A Novel Method for Vectorizing Historical Documents of Chinese Calligraphy

Junsong Zhang, Hongwei Lin, Jinhui Yu
Zhejiang University
State Key Lab of CAD&CG
Hangzhou, Zhejiang, 310058, China
{jszhang,hwlin,jhyu}@cad.zju.edu.cn

Abstract

We develop a novel method for feature point detection and employ it to generate outline font from historical document of Chinese calligraphy. The feature points at a character contour subdivide the contour into segments. Each segment can be then fitted by a parametric curve to obtain the outline font. Some experimental results are also presented in the paper.

1. Introduction

As the art of writing Chinese characters, Chinese calligraphy has a long history more than four thousand years. Currently, with the development of digital library, a large number of ancient Chinese calligraphy documents have been transformed into digitized form by scanning them into computer. Recording calligraphy document images at pixel level requires a lot of storage space [1]. Therefore, it is necessary to convert these character images to outline font, such as True-Type Font [2]. In True-Type Chinese font, every character is represented with one or several closed contours, and each closed contour is comprised of one or more contour segments, which are fitted with straight lines or Bézier curves according to their bending strength. Outline font form of ancient Chinese calligraphy is also convenient for research of digital calligraphy document, calligraphy character recognition, calligraphy document press, and calligraphy education.

However, the scanned document images of Chinese calligraphy are generally noisy, as a result, the detected character boundaries from the document images are usually rough, the feature points are thus hard to be detected accurately. Research on feature point detection(FPD) from planar shapes has been conducted for a couple of decades, and many methods have been proposed [3]. These methods mainly fall into two categories, curvature scale space (CSS) based methods [4][5] and region of support (ROS) based methods [6][7]. The ROS [8] based FPD algorithms mainly depend on the decision of a region of support for every point on the curve. However, the ROS is difficult to accurately determine, especially when the contour is noisy. Most of the previous methods need pre-defined parameters for determining the ROS. The scale space concept was introduced by Witkin [9]. And the CSS based FPD algorithm has been suggested that if too large a scale is selected, some feature points of fine features will be missed, and vice versa [8]. It is also computationally intensive due to the iterative convolutions.

In this paper, we propose a novel algorithm for detecting feature points on the noisy character contours extracted from historical calligraphy images. The key advantages of our method are that the calligraphic characteristics of the original characters is retained and the noises are removed from the character outline in the resulting outline font.

In recent years, some research has been carried out to concerning the geometric shape of Chinese font, such as shape description with parametric curves and straight lines [10][11], or skeletal strokes [12]. Shamir and Rappoport [13] presented an algorithm for automatically extracting typographic elements of typefaces from their outline representation, which is one of the most important stages in digital font production. Su et al. [14] proposed an algorithm to simulate the elegant brush strokes found in calligraphic characters and black ink paintings utilizing the interval spline. And Ip et al. [15] developed a fractal-based outline font description that is able to capture the outline characteristics of calligraphic writings. Other methods focused on the brush stroke's boundary [16][17][18] and its trajectory [19]. Shamir and Rappoport [20] introduced a parametric method to compactly represent existing outline based oriental fonts. Based on the shape representation of a character, the process of rasterization is applied to generate the image of the character. For the purpose of desktop publishing [21], the problems generating new fonts were addressed by [22] and Pan et al. [23]. Henmi and Yoshikawa [24] described a virtual calligraphy system. Wong and Ip

978 -1-4244-1579-3/07/$25.00 © 2007 IEEE

[25] used the cone and some ellipses to synthesize Chinese calligraphic writings. There are also hardware approaches to implementing brushes, such as the one by [26]. From the viewpoint of font design and geometry modeling of Chinese font, our work approaches this problem differently as compared to the research outlined above.

Another research field related to our work is about vectorizing image. The method concerned with vectorizing image is widely applied, such as in the area of raster to vector conversion of engineering drawings [27], the aim of vectorization is usually to convert the image represented by pixels into vectors and other simple geometrical units. For example, Janssen and Vossepoel [28] proposed a novel method for vectorizing line drawing images, their method is based on a sequence of a standard vectorization algorithm and maximum threshold morphology, and the curves are represented with sequences of short straight lines. To compute parametric curves for approaching the data points, Chang and Yan [29] presented an optimization technique to vectorize hand-drawn key frames in a computer-aided cartooning system. Van Nieuwenhuizen et al. [30] provided a method for computing a vector representation of an arbitrary line in a raster image, their method integrates contour following and median line determination, and also offers an interactive way to choose directions while tracking line crossings and branches. Hilaire and Tombre [31] presented a method to vectorize the graphical parts of paper-based line drawings. Their method includes separating the input binary image into layers of homogeneous thickness, isolating the skeleton from each layer, and segmenting the skeleton by a method based on random sampling. Yang et al. [32] proposed a method to vectorize Chinese characters in calligraphy documents. They focused on the approximation of character contours based on the adjustment of control points of cubic Bézier curves. They also detected the feature points from the traced character contours depending on a presetting curvature threshold. In fact, this method is impractical for a real-life application, because the characters from the document image may be comprised of several different contours. It is tedious work to set different curvature thresholds for different contours.

