Visual Comput (2005)
DOI 10.1007/s00371-005-0304-4

ORIGINAL ARTICLE

Hongwei Lin
Wei Chen
Guojin Wang

Curve reconstruction based on an interval B-spline curve

Published online: 14 July 2005

H. Lin · W. Chen (✉) · G. Wang
State Key Laboratory of CAD&CG,
Zhejiang University,
Hangzhou, China, 310027
chenwei@cad.zju.edu.cn

Abstract Curve reconstruction that generates a piece of centric curve from a piece of planar strip-shaped point cloud is a fundamental problem in reverse engineering. In this paper, we present a new curve-reconstruction algorithm based on an interval B-spline curve. The algorithm constructs a rectangle sequence approximating the point cloud using a new data clustering technique, which facilitates the determination of curve order implied in the shape of the point cloud. A quasicentric point sequence and two pieces of boundary point sequences are then computed, based on which a piece of interval B-spline curve representing the geometric shape of the point cloud is constructed. Its centric curve is the final reconstructed curve. The whole algorithm is intuitive, simple, and efficient, as demonstrated by experimental results.

Keywords Point cloud · Curve reconstruction · Interval B-spline curve · Reverse engineering

1 Motivation

In reverse engineering, curve reconstruction, in which attempts are made to reconstruct a piece of a curve in NURBS form from a piece of a planar strip-shaped point cloud, plays an important role in surface reconstruction [5, 15]. Many well-studied methods have been proposed recently. Pottmann and Randrup [10] proposed discretizing the involved plane into a binary image first and calculating its medial axis by a conventional image-thinning algorithm. A parametric curve is then fitted to points lying in the medial axis. Another method, presented by Goshtasby [3], computes a radial basis function (RBF) surface based on a point cloud and discretizes the RBF surface into an image. The reconstructed curve from the point cloud can be computed by tracing the spine of the image. Obviously, the accuracy of resultant curves using the above two methods is dominated by the image resolution.

Levin [6] introduced the moving least squares MLS method. For each point in the point cloud, its neighboring points are fitted by a curve with a weighted regression. The point is then replaced by some point on the curve. The whole procedure is repeated until the point cloud is thin enough to accomplish the curve reconstruction. Note that the reconstruction result is dependent on the size of the selected neighborhood. Lee [5] carefully studied the effect of neighborhood size and proposes an optimal neighborhood-size-selection scheme.

With the aim of estimating the reconstructing curve, Fang et al. [2] proposed computing the fitting curve based on the spring-energy minimization model. Taubin and Ronfard [13] made use of an implicit simplicial curve defined by a triangular mesh and the values at the mesh vertices to reconstruct a curve from an unordered point set. However, both methods are difficult to handle in cases with too much noise.

Generally speaking, the key problem in curve reconstruction is to transform the unordered point cloud into an orderly point set, where the order is implied in the geometric shape of the point cloud. Given that the order is known, curve reconstruction can be achieved by interpolating or fitting an ordered point set with a parametric curve. Based on this investigation, we propose the curve reconstruction algorithm based on an interval B-spline curve (CRIBC). For a piece of a strip-shaped planar point cloud, the plane is first discretized into a uniform grid, whose resolution and step size are determined by the intrinsic property of the point cloud. A sequence of rectangles is then computed to classify the point cloud using a new data clustering technique, which facilitates the determination of the order implied in the point cloud. Subsequently, we calculate a sequence of quasicentric points from the rectangle sequence together with two boundary point sequences. Thereafter, two pieces of B-spline curves fitting two boundary-point sequences are computed, and their control points are connected as interval control points of an interval B-spline curve. The centric curve of the interval B-spline curve is the final reconstructed curve.

Our algorithm makes use of interval B-spline curves introduced in [7, 11, 12, 14] to avoid error accumulation in floating-point operations of curve approximation. They are different from standard Bézier or B-spline curves, as each of their control points varies in a rectangular region. Hence, a piece of planar interval B-spline curve has a strip-like shape. In this paper, a generalized B-spline curve whose control point varies on a line segment is used to envelop the strip-shaped point cloud. Its boundary structure is much simpler than that of an interval B-spline curve.

The rest of this paper is organized as follows: in Sect. 2, an outline of the algorithm is given after introducing some basic concepts. Section 3 describes a new sequence-joining method for the generation of the rectangle and quasicentric point sequences. In Sect. 4, the computation of two boundary point sequences and the interval B-spline curve is presented. Section 5 elaborates upon experimental results. Finally, Sect. 6 summarizes the concepts presented.

2 Related concepts and outline of the algorithm

2.1 Related concepts

To make our explanation clear, we first introduce some related concepts. Suppose that S is a strip-shaped planar point cloud with N points. We can perform Delaunay triangulation [9] on it, yielding a planar triangular mesh M, whose vertices are points in S. Each vertex p of M has several adjacent edges, among which the largest edge length is denoted by $d_p^{\max}$. Accordingly, we define the average sampling radius as follows:

Definition 1. *The average of the largest adjacent edge length of all vertices in M is called the "average sampling radius," denoted by* $r_{\max}$, *i.e.,*

$$r_{\max} = \frac{\sum\limits_{p \in S} d_p^{\max}}{N}. \tag{1}$$

Meanwhile, we can calculate the arbitrarily-oriented minimum-area bounding box of S based on the convex hull of the point cloud as described in [9]. Note that both the average sampling radius and the arbitrarily-oriented minimum-area bounding box are intrinsic properties of a point cloud and are independent of special coordinates.

