Sketch Parameterization Using Curve Approximation

Zhengxing Sun, Wei Wang, Lisha Zhang, and Jing Liu

State Key Lab for Novel Software Technology, Nanjing University, P. R. China, 210093
szx@nju.edu.cn

Abstract. This paper presents a method of parameterization for online freehand drawing objects based on a piecewise cubic Bezier curve approximation. The target is to represent sketches in a compact format within a certain error tolerance with lower computation to be practically adaptable for the online graphics input. A set of user's intended breakpoints in digital ink is firstly produced in terms of pen speed and local curvatures. Each of strokes of a skechy shape is then parameterized by the optimization of piecewise Bezier curve approximation to minimize the fitting error between stroke path and the curve. The experimental results show both effective and efficient for a wide range of drawing graphic objects.

1 Introduction

As computers become integrated into everyday life, pen-based user interface is considered as a primary input method. Moreover, the feature to rapidly visualize and deliver human's ideas using graphic objects, which cannot be efficiently represented by speech or text, is highly desirable in graphic computing [1]. The rapid growth of graphic data has sustained the need for more efficient ways to represent and compress the sketchy graphic data. The data representing freehand sketching needs not only to be compressed in order to reduce the internal handling size and to transfer in low bandwidth, but also to preserve the original intention of user and the convenience of easy access of the information and for further processing such as shape recognition, cooperative design, idea permutation and so on.

For existing techniques, the pen movements are typically captured by a digitizing tablet and stored as sampled pen points of their paths, so called as digital ink, while an image for receptor is captured. The drawback of this technique is that sketches transferred in image usually require considerable storage capacity and cannot be modified by receptor. Although there have been a large amount of experiments on sketchy graphics recognition, such as feature-based [2][3], graph-based [4][5], machine learning [6][7][8] and Parametric methods such as polygon [9], B-spline [10] and Bezier curve [11], most of them guess and convert the drawing sketches into regular shapes. However, they are charged with desertion of users' intension, and are the burden of computation especially for mobile devices. Parametric methods fitting techniques have been considered in shape representation and classification. A benefit of these approaches is that they can approximate the path of pen movements during user drawing with a few parameters and they are computationally efficient. Only a few researches have bent themselves to this issue [11][12].

In this paper, a sketch approximating method is introduced, which performs the efficient parameterization of on-line freehand drawing objects captured in digital ink using a piecewise recursive cubic Bezier curve approximation. The target is to represent freehand sketches in a compact format, achieving high compression rate within a certain error tolerance and with lower computation to be practically adaptable solution for the real applications.

The remainder of this paper is organized as follows: The main idea of our proposed strategy is outlined in Section 2. In Section 3, the method of stroke fragmentation for generation of user intended breakpoints is given. In Section 4, we will discuss sketch parameterization by Recursive Bezier Curve Approximation in detail. Section 5 will present our experiments. Conclusions are given in the final Section.

2 The Proposed Strategy of Sketch Parameterization

The Bezier curve representation has been widely used since its coefficients can be easily obtained and its shape can be easily manipulated. A Bezier curve is defined using two anchor points, on-curve control points and at least one shape point, off-curve control point. The on-curve control points are the two end points of the curve actually located on the path of the curve, while the other off-curve control points define the gradient from the two end points, which are usually not located on the curve path. The off-curve points control the shape of the curve. The curve is actually a blend of the off-curve control points. The more off-curve control points a Bezier curve has, the more complicated shape can be represented; however the order of the mathematical curve equation becomes higher. The approximation of hand-drawing strokes using high order Bezier curve reduces the number of on-curve points and produces more compact data representation. However, the approximation of high order curve equation usually requires large amount of computation.

In order to easily manipulate a sketch, Raymaekers et al [11] have proposed a method to group the individual pixels of sketch into segments represented by a cubic Bezier curve. Whenever the least square error of the curve fitting passes a preset threshold, a new segment is created and two off-curve points of this segment can be constructed by means of least square minimization. For the computational efficiency and data compression of sketch, Park and Kwon [12] have recently presented a method of sketch approximation by piecewise fitting of a series of quadratic Bezier curves using least square error approximation. The common ground between the above two methods is that the curve control points are produced dependent only upon the local geometric curvature so that the difference between the curve and the corresponding data points is as small as possible. However, it is not a reliable way to deliver the users' intentions that the curve control points are determined alone by curvature information. It is also time consuming to fit the undetermined segmented points using least square error approximation.