The remainder of this paper is organized as follows. In Section 2, the preprocessing for outline font generation is introduced. In Section 3, a linear regression based FPD method is proposed. In Section 4, a curve approximating method is presented to express the character contours in outline font form. Section 5 present some experimental results and Section 6 draws a conclusion.

2. Character outline extraction and tracing

To extract the outline of characters from calligraphy documents images, we choose Canny operator [33] as the outline detector. Canny operator is an optimal edge detector and also the most commonly used edge detector. Canny operator works in a multi-stage process. It smoothes images using a Gaussian kernel to reduce false edge points, and suppresses a point if its gradient magnitude is smaller than either of its two neighbors along the gradient direction.

Let I denote an input binary character image, after the character outline is extracted, the value of outline pixels is set 0 and the value of other pixels is set 1 to get a binary image I_c. We save all pixels with 0 value in a list $L(l_1, l_2, ..., l_k)$, and record the pixels of each closed contour in l_k in clockwise direction according to the result of contour tracing. To trace a closed contour, we scan the pixels of I_c from bottom to top and left to right until a black pixel s is found, s is taken as the start point for contour tracing. The tracing process terminates when s is visited again. Most of Chinese characters consist of multiple strokes, thus we may get several closed contours after the character outline is extracted. In the case of multiple closed contour need to be traced, we use a flag f for every contour point, if a pixel is traced, its flag f is set to be 1, otherwise the flag f is set to be 0. The use of flag in our system ensured that all closed contours to be traced.

3. Feature points determination using linear regression method

In this section, we will explain the linear regression method for feature point determination from character contour. First, we select some basic strokes and overlapping stroke components to construct a sample database. Second, we use the linear regression to reveal the relation between the curvature sum of stroke contours and the standard deviation of Gaussian kernel. Based on the regression line, we can compute a standard deviation of Gaussian kernel for the given curvature sum of a new character contour and use it to smooth the given stroke contour. Candidates of feature point can be obtained by computing local extrema from the smoothed curvature function at this standard deviation. And the final feature points on the character contour are determined by applying an angle threshold derived with PCA method.

A Chinese character may contain basic strokes ranging from 1 to about 30, in which some basic strokes may overlap each other to form different components. These basic strokes and overlapping stroke components may result in closed contour corresponding to their outlines. So we should take both basic stroke and overlapping stroke components into account. We choose 120 representive strokes and components to construct the sample database.

After the sample database is constructed, we extract their outlines and calculate the absolute curvature for each point on the closed contour. The curvature function is then convolved with a Gaussian kernel. And the standard deviation of Gaussian kernel is selected according to the noise level of stroke contour by users interaction. Generally, the Gaussian function is defined as

$$g(s,\sigma) = \frac{1}{\sigma\sqrt{2\pi}} e^{\frac{-s^2}{2\sigma^2}} \tag{1}$$

where σ is the standard deviation of Gaussian kernel, acting as the scale. The convolution at the scale σ between the curvature function and the Gaussian function is

$$C(l,\sigma) = \int_{u=l-\frac{L}{2}}^{u=l+\frac{L}{2}} c(u) g(l-u,\sigma)\, du. \tag{2}$$

Here, L is the total length of the curvature curve, and it is equal to the number of total points on the contour. $c(u)$ is the absolute curvature at the contour point, and $C(l,\sigma)$is the convolution curvature curve at the scale.

According to our experiment, the selected standard deviation σ for smoothing a given character contour is related to the curvature sum of the character contour. To reveal the relation between the curvature sum and the standard deviation of Gaussian kernel, for each contour in the sample contour database, we compute the curvature sum by adding curvatures of all contour points on a single closed contour together, and select an optimal scale σ by user interaction. Taking the curvature sum as x-coordinates, the scale as y-coordinates, a discrete point set $(x_i, y_i), i = 1, 2, ..., n$, is obtained in the x-y plane.The relation between the curvature sum and the scale can be constructed using the linear regression model [34]. Thus, if given a character contour, after calculating its curvature sum, the scale for smoothing the curvature function can be determined. And then, the points on the character contour, which correspond to the local extrema of the smoothed curvature function, are extracted as the feature point candidates. Feature point candidates derived from the linear regression contain both true feature points and false feature points caused by noise. We preserve the true feature points and discard false ones.