The arbitrarily-oriented minimum-area bounding box can be further divided into a set of uniform gridpoints whose step size is selected as r_{max}. This grid is called a "global grid." A grid unit containing points is called a "feature unit," and a grid unit without points is called a "non-feature unit".

Definition 2. *A feature unit is called an "inner feature unit" if and only if its four nearest-neighbor grid units are feature units. Otherwise, it is a "boundary feature unit."*

Each grid unit in the global grid is labeled by integer coordinates (i, j). We call a rectangular subregion of the global grid a "subgrid." It can be represented with four integers as shown in Fig. 1:

$$\{imin \le i \le imax, jmin \le j \le jmax\} \tag{2}$$

We call the barycenter of the point set in each subgrid a "quasicentric point" because it is not exactly in the final reconstructed curve. For each quasicentric point P in a subgrid A, we define its global width as shown in Fig. 2. Specifically, a vector T originating from P is first constructed by a rule that will be described later. The line passing P perpendicular to T is denoted by L. All points in the boundary feature units through which L passes are projected to L. The distance between the two outermost projected points is called the global width at P.

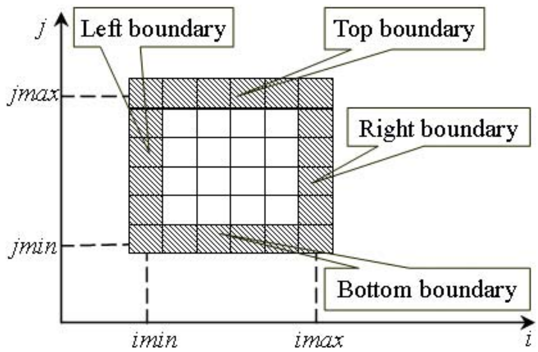

Fig. 1. Subgrid and its four boundaries

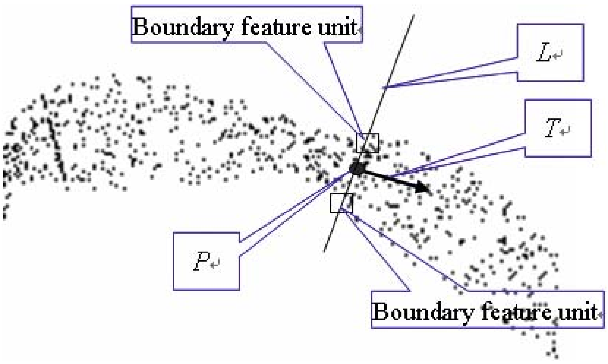

Fig. 2. Global width of quasicentric point P

2.2 Algorithm outline

The curve reconstruction algorithm based on an interval B-spline curve consists of four stages as follows.

Algorithm 1: Curve reconstruction based on an interval B-spline curve

1. Compute the average sampling radius of the point cloud by means of Delaunay triangulation. Compute the arbitrarily-oriented minimum-area bounding box of the point cloud, and discretize the bounding box into a uniform grid.
2. Compute the rectangle sequence using the new data clustering technique mentioned below.
3. Compute the quasicentric point sequence and two pieces of boundary point sequences.
4. Compute the interval B-spline curve enveloping the point cloud. Its centric curve is taken as the reconstructed curve.

We will describe stages 2–4 in detail in the following sections.

3 Sequence-joining to determine order

The goal of sequence-joining is to classify the point cloud S into a set of point clusters. Each cluster is represented by a rectangle subgrid, facilitating the determination of the order implied in the point cloud. Our new method, called the sequence joining method (SJM), is different from the classical joining method in two respects. First, the joining primitives of SJM are feature units instead of points. Second, a reasonable joining scheme is used during the joining procedure taking into account the shape of the point cloud. Figure 3 illustrates one example using SJM where the quasicentric curve is obtained by connecting the quasicentric point sequence.

Basically, SJM is a special-purpose data clustering technique. We will first give a brief introduction to the classical joining method. We then describe our shape-based joining scheme using feature units and subgrids. Thereafter, SJM is described in detail.

3.1 Classical joining method

Conceptually, the classical joining method [4] in cluster analysis takes four stages:

1. Take each point in the point cloud as a cluster;
2. Compute the distance between each pair of clusters, and merge the pair of clusters with minimum distance into one cluster;
3. Repeat step 2 until all points are joined to one cluster;
4. Construct the final clustering by some joining scheme.

We take eleven points shown in Fig. 4 as an example. The first step joins a and b into cluster 1, the second step joins d and e into cluster 2, the fifth step joins a, b, and c into cluster 5, and so on. The final step constructs cluster 10, which contains all 11 points. Thereafter, a predetermined clustering operation is performed according to a number of criteria. For example, if the joining stops at