We propose a method to parameterize sketches using a cubic Bezier curve, where the complicated shaped strokes are represented by piecewise approximated cubic Bezier curves, because the Bezier curves can only represent simple arc shape curves. **Fig. 1** shows the processing diagram of our proposed approximation method. This process has two independent sub-processing modules, stroke fragmentation and curve approximation loop. The stroke fragmentation is performed only once per given input set of strokes to generate a series of the user intended breakpoints at each stroke, where the speed of pen movement reaches the minimal value and where the curvatures change sharply. These candidate breakpoints are selected as the initial on-curve control points in curve approximation loop. The curve approximation routine, including curve control points (including on-curve and off-curve control points) selection, Bezier curves approximation and fitting error evaluation, operates recursively till the piecewise fitting error is satisfactory. The recursive curve approximation loop has two nested sub-processes. The inner is the optimization of off-curve control points, which optimizes the piecewise curve approximation to minimize the fitting error. The outer is the adjustment of on-curve control points, which generates some new on-curve control points by bisecting each of current curves to increase the pieces of segments and reduce the fitting error. The inputting graphic object, drawn by a set of strokes, can finally be represented as a set of parameters of some pieces of cubic Bezier curve. The risk generating excessive breakpoints (on-curve points) on the stroke because of using series of cubic Bezier curves (compared to approximated by high order Bezier curves) can somewhat avoided by optimization of off-curve control points.

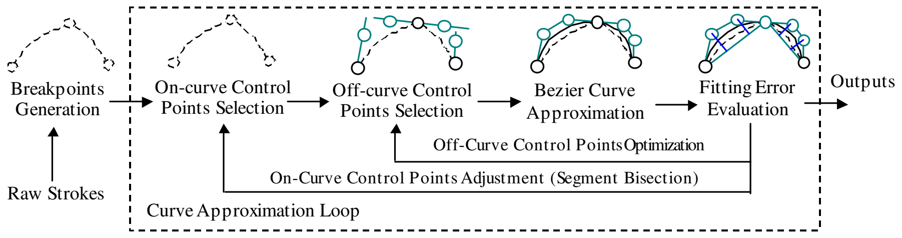

Breakpoints Generation
Raw Strokes
On-curve Control Points Selection
Off-curve Control Points Selection
Bezier Curve Approximation
Fitting Error Evaluation
Outputs
Off-Curve Control Points Optimization
On-Curve Control Points Adjustment (Segment Bisection)
Curve Approximation Loop

Fig. 1. The Diagram of sketch parameterization using recursive curve approximation

3 User Intended Breakpoints Generation

Stroke fragmentation is a very basic problem, making it widely applicable to intelligent ink manipulation as well as other higher-level digital ink analyses. The goal of stroke fragmentation is to fragment a wide variety of sketched symbols into simpler structures so that they could faithfully represent the original form with a less complex form. The key challenge of stroke fragmentation is to find out which bumps and bends (breakpoints) are intended and which are accident. Most of existing methods do stroke fragmentation based on curvature only. Usually, curvature information alone is not reliable enough to determine such points. Instead, the speed based methods have proven to be a much more reliable measure to determine the intended breakpoints for the observation that the speed of the pen tip significantly reduces at the intended corner points [5][13]. For sketch parameterization, the purpose of stroke fragmentation is to extract sharp turning pen movement points and bending points at each stroke quickly by the information of pen speed and curvature, which are located as initial on-curve control points of Bezier curves.

3.1 Candidate Breakpoints Selection Based on Pen Speed

It has been discovered that it is natural to slow the pen when making many kinds of intentional discontinuities in the shape [5][13], for instance the corners formed by two lines. Similarly, when drawing a rectangle with a single pen stroke, users would likely slow down at the corners, which are supposed to be the segment points. **Fig. 2.** shows the speed profile for a typical square in our experiments. The corners can be easily identified by the low pen speed.