4. Contour segment approximation

With the feature points determined above, a character contour is divided into some contour segments, and each contour segment is constituted by a sequence of contour points between two adjacent feature points. We depict the contour segment with a parametric curve. In computer-aided geometric design, computer graphics, computer vision, and pattern recognition, fitting contour points with a smooth curve is a frequently encountered issue. It is beyond our scope to give a comprehensive review of the different methods for curve fitting. Our method for approximating the contour segment is similar to that of Sarfraz and Khan's [35], but with some modifications. We describe the contour segments with Bézier curves and straight lines. Bézier curves are a common choice for the approximating curve. In the case the Bézier curve is selected, we need to determine the control points of Bézier curves. And to represent the contour segments using Bézier curves, the control points for Bézier curve need to be determined. We use the least square fitting to determine the control points of Bézier curves. In the following, we just describe how to obtain the control points of a cubic Bézier curve. As for quadric Bézier curve, it is computed with the same way. Generally, a cubic Bézier is expressed as follows:

$$\begin{aligned} Q(t_i) = (1-t_i)^3 p_0 + 3t_i(1-t_i)^2 p_1 + 3t_i^2(1-t_i) p_2 \\ + t_i^3 p_3, 0 \le i \le n-1 \end{aligned} \tag{3}$$

where P_0, P_1, P_2, P_3 are the four control points of the cubic Bézier, and $P_0 = S, P_3 = E$. The least square fit is obtained by choosing the P_1 and P_2 , so that

$$\rho = \sum_{i=0}^{n-1} [p_i - Q(t_i)]^2 \tag{4}$$

achieve a minimum. This requires:

$$\frac{\partial \rho}{\partial p_1} = 0, \frac{\partial \rho}{\partial p_2} = 0. \tag{5}$$

P_1 and P_2 can be obtained by solving the above equation, which together with two end points P_0 and P_3 are used as the control points to generate a cubic Bézier to fit the contour segment. In Fig.1 and Fig.2. Where the green, blue, and black curves suggest that the contour segments are fitted with straight lines, quadric Bézier, and cubic Bézier respectively.

5. Results

In this section, two test examples are presented to demonstrate our method. The test calligraphy images were obtained through scanning of published tablet images.

In Fig.1 and Fig.2, the left of each figure is the original binary image, the middle illustrates the detected outlines and feature points, and the right demonstrates the fitting result with parametric curves. Although the images are noisy on the character's outline, especially the character in Fig.2, our method is still able to detect the feature points correctly. And the fitting result of the character outline looks satisfactory.

It should be pointed out that, in the case that the character contour is very noisy, our method is unable to achieve good results, because false feature points caused by noise are compatible with true ones in sizes.

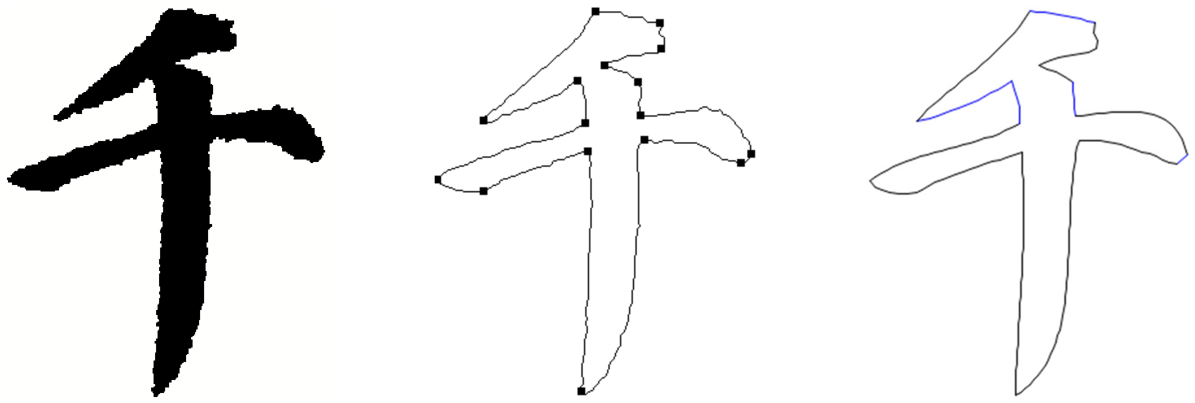

Figure 1. Chinese character ”Qian”. Left: binary image; Middle: contour and feature points; Right: the fitting result.