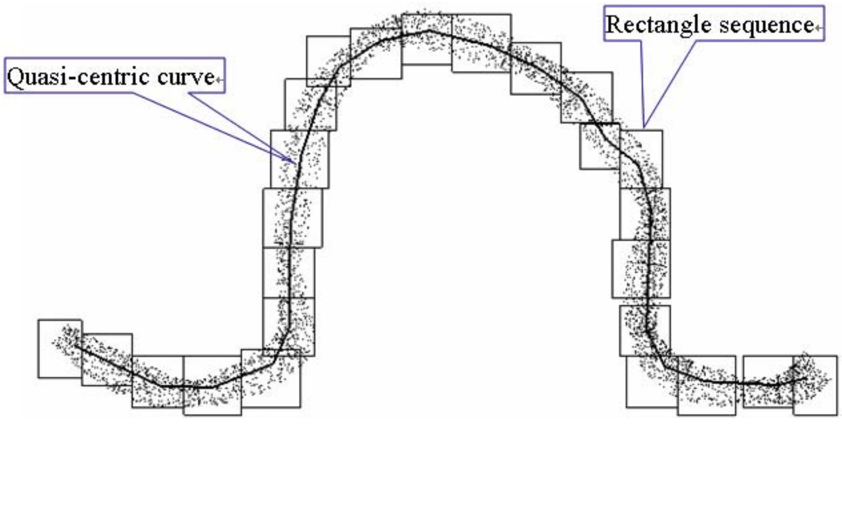

Fig. 3. The rectangle sequence constructed using the sequence-joining method

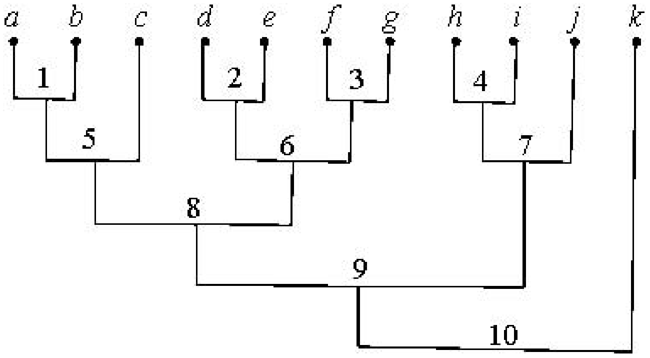

Fig. 4. The clustering scheme for the joining procedure

the eighth step, 11 points can be classified into three clusters; namely, $\{a, b, c, d, e, f, g\}$, $\{h, i, j\}$ and $\{k\}$.

3.2 Shape-based joining scheme

Note that the clustering primitives are feature units or subgrids instead of points. The joining procedure starts with an initial global grid, or say, a set of feature units, and any successive joining operation generates subgrids. Suppose that the subgrid to be processed takes the form $\{imin \leq i \leq imax, jmin \leq j \leq jmax\}$. If its left boundary $\{i = imin, jmin \leq j \leq jmax\}$ contains any unhandled feature unit, the boundary is called an "open boundary." Otherwise, it is called a "closed boundary." Open boundaries need to be extended outwards, i.e., decreasing *imin* by one. On the other hand, there is no point outside of the closed boundary because the side length of any grid unit is the average sampling radius. Thus, the closed boundary remains unchanged. Along the other three boundaries, i.e.,$\{i = imax, jmin \leq j \leq jmax\}$, $\{imin \leq i \leq imax, j = jmin\}$ and $\{imin \leq i \leq imax, j = jmax\}$, similar processes are carried out. We call the procedure of updating the four boundaries of a subgrid a "joining cycle" for the cluster.

Note that the joining procedure is recursive and results in a rectangle sequence, which is an approximation to the point cloud. In order to make the approximation as close as possible, we propose a shape-based joining scheme. A subgrid stops joining and becomes the final cluster if and only if it satisfies one of following conditions: first, two of its opposite boundaries are both closed boundaries. Second, the length of its the longest boundary is greater than the global width at the quasicentric point of the subgrid prior to the generation of the subgrid.

3.3 Sequence-joining method

We now describe the detailed procedure of SJM. First, the feature unit with minimal i and j coordinates is set as the seed unit for joining. Beginning at the seed unit, the first subgrid, denoted by $\{imin \leq i \leq imax, jmin \leq j \leq jmax\}$, can be constructed by the aforementioned shape-based joining scheme. However, the conditions to stop joining should be adjusted slightly here because the current global width cannot be calculated. Consequently, we adopt a different joining criterion, i.e., if two arbitrary boundaries of the underlying subgrid are both closed boundaries, the subgrid stops joining and becomes the first subgrid.

The first subgrid needs to grow further along either the i or j direction. To this end, four neighboring subgrids with the same size as the current subgrid are constructed:

1. $\{imin \leq i \leq imax, jmax{+}1 \leq j \leq jmax + lenj\}$,
2. $\{imin \leq i \leq imax, jmin - lenj \leq j \leq jmin - 1\}$,
3. $\{imin - leni \leq i \leq imin - 1, jmin \leq j \leq jmax\}$,
4. $\{imax{+}1 \leq i \leq imax + leni, jmin \leq j \leq jmax\}$.

Here, $leni = imax - imin + 1$ and $lenj = jmax - jmin + 1$ are the side lengths of the current subgrid along the i and j directions, respectively.

We choose the subgrid that is adjacent to the open boundaries of the current subgrid and has a maximal number of points as the initial state of the next subgrid. In Fig. 5a, *SB* and *SA* are two potential candidates. *SB* is chosen because it has more points than *SA* (see Fig. 5b). The initial state *SB* is further processed as follows: first, the boundaries that do not contain unhandled feature units are moved incrementally inward until the resultant boundaries are all open boundaries (see Fig. 5c). Second, suppose the length of the boundary of the resultant subgrid SB', which is adjacent to *SC*, is l_1, and the other side length is l_2. We fix the boundary adjacent to *SC* and translate its opposite boundary along the direction to the fixed boundary by $\lceil l_2/2 \rceil$. The other two boundaries of SB' are translated toward each other by $\lceil l_1/4 \rceil$ (see Fig. 5d). Here, the operator $\lceil\ \rceil$ is used to compute the floor integer of a floating point. The adjusted subgrid $SB*$ is then set as the new seed, and the second subgrid can be generated using the same method as that used for the first subgrid.