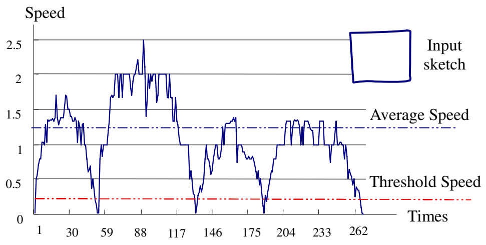


Fig. 2. 1 Illustration of speed profile for a square

Given a digital stroke ***S***, which contains *N* numbers of time-ordered points {$\boldsymbol{P_1}$, $\boldsymbol{P_2}$,…, $\boldsymbol{P_N}$}. There is also a corresponding speed sequence: {$\boldsymbol{SP_1}$, $\boldsymbol{SP_2}$,…, $\boldsymbol{SP_N}$}, where $\boldsymbol{SP_i}$ represents the pen speed at the point $\boldsymbol{P_i}$. The point $\boldsymbol{P_i}$ would then be selected as a candidate breakpoint when the following conditions are satisfied:

$$\begin{cases} SP_i < \overline{SP} * a \\ \forall j \;\; SP_i < SP_j, \mid j - i \mid < N * b \end{cases} \tag{1}$$

where, $\overline{SP}$ is an average pen speed of the whole stroke, *a* and *b* are two thresholds which will affect the spacing of the candidate breakpoints dependent on the length of a stroke and must be set by means of some statistical experiments.

3.2 Candidate Breakpoints Generation Based on Curvature

In some case, some users' intended breakpoints cannot be located with pen speed because users probably draw it smoothly without the variation of pen speed. The information of curvature must be used to capture such points. We develop an approximate method to measure the variation of curvature of a segment or a stroke and use greedy method based on sliding window algorithm (SWA) to pick up the candidate breakpoints. Given a window with fixed sizes that envelops a lot of successive ink points of a stroke, the distances $\boldsymbol{d_{ink}}$ from each sample point to the line connecting the first point and last point in the window can be calculated and signed according to their relative positions along with the drawing direction, that is, the distance located in left side is positive and in right side is negative, as shown in **Fig. 3**(a). The sum of distances of all ink points between these two points can be seen as a measurement of the curvature of a segment affected by two end points.

Accordingly, starting with the first sample point of a stroke, we set a sliding window initially with one unit width that envelops at least two successive ink points of the stroke, the measurement of the curvature of segment in the window are calculated, as shown in **Fig. 3**(b). The width of window broadens step by step alone with drawing direction of stroke and the measurement of the curvature of segment in the window are incrementally calculated until the measurement exceeds the experimental threshold or all ink points of the stroke are tested. Whenever a measurement exceeds the experimental threshold, the last point in that window is chosen to be a candidate breakpoint and as a new start point of next repetition. Then, the window with original size moves to the position of that point, the same process described above is repeated until all successive points in a window are examined, as shown in **Fig. 3**(b). It can minimize the chance of over-splitting caused by jitter of pen movements. The window size are dependent on the density of ink points for a stroke and can be given as a function of the perimeter and bounding box of each stroke. The step of window enlargement and the threshold of measurement of curvature must be experimental defined by making a tradeoff between precision and efficiency. In our experiments, two parameters set as 15 (pixels) and 150 (pixels) respectively.

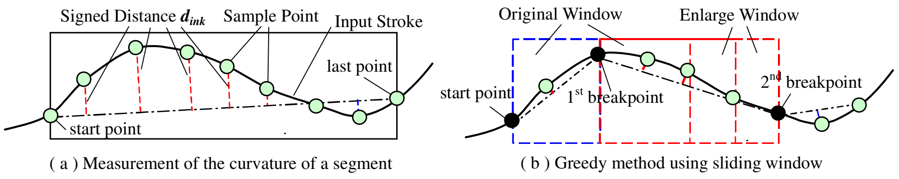


Fig. 3. Candidate Breakpoints Generation Using Curvature

After the processing described above, two portions of candidate breakpoints can be got besides two end points of a stroke, which are either the ink points where the pen speed reaches the local minimum or the ones where the measurement of curvature is local maximum. However, some of them are redundancy, and the candidates mergence is required to remove the superfluous or adjacent points. Ultimately, any two adjacent breakpoints separate the segment between them as a sub-stroke and will be used as the initial on-curve control points of a cubic Bezier curve.