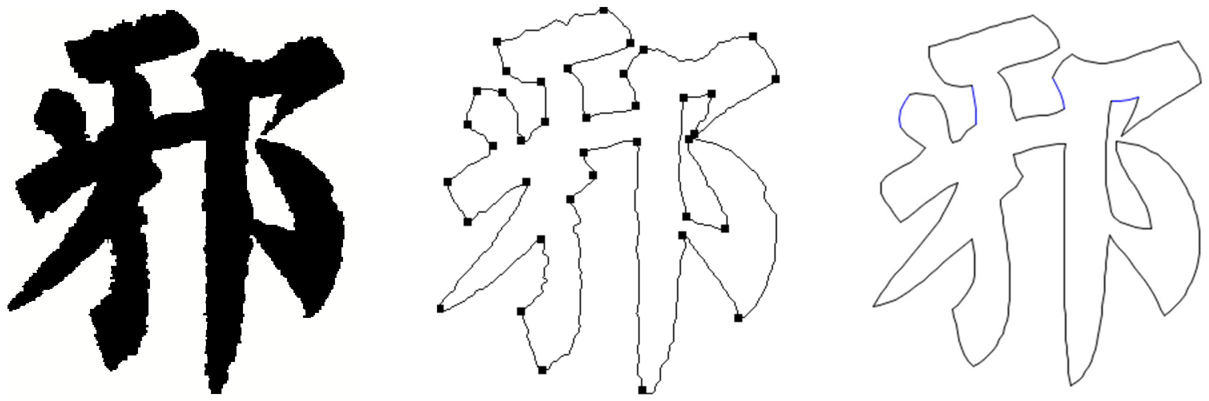

Figure 2. Chinese character ”Xie”. Left: binary image; Middle: contour and feature points; Right: the fitting result.

6. Conclusion

In this paper, a scheme is proposed for the vectorization of ancient Chinese calligraphy document images. With the scheme feature points are detected from character contours and contour segments are approximated with the straight lines and Bézier curves. The resultant contours look much smoother than that of the noisy contours as demonstrated by the examples in this paper. And the generated outline font preserves the style of the original calligraphy well, which can facilitate the further operation of the calligraphic characters, such as using the outline of calligraphic characters in advertising design. We also expect that the system can be used for computer-aided typeface design.

Acknowledgements

The work described in this paper was supported by the grants from Hi-tech Research and Development Program (863) of China (Project No. 2006AA01Z303310), and the Foundation of State Key Basic Research (973) of China (Project No. 2004CB719400), and the National Natural Science Foundation of China (Project No. 60673007 and No.60503057).

References

[1] Wright, T., *History and Technology of Computer Fonts*, IEEE Annals of the History of Computing, 20(2), 30-34, 1998.

[2] Apple Computer Inc., *The TrueType Font Format Specication*, Version 1.0.,1990.

[3] Marji, M., *On the detection of dominant points on digital planar curves*, PhD Thesis, Wayne State University, Detroit, Michigan, 2003.

[4] Rattangsi, A., Chin, R., *Scale-Based Detection of Corners of Planar Curves*, IEEE Transactions on Pattern Analysis and Machine Intelligence, 14 (4), 430-449, 1992.

[5] Sezgin, T.M., Davis, R., *Scale-space Based Feature Point Detection for Digital Ink*, In Making Pen-Based Interaction Intelligent and Natural, 145-151, 2004.

[6] Rosenfeld, A., Johnston, E., *Angle Detection on Digital Curves*, IEEE Transactions on Computers, 22(9), 875-878, 1973.

[7] Rosenfeld, A., Weszka, J.S., *An Improved Method of Angle Detection on Digital Curves*, IEEE Transaction on Computers, 24, 940-941, 1975.

[8] Teh, C.H., Chin, R.T., *On the Detection of Dominant Points on Digital Curves*, IEEE Transactions on Pattern Analysis and Machine Intelligence, 11(8), 859-872, 1989.

[9] Witkin, A.P., *Scale-space Filtering*, Proc. 8th Int. Joint Conf. Art. Intell, 1019-1022, 1983.

[10] Chua, Y., *Bezier brush strokes*, Computer Aided Design, 22(9),550-555, 1990.

[11] Nishita, T., Takita, S., Nakamae, E., *A display algorithm of brush strokes using Bezier functions*, Proceedings of Computer Graphics International Conference. Lausanne, Switzerland, 244-257, 1993.