By connecting two quasicentric points of the first and second subgrids, i.e., P_1 and P_2, and defining $\boldsymbol{T} = \overrightarrow{P_1 P_2}$, we get the new global width W_2 at P_2. Choosing a seed according to the above procedure, a third subgrid can be constructed as described in the 3.2. Similarly, we can compute W_3 and the fourth subgrid, W_4 and the fifth subgrid, and so on. This joining procedure continues until the current subgrid contains four closed boundaries. Note that if the last subgrid contains points that belong to the first subgrid, we state that the point cloud is in closed form. Otherwise, we take another joining procedure from the first subgrid along the other direction until the final rectangular subgrid sequence is achieved.

The resultant sequence needs to be adjusted slightly. Because the condition to stop joining for the first and second subgrids is quite simple, they do not represent the real shape of the segment of the point cloud that they cover. We replace them with new subgrids that are grown from one of their neighboring subgrids (see Fig. 6).

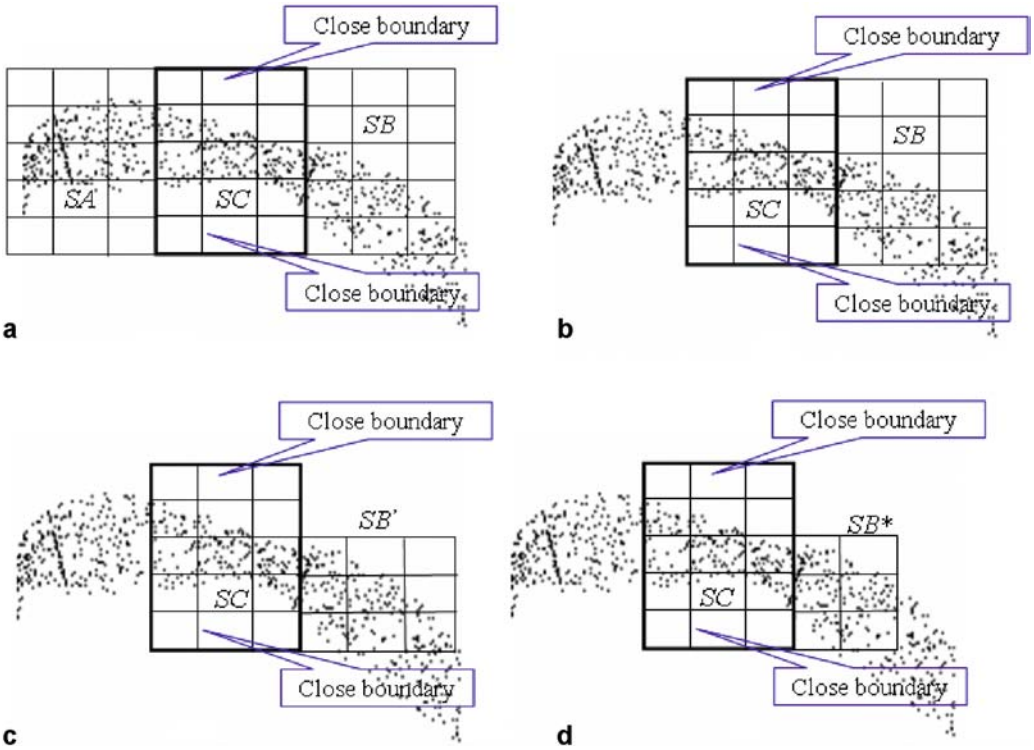

Fig. 5. Determination of the joining direction. **a** The subgrid *SC* is the current subgrid; *SA* and *SB* are the potential initial states of the next subgrid. **b** The subgrid *SB* contains a maximal number of points and is chosen as the initial state of the next subgrid. **c** *SB* is adjusted to *SB*′. **d** *SB*′ is further adjusted to *SB**, which is set as the seed of the next subgrid

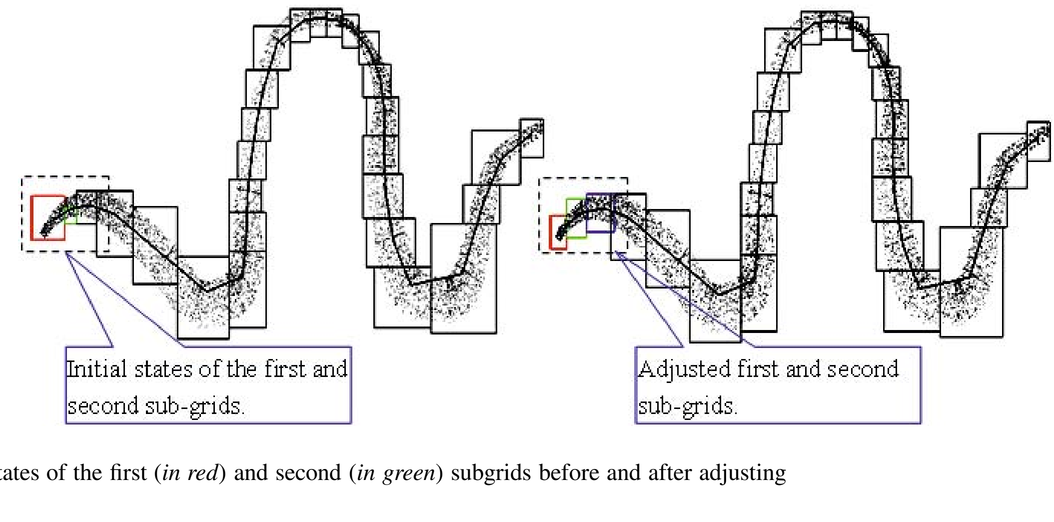

Fig. 6. The states of the first (*in red*) and second (*in green*) subgrids before and after adjusting

Finally, we get a well-shaped subgrid sequence covering the point cloud, whose elements are rectangles. The barycenters of the point sets in these rectangles construct a quasicentric point sequence of the point cloud. And the quasicentric curve of the point cloud can be computed by connecting these quasicentric points in an orderly fashion, as shown in Fig. 3. In summary, we present the recursive SJM algorithm in a pseudo-C language, where *e* is the subgrid employed in the joining, and *width* is the global width at a previous quasicentric point.