4 Sketch Parameterization Using Piecewise Cubic Bezier Curve

4.1 Principle of Sketch Approximating by Cubic Bezier Curve

For a fragmentized stroke (or a segment of the stroke) with some breakpoints (can be two end points only) to be eligible for the further processing, a cubic Bezier curve approximation method is applied to find out the curve control points. A cubic Bezier curve with two on-curve control points and two off-curve points are estimated which satisfy following conditions as shown in **Fig. 4**.

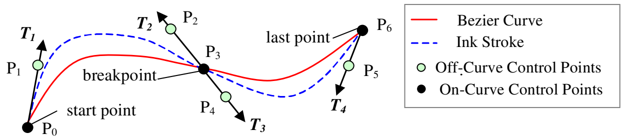


Fig. 4. Sketch approximating by cubic Bezier curve

(1). The two on-curve control points of the estimated curve should be breakpoints (including first and last ink points) of the stroke; such as, $\boldsymbol{P_0}$ and $\boldsymbol{P_3}$, $\boldsymbol{P_3}$ and $\boldsymbol{P_6}$**,** shown in **Fig. 4**.

(2). The two off-curve control points of the estimated curve should be located on the corresponding tangents at two end (on-curve control) points of the segment respectively; for example, $\boldsymbol{P_1}$.is located on the tangent $\boldsymbol{T_1}$ at $\boldsymbol{P_0}$, $\boldsymbol{P_2}$.is on the tangent $\boldsymbol{T_2}$ at $\boldsymbol{P_3}$, and so on, as shown in **Fig. 4**.

(3). For any two adjacent segments that share a common breakpoint such as $\boldsymbol{P_3}$ in **Fig. 4**, their tangents at that point should be collinear in order to make the curves connect smoothly, that is, $\boldsymbol{T_2}$ and $\boldsymbol{T_3}$ are collinear vectors in **Fig. 4**. In other words, the first off-curve control point in sequential segment should be collinear with two last curve control points in previous segment; for example, $\boldsymbol{P_2}$, $\boldsymbol{P_3}$ and $\boldsymbol{P_4}$ should be collinear.

(4). The fitting error between actual stroke and the approximated curve must be within an acceptable range by optimization of off-curve control points. If the fitting error exceeds a certain threshold, the on-curve control points should be adjusted. The recursive curve fitting operation is repeated for each of the newly generated segment till the error falls within tolerance.

For a given segment (or a whole stroke) with two end points $\boldsymbol{P_0}$ and $\boldsymbol{P_3}$, its approximated cubic Bezier curve with two on-curve control points $\boldsymbol{P_0}$ and $\boldsymbol{P_3}$ and two off-curve control points $\boldsymbol{P_1}$ and $\boldsymbol{P_2}$ can be represented as follow, as shown in **Fig. 4**.

$$P(u)=\sum_{k=0}^{3} P_k BEZ_{k,3}(u),\ 0\le u\le 1 \tag{2}$$

where, $BEZ_{k,3}(u)=C(3,k)u^k(1-u)^{3-k}$ is a cubic Bernstein polynomial function, and two off-curve control points $\boldsymbol{P_1}$ and $\boldsymbol{P_2}$ are defined based on the location and tangent of two on-curve control points (two end points of a segment or a stroke) $\boldsymbol{P_0}$ and $\boldsymbol{P_3}$, as follow respectively:

$$\begin{cases} P_1 = c_1 * T_1 + P_0 \\ P_2 = c_2 * T_2 + P_3 \end{cases} \tag{3}$$

where, $\boldsymbol{T_1}$ and $\boldsymbol{T_2}$ are two unit tangent vectors of two end points of a segment respectively, which can be approximated by calculating the delta difference of the end points to its adjacent ink points denoted as ΔP_0 and ΔP_3 respectively; c_1 and c_2 are two variable coefficients to locate the off-curve control points on the tangent of end points of the segment, usually we can set c_1=c_2=c, which is proportionate to the length of segment for uniform fitting of the segment.