[12] Shu, S., Lee, I., *Drawing and animation using skeletal strokes*, Proceedings of SIGGRAPH'94. Orlando, Florida, 109-118, 1994.

[13] Shamir, A., Rappoport, A., *Extraction of typographic elements from outline representations of fonts*, Computer Graphics Forum, 15(3), 259-268, 1996.

[14] Su, S.L., Xu, Y.Q., Shum, H.Y., Chen, F.L., *Simulating Artistic Brushstrokes Using Interval Splines*, In Proceedings of the 5th IASTED International Conference on Computer Graphics and Imaging (CGIM '02), Kauai, Hawaii, 85-90, 2002.

[15] Ip, H.H.S., Wong, H.T.F., Mong, F.Y., *Fractal coding of Chinese scaleable calligraphic fonts*, Computers&Graphics. 18, 343-351, 1994.

[16] Ahn, J.W., Kim, M.S., Lim, S.B., *Approximate general sweep boundary of a 2D curved object*, Computer Vision, Graphics & Image Processing. 55 (2), 98-128, 1993.

[17] Posch, K.C., Fellner, W.D., *The circle-brush algorithm*, ACM Transactions on Graphics. 8(1), 1-24, 1989.

[18] Lim, S.B., Kim, M.D., *Oriental character font design by a structured composition of stroke elements*, Computer Aided Design. 27(3), 193-207, 1995.

[19] Hobby, J.D., *Digitized Brush Trajectories*, Ph.D thesis, Stanford University, 1985.

[20] Shamir, R., Rappoport, A., *Quality enhancements of digital outline fonts*, Computers & Graphics. 21(6), 713-725, 1997.

[21] Hao, L., Zhou, H., *A new contour fill algorithm for outlined character image generation*, Computers & Graphics, 19(4), 551-556, 1995.

[22] Coueignoux, P., *Character generation by computer*, Computer Graphics & Image Processing, 16(3), 240-269, 1981.

[23] Pan, Z.G., Ma, X.H., Zhang, M.M., Shi, J.Y., *Chinese font composition method based on algebraic system of geometric shapes*, Computers & Graphics, 21(3), 321-328, 1997.

[24] Henmi, K., Yoshikawa, T., *Virtual lesson and its application to virtual calligraphy systems*, Proceedings of IEEE international Conference on Robotics & Automation, 1275-1280, 1998.

[25] Wong, H.T.F., Ip, H.H.S., *Virtual brush: a model based synthesis of Chinese calligraphy*, Computers & Graphics. 24(3), 99-113, 2000.

[26] Greene, R., *The drawing prism: a versatile graphic input device*, Proceedings of SIGGRAPH. 103-109, 1985.

[27] Gibson, L., Lucas, D., *Vectorization of Raster Images Using Hierarchical Methods*, Computer Graphics & Image Processing. 20(1), 82-89, 1982.

[28] Janssen, R.D.T., Vossepoel, A.M., *Adaptive Vectorization of Line Drawing Images*, Computer Vision and Image Understanding, 65(1), 38-56, 1997.

[29] Chang, H.H., Yan, H., *Vectorization of hand-drawn image using piecewise cubic Bezier curves fitting*, Pattern Recognition, 31(11), 1747-1755, 1998.

[30] Van Nieuwenhuizen, P.R., Kiewiet, O., Bronsvoort, W. F., *An Integrated Line Tracking and Vectorization Algorithm*, Computer Graphics Forum, 13(3), 349-359, 1994.

[31] Hilaire, X., Tombre, K., *Robust and Accurate Vectorization of Line Drawings*, IEEE Transactions on Pattern Analysis and Machine Intelligence, 28(6), 890-904, 2006.

[32] Yang, H.M., Lu, J.J., Lee H.J., *A Bezier Curve-Based Approach to Shape Description for Chinese Calligraphy Characters*, 6th International Conference on Document Analysis and Recognition (ICDAR 2001), Seattle, WA, USA, 276-280.2001.

[33] Canny, J., *A Computational Approach to Edge Detection*, IEEE Transactions on Pattern Analysis and Machine Intelligence, 8(6), 679-698, 1986.

[34] Weisstein, Eric W., *Least Squares Fitting*, From MathWorld–A Wolfram Web Resource, http://mathworld.wolfram.com/LeastSquaresFitting.html, 2002.

[35] Sarfraz, M., Khan, M., *An Automatic Algorithm for Approximating Boundary of Bitmap Characters*, Future Generation Computer Systems, 20(8), 1327-1336, 2004.