Algorithm 2: Recursive sequence joining method

```
void SequenceJoining(SubGrid *e, double width)
{
    double maxlen = the length of the longest
                    boundary of e;
```

if (*maxlen* $>$ *width* OR two opposite boundaries of e are closed boundaries)
    return;
if (the boundary $\{i = e \rightarrow imax, e \rightarrow jmin \leq j \leq e \rightarrow jmax\}$ is an open boundary)
    $e \rightarrow imax = e \rightarrow imax + 1$;
if (the boundary $\{i = e \rightarrow imin, e \rightarrow jmin \leq j \leq e \rightarrow jmax\}$ is an open boundary)
    $e \rightarrow imin = e \rightarrow imin - 1$;
if (the boundary $\{e \rightarrow imin \leq i \leq e \rightarrow imax, j = e \rightarrow jmax\}$ is an open boundary)
    $e \rightarrow jmax = e \rightarrow jmax + 1$;
if (the boundary $\{e \rightarrow imin \leq i \leq e \rightarrow imax, j = e \rightarrow jmin\}$ is an open boundary)
    $e \rightarrow jmin = e \rightarrow jmin - 1$;
SequenceJoining(e, *width*);
}

4 Computation of an interval B-spline curve

Using the method proposed in [14], the rectangle sequence can be interpolated or approximated by an interval B-spline curve. However, the resultant interval B-spline curve cannot envelop the point cloud tightly because the approximation of the rectangle sequence to the point cloud is quite rough. In order to represent the geometric shape of the strip-shaped point cloud correctly, we propose a new approach to compute the interval B-spline curve. Based on the quasicentric point sequence computed in Sect. 3, we first compute two pieces of the boundary point sequences of the point cloud. We then fit them with two pieces of B-spline curves with the same order. Corresponding control points of the two B-spline curves are connected by lines. Taking the connected lines as interval control points, an interval B-spline curve tightly enveloping the point cloud is determined.

The computing of the boundary point is similar to that of the global width at a quasicentric point (see Fig. 2). Given a quasicentric point, we assign a vector to the point and construct a line perpendicular to the vector. All points in the boundary units through which the line passes are projected onto the line. The two outermost projected points are regarded as two boundary points. The assigned vector to each quasicentric point is determined in two ways. If the point cloud is open, backward and forward difference vectors are used for the first and last points in the quasicentric point sequence. For other points, the central difference vectors are adopted. If the point cloud is closed, the central difference vector is taken for each point.

At this point, we have a sequence of triples, each of which consists of one quasicentric point and two boundary points. To construct two pieces of boundary sequences, the correspondence amongst them can be determined one by one as follows:

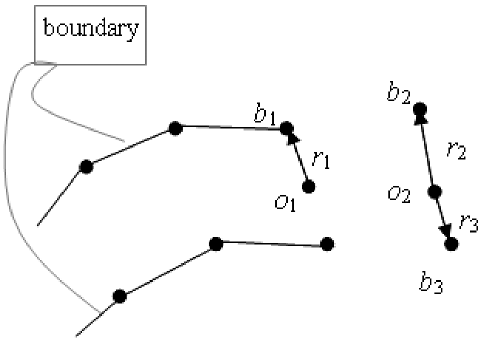

Fig. 7. Illustrations of the boundary points

- Take a piece of known partial boundary as shown in Fig. 7. Suppose b_1 is its last point, which corresponds to the quasicentric point o_1, and b_2, b_3 are two new boundary points corresponding to the quasicentric point o_2.
- Construct $\boldsymbol{r}_1 = \overrightarrow{o_1 b_1}$, $\boldsymbol{r}_2 = \overrightarrow{o_2 b_2}$, and $\boldsymbol{r}_3 = \overrightarrow{o_2 b_3}$, and normalize them.
- If $\boldsymbol{r}_1 \cdot \boldsymbol{r}_2 > \boldsymbol{r}_1 \cdot \boldsymbol{r}_3$, the angle between $\boldsymbol{r}_1$ and $\boldsymbol{r}_2$ is less than the angle between $\boldsymbol{r}_1$ and $\boldsymbol{r}_3$; hence, b_1 and b_2 belong to the same boundary while b_3 lies in the other boundary.

Subsequently, we fit two constructed boundary point sequences with two iterative B-spline curves, as described in [1, 8]. Because both have the same number of points, the number of control points of the two fitting B-spline curves is the same. Connecting two corresponding control points with separate lines and taking these lines as interval control points, we get an interval B-spline curve enveloping the point cloud. According to [7], the boundaries of the interval B-spline curve are two B-spline curves fitting the two boundary point sequences. The centric curve of the interval B-spline curve is the final reconstructed curve.