Therefore, an approximated cubic Bezier curve can be represented as:

$$P(u) = [2(P_0 + P_3) + c(\Delta P_0 + \Delta P_3)]BEZ_{k,3}(u),\ 0 \le u \le 1 \tag{4}$$

4.2 Fitting Error Evaluation

To inspect the suitability of the sketch parameterization, the fitting error between actual ink points and corresponding points of the approximated curve must be evaluated. We evaluate the fitting error by two steps. Firstly, several pairs of points on the original ink segment and corresponding approximated Bezier curve are selected using fixed size sliding window, and all Euripidean distances from them to their opposite chord connecting two end points of a segment are calculated one by one as shown in **Fig. 5**, including a set of d_{ink} for each selected ink point and a set of d_{curve} for each selected curve point (for illustrating clearly, some scenes in Fig.5 such as space between sliding windows and lines in **Fig. 5** are artificial). Then, the distance difference (d_{curve}-d_{ink}) for each pair is calculated, and the fitting error is defined as the ratio of the maximal distance difference $(d_{curve}\text{-}d_{ink})_{max}$ to the length of chord L, which reflects approximately the maximal difference of the local curvature distribution between the original segment and the approximated Bezier curve at a certain extent as described in section 3.2.

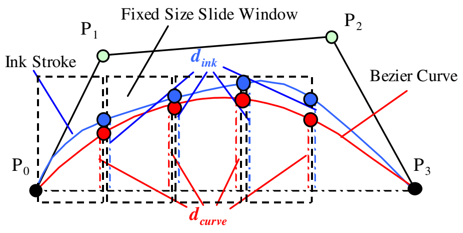


Fig. 5. The distance calculation using fixed size slide window for fitting error evaluation

Accordingly, the fitting error can then be represented as:

$$e_f = \max[e_f^1, e_f^2, \cdots, e_f^i, \cdots,, e_f^m], \text{Where } e_f^i = \left| \frac{(d_{curve}^i - d_{ink}^i)}{L} \right| \tag{5}$$

where, m is the numbers of selected on-curve points. The value of m must not be less than three, and the points with the maximal local curvature on the original ink path should be selected as possible. Meanwhile, The selected points from the original ink path and the approximated Bezier curve should be corresponding. By defining a local coordinate system, the distance difference (d_{curve}-d_{ink}) for each pair of selected points, as shown in **Fig. 5**, can be easily calculated as follows:

$$|d_{curve}^i - d_{ink}^i| = |[2(y_0 + y_3) + c(\nabla y_0 + \nabla y_3)]BEZ_{k,3}(u_i) - y_{ink}^i| \tag{6}$$

4.3 Sketch Parameterization

As described in section 4.1, every stroke of a sketchy object can be parameterized by piecewise cubic Bezier curve within an acceptable fitting error tolerance with a set of parameters as follows:

$$A\ stroke \propto \{(Pm_i \mid i = 0,1,2,\cdots,k),(c_i \mid i = 1,2,\cdots,k),e_f\}, e_f \leq e_{\exp} \tag{7}$$

where, $\{Pm_i\}$ is a series of on-curve control points, Pm_0 and Pm_k are two end points of a stroke respectively, $\{Pm_i|i=1,2,\ldots,k-1\}$ is a sequence of middle on-curve control points that fragmentize a stroke into $\boldsymbol{k}$ numbers of segments (initially, there are breakpoints generated by stroke fragmentation), $\{\boldsymbol{c}_i\}$ is a set of proportional coefficients for locating the off-curve control points of the approximated cubic Bezier curve for each of segments of a stoke, e_f and e_{exp} are the actual fitting error and the experimental threshold of the fitting error respectively.

