FS, HYKT, YH and LS font libraries respectively in the benchmark. Fig. 5 shows evaluations of these three algorithms according to the proposed benchmark. Either bar chart shows a different evaluation metric computed for algorithms in all four font libraries and averaged across the entire set of 639 characters in each library. In these two cases, lower bars represent better results.

In general, we can see that Lian's method and our method have great performances compared with the

(a) Stroke skeleton; (b) A demo-stroke; (c) A complete stroke

Fig. 4　A demonstration of assigning stroke segments and singular regions to compose a complete stroke

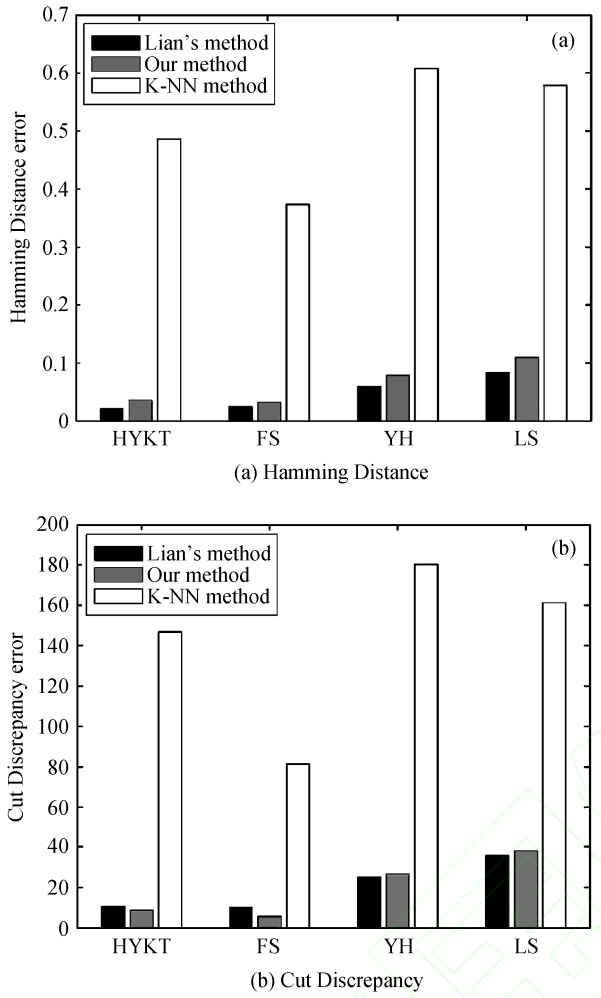


Fig. 5 Evaluation of stroke extraction algorithms with two metrics

naive K-NN method. Specifically, the bar chart in Fig. 5(a) shows that Lian's method has the best result in region detection according to the Hamming Distance metric in all font libraries. Fig. 5(b) implies that from the aspect of Cut Discrepancy metric, Lian's method performs best in YH and LS font libraries while our method has the lowest boundary error in HYKT and FS font libraries.

We can also calculate the precision of the stroke extraction algorithm using the combination of the two metrics. A character image is correctly segmented only if its Hamming Distance error is less than 0.1 and Cut Discrepancy error is less than 20, which means both the region and the boundary of the strokes are precisely extracted. Under this definition, precision of the three algorithms is shown in Table 1. From the table, it is summarized that Lian's method gets the highest score on the datasets of HYKT and YH style, and our method has the highest precision on FS and LS datasets. To make a horizontal comparison, it can be concluded that results evaluated on the datasets of HYKT and FS style are much better than YH and LS for all three algorithms. The conclusion is not surprising, since strokes of HYKT and FS style are more regular while those of YH and LS style vary a lot from the standard Kaiti font library, bringing difficulties for stroke extraction.

Table 1 Precisions of stroke extraction algorithms

	HYKT/%	FS/%	YH/%	LS/%
Naïve K-NN	2.82	7.51	2.03	1.72
Lian's Method	93.58	92.96	73.87	51.80
Our Method	90.92	95.31	72.61	54.46

5 Conclusion

This paper describes a benchmark for stroke extraction algorithms of Chinese characters. By using an interactive tool, we extract strokes manually to construct a standard database. Different algorithms will be evaluated by comparing their results with standard strokes and two proposed metrics will be given to show their performances. We find that each algorithm is given a quantitative and objective evaluation and the proposed benchmark is able to distinguish different algorithms effectively. However, there also exists some limitations: 1) many font libraries (i.e., *Songti* library) have not been contained in the benchmark; 2) the sample set of only 639 characters might be too small; 3)two metrics are not enough for more detailed evaluation information. It is for certain that more font libraries contained, more convincing results the benchmark will show. As we know, the most popular official character set GB2312 includes 6763 simplified Chinese characters. In fact, we are now expanding our benchmark database by providing standard strokes of various Chinese fonts under the GB2312 national standard. It is a long-playing work. In this way, we will expand our

stroke database in benchmark and try to find out more evaluation metrics in future work to give a more comprehensive evaluation. Besides, to prove that our benchmark works well, we are going to carry out the experiment in the future to check if the “better stroke extraction approach” evaluated by our benchmark is also a stroke extraction approach that results in a better effect of document analysis or the pattern recognition.

References

[1] Ku K M, Chiu P. Fast stroke extraction method for handwritten Chinese characters by cross region analysis. Electronics Letters, 1994, 30(15): 1210–1212

[2] Su Y M, Wang J F. Decomposing Chinese characters into stroke segments using SOGD filters and orientation normalization. Proceedings of the 17th International Conference on Pattern Recognition, 2004, 2: 351–354

[3] Sun Y, Qian H, Xu Y. A geometric approach to stroke extraction for the Chinese calligraphy robot // 2014 IEEE International Conference on Robotics and Automation. Hong Kong, 2014: 3207–3212

[4] Wang X, Liang X, Sun L, et al. Triangular mesh based stroke segmentation for Chinese calligraphy // 12th International Conference on Document Analysis and Recognition. Washington, DC, 2013: 1155–1159

[5] Lian Z, Xiao J. Automatic shape morphing for chinese characters // SIGGRAPH Asia 2012 Technical Briefs. Singapore, 2012: Article No. 2

[6] Liu C L, Kim I J, Kim J H. Model-based stroke extraction and matching for handwritten chinese character recognition. Pattern Recognition, 2001, 34(12): 2339–2352

[7] Zeng J, Liu Z Q. Stroke segmentation of Chinese characters using Markov random fields // 18th International Conference on Pattern Recognition. Hong Kong, 2006: 868–871

[8] Chen X, Golovinskiy A, Funkhouser T. A benchmark for 3D mesh segmentation // ACM Transactions on Graphics (TOG). 2009, 28(3): 341–352

[9] Myronenko A, Song X. Point set registration: coherent point drift. IEEE Transactions on Pattern Analysis and Machine Intelligence, 2010, 32(12): 2262–2275

[10] He X C, Yung N H. Corner detector based on global and local curvature properties. Optical Engineering, 2008, 47(5): 057008

[11] Chew L P. Constrained delaunay triangulations. Algorithmica, 1989, 4(1–4): 97–108

[12] Cao R, Tan C L. A model of stroke extraction from Chinese character images // Proceedings 15th International Conference on Pattern Recognition. Barcelona, 2000: 368–371