5 Experimental results

The proposed algorithm was implemented on a PC platform with Microsoft Visual C++ 6.0. All testing point clouds were obtained using the method presented in [5]. For a given planar curve $\boldsymbol{C}(t)$, points are sampled randomly in the region embedded by two offset curves, $\boldsymbol{C}(t) + \boldsymbol{N}(t)r(t)$ and $\boldsymbol{C}(t) - \boldsymbol{N}(t)r(t)$, where $r(t)$ denotes a variable sampling radius, and $\boldsymbol{N}(t)$ is the unit normal vector of $\boldsymbol{C}(t)$. We performed several experiments to demonstrate the capabilities of our CRIBC algorithm for point clouds with a non-uniformly sampling radius, dif-

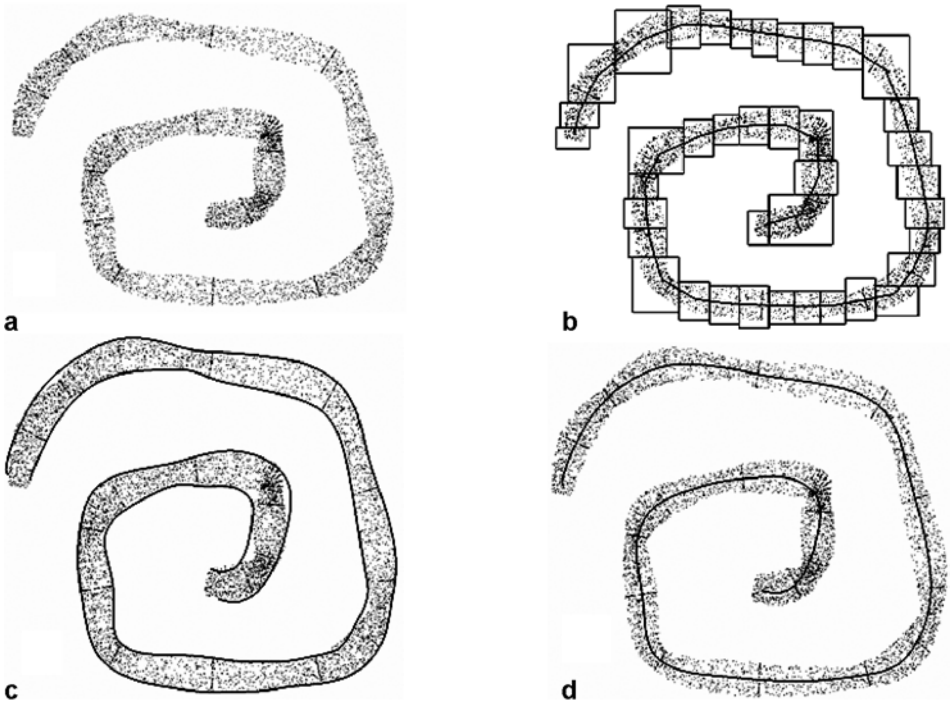

Fig. 8. a A helix-shaped point cloud with comparatively uniformly varying sampling radius. **b** The rectangle sequence and the quasicentric curve. **c** The two boundaries of the interval B-spline curve. **d** The reconstructed curve

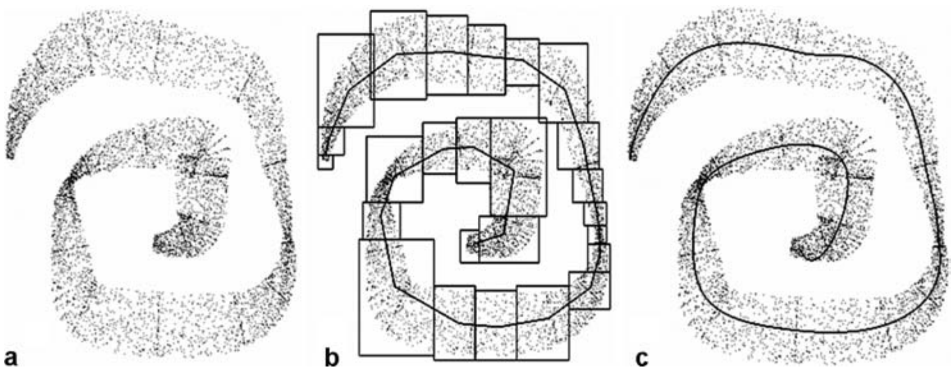

Fig. 9. a A helix-shaped point cloud with non-uniformly varying sampling radius. **b** The rectangle sequence and the quasicentric curve. **c** The reconstructed curve

ferent sampling densities, or sections approaching each other closely. These circumstances are intractable with traditional curve-reconstruction algorithms.

The top left image of Fig. 8 shows a helix-shaped point cloud whose sampling radius varies comparatively uniformly. The rectangle sequence covering the point cloud and the quasicentric curve, the two boundaries of the interval B-spline curve, and the reconstructed curve are illustrated in other images of Fig. 8, respectively.

The left image in Fig. 9 demonstrates another helix-shaped point cloud sampled from the same curve as that sampled for Fig. 8. The difference lies in that its sampling radius varies quite non-uniformly. Our algorithm constructs a sequence of different-sized rectangles along with a quasicentric curve as shown in the middle image. The final reconstructed curve (the right image) represents almost the same shape as that of Fig. 8.

Figure 10 shows another example of a point cloud with very non-uniform sampling radius.

In Fig. 11, a point cloud with different sampling densities is handled. Our algorithm reconstructs desirable result no matter how variable the sampling density is.