Our key ideas of sketch parameterization are (i) to optimise off-curve control points of the piecewise curve approximation for each segments of a stroke to minimize the fitting error and (ii) minimize the numbers of new added breakpoints for each segment as possible to preserve the users' intention. That is, for each segment, the optimization of two off-line curve control points is the priority. The new ink breakpoint would be generated and inserted to bisect the segment into two sub-segments only if the fitting error of optimization of off-line curve control points great than the experimental threshold. Accordingly, for an inputting stroke, the task of sketch parameterization using piecewise cubic Bezier curve can be seen as a process to search out some parameters of curve approximation, including a series of stroke breakpoints and a set of the proportional coefficients, to fit as closely as possible to each of strokes. In fact, this process is the recursive piecewise cubic Bezier curve approximation within an acceptable approximated error, as shown in **Fig. 1**.

As described in equation (5) and (6), for every segment of a stroke between each pair of on-curve control points $\{Pm_{i-1},Pm_i|i=1,2,\ldots,k\}$, the fitting error is variable with this pair of points and a proportional coefficient $\{c_i|i=1,2,\ldots,k\}$ corresponding to the selection of off-curve control points, that is:

$$e_f^i \propto f(\{Pm_{i-1}, Pm_i\}, c_i), i = 1,2,\cdots,k \tag{8}$$

Based on the generation of the users' intended candidate breakpoints, the proportional coefficients $\{c_i\}$ for every segment of a stroke between each pair of on-curve control points can then be gained from solving the following optimization problem:

$$\begin{cases} Minimize: f(c_i) = e_f^i = \max\left[\left|\frac{(d_{curve}^1 - d_{ink}^1)}{L}\right|, \left|\frac{(d_{curve}^2 - d_{ink}^2)}{L}\right|, \cdots, \left|\frac{(d_{curve}^m - d_{ink}^m)}{L}\right|\right] \\ Subject\ to\ constrains: 2(Pm_{i-1} + Pm_i) + c(\nabla Pm_{i-1} + \nabla Pm_i) = 0, c \geq 0 \end{cases} \tag{9}$$

Equation (9) can be solved easily by traditional optimal approach such as Newton's method. If the value of the fitting error for an optimal solution of equation (9) is greater than the experimental threshold, a new on-curve control points must be generated between the current pair of on-curve control points by bisecting the current segment into two newly sub-segments. The process of solving the optimization problem for each of sub-segments described in equation (9) would then be repeated till the desired fitting error is achieved. A stroke is parameterized after all its segments are approximated by piecewise cubic Bezier curve suitably.

As a result, a sketchy object with *n* number of inputting strokes can be represented as a set of parameters of some pieces of cubic Bezier curve as follow:

$$Sketch \propto \left\{\left(Pm_i^j \mid i=0,1,2,\cdots,k^j\right),\left(c_i^j \mid i=1,2,\cdots,k^j\right),e_f^j\right\},\ j=1,2,\cdots,n \qquad (10)$$

4.4 Curve to Line Approximation

For some applications such as sketch recognition and display of sketch, the curve components should be approximated as a line component instead of curve representation within the allowable error range. Since the line approximation from raw digital ink data is a computationally intensive process, the Bezier curve fitting parameters can be used directly to examine the possibility of a line approximation.

One of the simplest methods of line approximation is to replace the approximated cubic Bezier curve with its three boundaries of the control polygon, which connects orderly all control points. The process is the recursive dichotomy (*u*=1/2 in equation of Bezier curve) of a cubic Bezier curve, and the control points of each segment can be obtained by de Casteljau algorithm till all control points are located within the error tolerance boundary, as shown in the **Fig. 6**. If two conditions are satisfied, i) the Euclidean distances between the on and off control points of each segment are smaller than allowable delta tolerance, and ii) the minimum Euclidean distance between the off-line point and the straight line between the two end (on-curve) control points is smaller than the error tolerance. Then, the off-line control point is assumed to be within the error tolerance. If the test is successful, a piece of Bezier Curve can be represented as three straight lines by connecting orderly each of control points.

