HOSTED BY

ELSEVIER

Available online at www.sciencedirect.com

ScienceDirect

Journal of Computational Design and Engineering 3 (2016) 14–23

www.elsevier.com/locate/jcde

A graph-based method for fitting planar B-spline curves with intersections

Pengbo Bo*, Gongning Luo, Kuanquan Wang

School of Computer Science and Technology, Harbin Institute of Technology, China

Received 12 March 2015; received in revised form 8 May 2015; accepted 11 May 2015
Available online 10 June 2015

Abstract

The problem of fitting B-spline curves to planar point clouds is studied in this paper. A novel method is proposed to deal with the most challenging case where multiple intersecting curves or curves with self-intersection are necessary for shape representation. A method based on Delauney Triangulation of data points is developed to identify connected components which is also capable of removing outliers. A skeleton representation is utilized to represent the topological structure which is further used to create a weighted graph for deciding the merging of curve segments. Different to existing approaches which utilize local shape information near intersections, our method considers shape characteristics of curve segments in a larger scope and is thus capable of giving more satisfactory results. By fitting each group of data points with a B-spline curve, we solve the problems of curve structure reconstruction from point clouds, as well as the vectorization of simple line drawing images by drawing lines reconstruction.

Keywords: Curve fitting; Curve reconstruction; B-spline; Point cloud

1. Introduction

2D shapes represented by unorganized data points are frequently encountered in numerous applications, where data points are obtained using scanning devices or extracted from digital images. Unorganized data points (point clouds) are however not a suitable representation for geometric processing. Therefore, converting point clouds into parametric representations such as polylines or B-spline curves is highly demanded in many applications.

In this paper, we study the problem of computing a set of disjoint parametric curves with possible intersections from a point cloud. This problem of *curve extraction* has numerous applications in reverse engineering, where B-spline parametric curves/surfaces are required. Extracting shapes from data points as spline curves also enjoys important applications in digital image processing if the content of an image is a 2D shape composed of curves, such as line drawing images, blueprints and hand written characters. Converting digital images of line drawings into B-spline curves is also a special instance of image vectorization.

A reasonable scheme to this problem is computing meaningful groups of data points together with polyline curves. The latter can be refined by a curve fitting approach. The key point however is to identify combination of curve segments joining at intersections to form continuous curves passing through each other. The main difficulties include (1) the intersection region often contains much noise and thus may give disturbing information for recovery of intersecting curves and (2) tangential lines of joining curves near intersection regions are not sufficient for determining curve joinings. Refer to Fig. 1(a) for an intersection part of a noisy point cloud; Fig. 1 (b) shows three joining curve segments and estimated tangent lines. We see that it is hard to give correct merging of curves using only tangent lines.

In this paper, we propose a framework for fitting B-spline curves to a point cloud, where the curves may intersect with each other and curves with self-intersection are also allowed. The proposed approach consists of two phases. The first phase divides the point cloud into a set of groups of data points where each group of data points represents a curve shape.

*Corresponding author.
E-mail address: pengbo@hitwh.edu.cn (P. Bo).

http://dx.doi.org/10.1016/j.jcde.2015.05.001

This step recovers the topology information of data points. The second step reconstructs a B-spline curve fitting to each group of data points. This step is a geometry recovery step. The main contributions of this paper include

- A unified framework for handling noise, outliers and curve components of a point cloud.
- A graph based method to identify pairs of curve segments which can be merged into a single curve.

2. Related work

2.1. Curve reconstruction

Curve reconstruction from point clouds with or without noise is a well studied problem. Various aspects of curve reconstruction have been addressed including robustness to noise, handling of outliers and feature preservation. Numerous techniques have been employed such as Voronoi diagram, spectral analysis, image processing and optimal transport. Optimal transport is used to reconstruct 2D shapes with polyline structures which performs well for data points with shape features and large noise [1]. A large body of methods makes use of voronoi diagram to deal with curve reconstruction from a set of sampling data points. The advantage of this class of methods is that their accuracy can be proved if an appropriate sampling density is satisfied [2]. Amenta et al. proposed the crust method which utilized the β skeleton and voronoi diagram [3,4]. Wang et al. proposed a curve reconstruction method based on circular neighboring projection and normal-based smoothing [5]. However, these methods do not give a segmentation of data points. The reconstruction result is a structure of curves instead of independent curves.