Figure 12 demonstrates a complex example in which a point cloud is sampled from a closed curve with non-

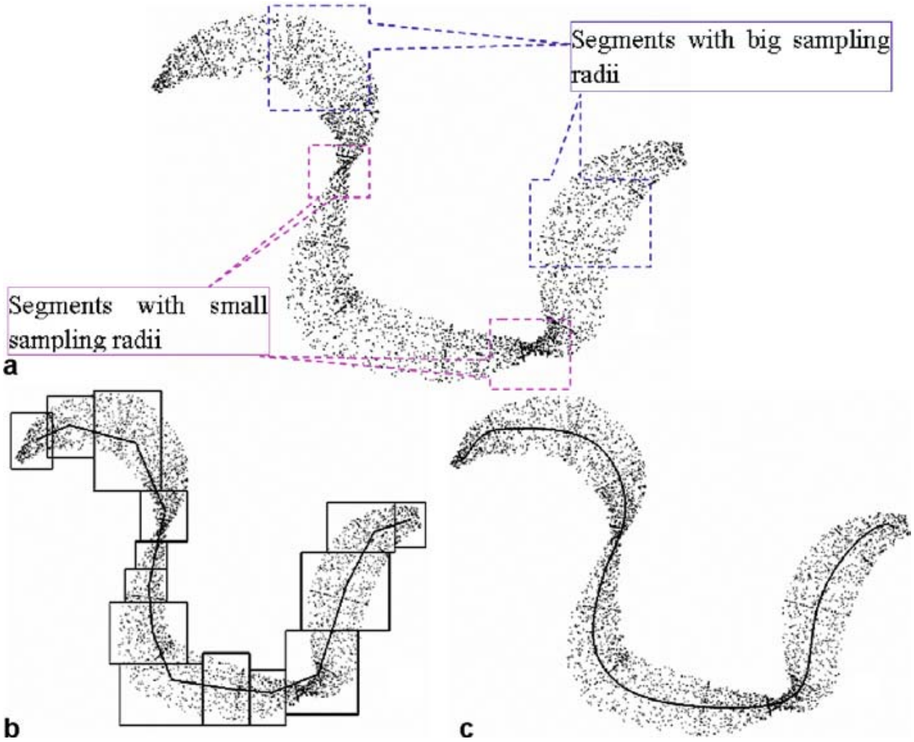

Fig. 10. a A point cloud with non-uniformly sampling radius. **b** The rectangle sequence and the quasicentric curve. **c** The reconstructed curve

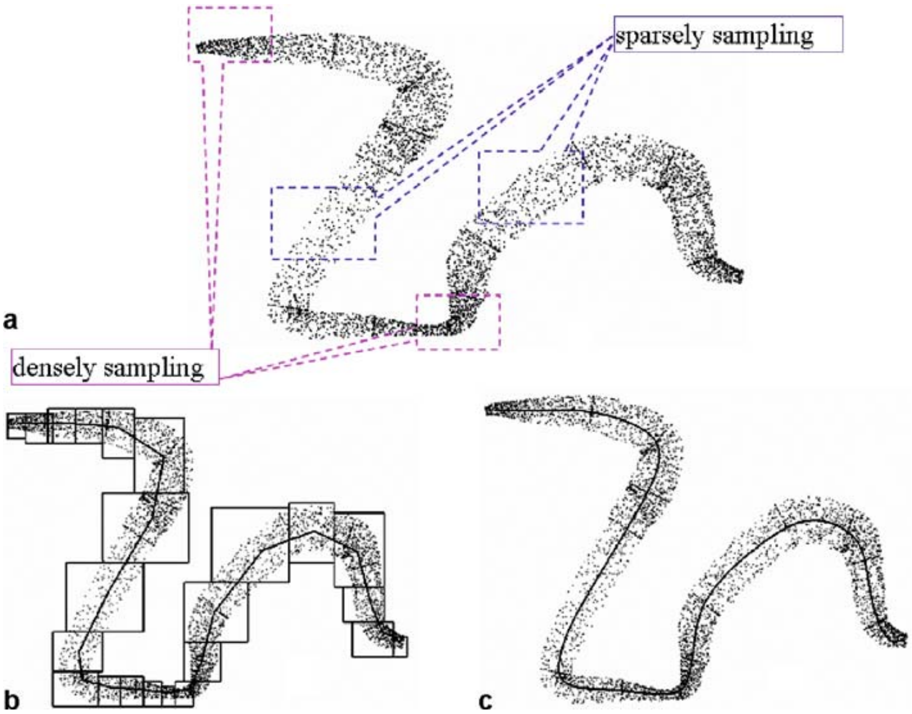

Fig. 11. a A point cloud with different sampling densities. **b** The rectangle sequence and the quasicentric curve. **c** The reconstructed curve