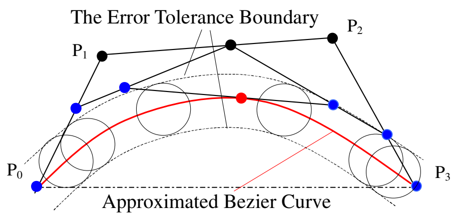


Fig. 6. Illustration of curve to line approximation

5 Experiments and Discussion

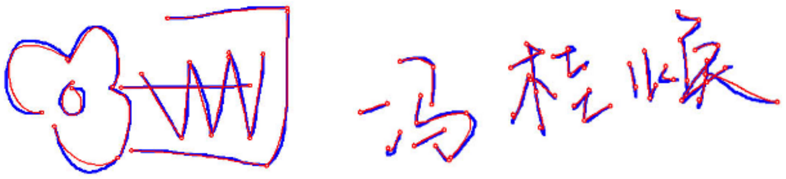
In the experiment, two different types of freehand drawings are designed: graphic drawings and Chinese characters. Some of our experimental results are listed in **Table 1**. The first column is the original inputting sketch, and the second is the fitted sketch where all the segments have been modeled curves and some of the segments between every two breakpoints are subdivided because the error is higher than the threshold. The number of strokes, candidate breakpoints and fragmented segments for each sketch are listed in third, fourth and fifth column respectively. The data listed in sixth column is the maximal among all of the fitting errors to fit for every segment using piecewise Bezier curve in a sketch. The compression ratio is the ratio of file storage of the parameterized sketch to the original ink points. All experiments are done on an Intel PC with a 2.8 GHz CPU and 512MB memory running on Microsoft Windows XP Professional.

Table 1. Some instances of our experiments for sketch parameterization

Input Sketch	Fitted Sketch	Number of Strokes	Pieces of Segments	Max. Fitting Error	Compression Ratio	Computing Cost (ms)
		2	4	0.0802	0.160	0.045
		1	3	0.0510	0.108	0.078
		2	4	0.0756	0.165	0.153
		8	10	0.0714	0.173	0.176
		14	23	0.0975	0.206	0.369
		5	5	0.0478	0.129	0.107
		8	9	0.0783	0.179	0.201
		15	22	0.0856	0.197	0.485

From **Table 1**, we can see that our method of sketch parameterization is both effective and efficient. Firstly, the approximating time for all inputting sketches is less than 0.5 milliseconds. This indicates that our algorithm can provide the efficient-computation for a wide range of freehand drawing graphic objects, and is suitable for online freehand sketches inputting. Secondly, all drawing objects or handwritings in Chinese characters can be parameterized by some piecewise Bezier curves within allowed fitting error (in our experiments, the allowed fitting error is set to 0.1). These connote that sketch parameterization using a piecewise cubic Bezier curve approximation is adaptable for a wide range of freehand drawing graphic objects. It is very useful for many applications where the original of ink path must be hold and transmitted or re-used, such as conceptual expression for product or software design, computer supported cooperative work, message exchange in mobile computing, pervasive computing and ambient intelligence, electric notes/white-board for digital classroom/office, and so on. Thirdly, all freehand drawings can be approximated with more than fifteen percent of data compression. This means that it can reduce the redundancy of on-line freehand drawing data within expected fitting tolerance for graphic data exchange and transmission. This would be very useful especially in low bandwidth wireless networks.

Fig. 7 illustrates the visual approximated effects of our two typical cases for sketch parameterization.



(a) Approximation of drawn symbols (b) Approximation of Chinese signature

Fig. 7. Examples of our experiments for sketch parameterization

6 Conclusions

In this paper, we present a practical solution for efficient representation of hand drawing graphics. The goals of the proposed method are to parameterize and reduce the redundancy of online hand drawing graphic data. A piecewise Bezier curve approximation is implemented in recursive architecture. For the computational efficiency, the cubic Bezier curve representation is only used in our method since it is relatively simple and the curve coefficients can be easily obtained. But the cubic Bezier curves can only represent simple arc shape curves. Thus, complicate shaped strokes are represented by piecewise approximated cubic Bezier curves in our approach. The advantages of our method can be concluded as follows. Firstly, it is suit for the retainable freehand drawing of a wide range of graphic objects without loss of user originals. Secondly, it can reduce the redundancy of on-line freehand drawing data and provide the efficient-computation for graphic data exchange and transmission. Thirdly, it is user-independent because it is dependent only on the allowed fitting error without loss of user intention. The experimental results show both effective and efficient.