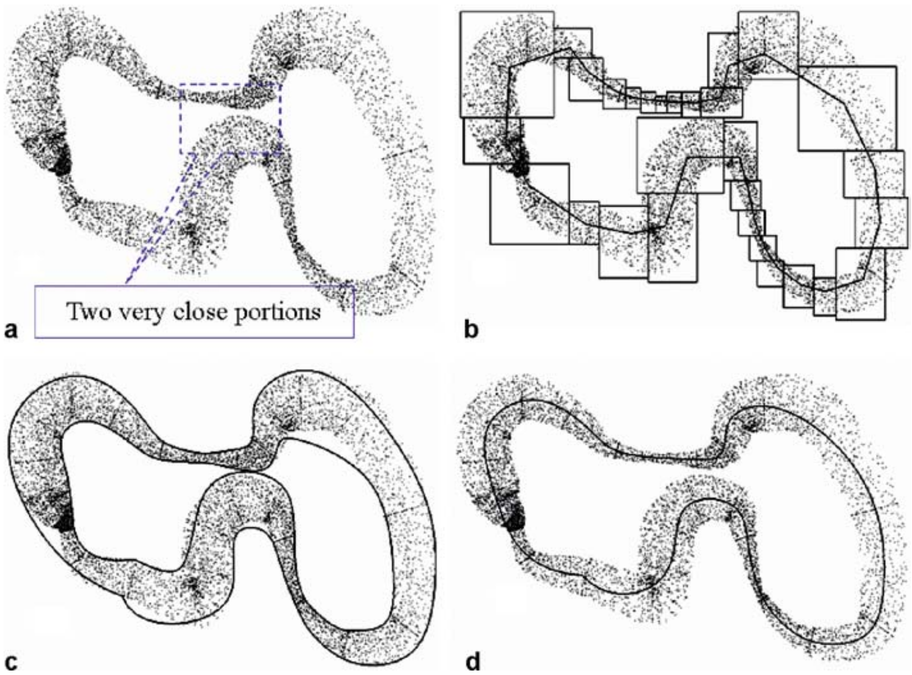

Fig. 12. a A close point cloud with non-uniformly sampling radius, different sampling densities, and two portions approaching each other closely. **b** The rectangle sequence and the quasicentric curve. **c** The two boundaries of interval B-spline curve. **d** The reconstructed curve

uniform sampling radii and different sampling densities. In addition, there are two portions that are very close to each other. The resultant curve presents a satisfactory shape.

6 Conclusions

In this paper, we presented a new robust curve-reconstruction algorithm based on an interval B-spline curve. First, we proposed the SJM to cluster the point cloud into a rectangle sequence, which facilitates the determination of the curve order implied by the geometric shape of the point cloud. Second, we described how to compute two boundary point sequences using the quasicentric point sequence. Third, we presented a sort of generalized interval B-spline curve whose boundaries are two pieces of B-spline curves. By fitting two pieces of boundary point sequences with two iterative B-spline curves, we obtained an interval B-spline curve enveloping the strip-shaped point cloud. The interval B-spline curve filters the noises of the point cloud, and its centric curve is the final reconstructed curve. The principle of using an interval B-spline curve is intuitive, and its computation is simple as demonstrated by our experimental results. We believe that the proposed method will have wide applications in reverse engineering and related fields.

Acknowledgement This work is supported by the 973 Project of China (No. 2004CB719400), National Natural Science Foundation for Innovative Research Groups (No. 60021201), and the National Natural Science Foundation of China (No. 60373033, No. 60333010).

References

1. De Boor C (1979) How does Agee's smoothing method work? In: Proceedings of the 1979 Army Numerical Analysis and Computers Conference, ARO Report 79-3, Army Research Office, pp 299–302
2. Fang L, Gossard DC (1995) Multidimensional curve fitting to unorganized data points by nonlinear minimization. Comput Aided Des 27(1):48–58
3. Goshtasby AA (2000) Grouping and parameterizing irregularly spaced points for curve fitting. ACM Trans Graph 19(3):185–203
4. Jain AK, Murty MN, Flynn PJ (1999) Data clustering: a review. ACM Comput Surveys 31(3):264–323
5. Lee I K (2000) Curve reconstruction from unorganized points. Comput Aided Geom Des 17:161–177
6. Levin D (1998) The approximation power of moving least-squares. Math Comput 67:1517–1531
7. Lin H, Liu L, Wang G (2002) Boundary evaluation for interval Bézier curve. Comput Aided Des 34(9):637–646
8. Lin H, Wang G, Dong C (2004) Constructing iterative non-uniform B-spline curve and surface to fit data points. Science in China Series F 47:315–331

9. O'Rouke J (1994) Computational geometry in C. Cambridge University Press, Cambridge
10. Pottmann H, Randrup T (1998) Rotational and helical surface approximation for reverse engineering. Computing 60:307–322
11. Sederberg TW, Farouki RT (1992) Approximation by interval Bézier curves. IEEE Comput Graph Appl 15(2):87–95
12. Shen G, Patrikalakis NM (1998) Numerical and geometric properties of interval B-Splines. Int J Shape Model 4(1/2):35–62
13. Taubin G, Rondfard R (1996) Implicit simplicial models for adaptive curve reconstruction. IEEE Trans Pattern Anal Mach Intell 18(3):321–325
14. Tuohy ST, Maekawa T, Patrikalakis NM (1997) Approximation of measured data with interval B-splines. Comput Aided Des 29(11):791–799
15. Várady T, Martin RR, Cox J (1997) Reverse engineering of geometric models – an introduction. Comput Aided Des 29(4):255–268

DR. HONGWEI LIN is now working at the State Key Laboratory of CAD&CG, Zhejiang University, China. He received his BSc from the Department of Applied Mathematics at Zhejiang University in 1996, and his PhD from the Department of Mathematics at Zhejiang University in 2004. He worked as a communication engineer from 1996 to 1999. His current research interests include computer-aided geometric design, computer graphics, and image processing.

DR. WEI CHEN is an associate professor at the State Key Lab of CAD&CG at Zhejiang University, P.R. China. He received his PhD degree in 2002 from the Department of Applied Mathematics of Zhejiang University. He has performed research in volume rendering and related technical areas for the past three years. His current interests include hardware-accelerated visualization, photo-realistic rendering, bio-medical imaging, and digital geometry processing.

GUOJIN WANG was born in Shanghai, China in October, 1944. He is now a professor and supervisor of the doctoral program at the Institute of Computer Images and Graphics at Zhejiang University, China. His research interests include computer-aided geometric design and computer graphics. He is a commissary of the Teaching-Guidance Committee of Engineering Mathematics for Chinese Universities under the National Education Committee of China. He is also the principal of the Applied Mathematics Subject, which is one of the State Key Subjects.