Theoretically, sketch parameterization must be done with the minimization of fitting error, the maximization of computing efficiency and the optimization of data compression. However, there is a conflict in sketch parameterization between the fitting tolerance and the computing complexity as well as the data compression ratio. That is to say, the approximated curve can fit better to original ink path by generating more segments for each of strokes in a sketch, however, the run time would be higher and the ratio of data compression would be lower. Therefore, a compromise must be made for sketch parameterization between the minimization of fitting error, the maximization of computing efficiency and the optimization of data compression dependent on the requirements of particular applications. For example, the accurate fitting may be preferential for cooperative carton design over Internet, the data compression must be priority for message exchange and transmission using carton over wireless network, the computing complexity would be the most important in drawing graphic objects with portable devices. Efficient high order curve approximation and optimization idea may be highly desired as a solution in the near future.

Acknowledgement

This paper is supported by the grants from the National Natural Science Foundation of China [Project No. **69903006** and **60373065**] and the Program for New Century Excellent Talents in University of China [Project No. **NCET-04-04605**].

References

1. Zhengxing Sun and Jing Liu, Informal user interfaces for graphical computing, Lecture Notes in Computer Science, Vol. 3784, 2005, pp. 675-682.
2. Rubine Dean, Specifying gestures by example, Computer Graphics, 1991, Vol. 25, pp. 329-337.
3. Fonseca M. J., Pimentel C. and Jorge J. A., CALI-an online scribble recognizer for calligraphic interfaces, In: AAAI Spring Symposium on Sketch Understanding, AAAI Press, 2002, pp. 51-58.
4. Xiaogang Xu, Zhengxing Sun, Binbin Peng, et al, An online composite graphics recognition approach based on matching of spatial relation graphs, International Journal of Document Analysis and Recognition, Vol. 7, No.1, 2004, pp. 44-55.
5. Calhoun C., Stahovich T.F., Kurtoglu, T. et al: Recognizing Multi-Stroke Symbols. In: AAAI Spring Symposium on Sketch Understanding, AAAI Press, 2002, pp. 15-23.
6. Zhengxing Sun, Wenyin Liu, Binbin Peng, et al, User Adaptation for Online Sketchy Shape Recognition, Lecture Notes in Computer Science, Vol. 3088, 2004, pp. 303-314.
7. Sezgin T. M. and Davis R., HMM-Based Efficient Sketch Recognition, Proceedings of the 10th international conference on IUI, Jan., 2005, San Diego, California, USA.
8. Zhengxing Sun, Wei Jiang and Jianyong Sun, Adaptive online multi-stroke sketch recognition based on Hidden Markov model, Lecture Notes in Artificial Intelligences, Springer-Veralg, 2006, to be published.
9. Wenyin Liu, Xiangyu Jin and Zhengxing Sun, Sketch-Based User Interface for Inputting Graphic Objects on Small Screen Device, Lecture Notes in Computer Science, Vol. 2390, 2002, pp. 67-85.
10. Huang Z and Cohen F, Affine-invariant b-spline moments for curve matching. IEEE Transactions on Image Processing. Vol. 5, No.10, 1996, pp. 1473-1480.
11. Raymaekers C., Vansichem, G., and Reeth F. V., Improving sketching by utilizing haptic feedback, In AAAI Spring Symposium on Sketch Understanding, AAAI Press, 2002, pp. 113-117.
12. Park Jaehwa and Kwon Young-Bin, An Efficient Representation of Hand Sketch Graphic Messages Using Recursive Bezier Curve Approximation, Lecture Notes in Computer Science, Vol. 3211, 2004, pp. 392-399.
13. Sezgin T. M., Stahovich T. and Davis R., Sketch based interfaces: early processing for sketch understanding. Proceedings of the 2001 Workshop on PUI, Orlando, Florida, 2001, pp. 1-8.