Some recent works discuss recovering multiple curves or curves with self-intersections from point clouds [6–8]. Ardeshir proposed a method for grouping and fitting multiple curves to point clouds with relatively simple shape [9]. Furferi et al. presented a method for fitting weighted B-spline curves using PCA analysis [10]. Yan et al. proposed a method for curve fitting based on the fuzzy C-means clustering method [11]. Zhao et al. presented a method for fitting non-simple curves using skeleton extraction and refinement [12]. Our proposed method is different to these methods in that we make use of a skeleton representation of point cloud which provides more information on curve shape and topological relationship of curves.

2.2. Curve fitting

Curve fitting mostly refers to the problem of parametric curve approximation to noisy data points. Most existing works assume that the data points come from a single curve without self-intersections. A fitting process starts with an initial curve which is updated by minimizing some objective function measuring fitting quality and curve fairness.

Existing works discuss this problem from various aspects. A parametric curve is often used for shape reconstruction of a point cloud. Levin proposed the moving least squares method (LMS) for curve fitting [13]. Lee discussed the deficiencies of direct application of MLS in curve fitting and proposed some improvements by introducing the Minimal Spanning Tree (MST) in a pre-processing phase [14]. Some works on curve fitting focus on the fitting speed of minimizing squared

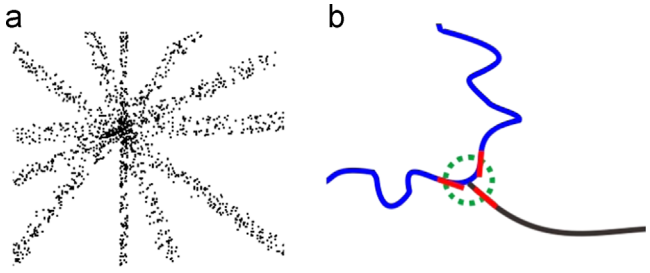

Fig. 1. (a) There is large noise at intersection regions of crossing curves. (b) Tangent vectors of meeting curves are insufficient to decide their role as curve parts of large curves.

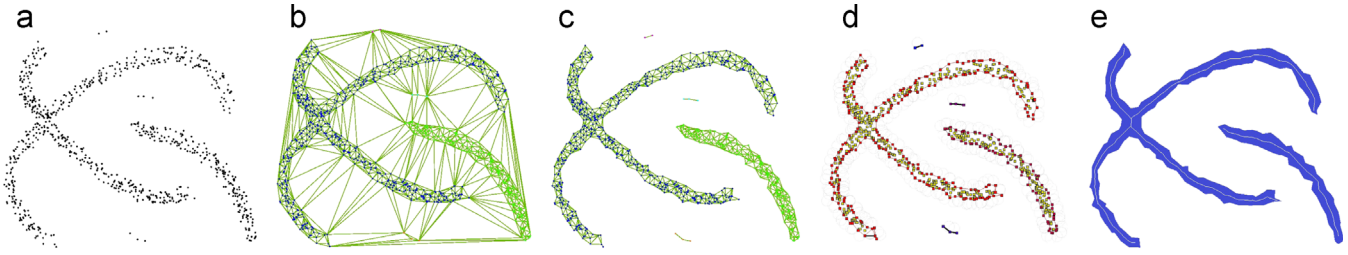

Fig. 2. A Delauney triangulation based method for finding curve components and removing outliers. The data points lie in a bounding box of size 650 by 450. (a) Data points. (b) Delauney triangulation of data points. (c) Mesh after removing long edges e where $length(e) > 23$. (d) The α-shape of data points. (e) Image generated by filling remaining triangles after deleting outliers. An skeleton is also shown which is obtained by applying image thinning algorithm and removing spurious tails. Note two vertices in the skeleton which are very close to each other will be merged into a single vertex.

orthogonal distance. This category of methods includes PDM (point distance minimization) method, TDM (tangent distance minimization) method and SDM (Squared distance minimization) method [15,16]. These methods are inherently traditional optimization methods and a good initialization is needed. These methods cannot be applied directly for fitting multiple curves to a point cloud of complex shape.

2.3. 2D skeletonization

Skeleton extraction is closely related to curve reconstruction which is an important process in many applications such as hand-written character recognition and digital map recognition [17,18]. Image thinning algorithms are widely used for skeleton extraction of 2D shapes, which recursively remove boundary pixels to give an image with one-pixel width [19]. The remaining pixels can be easily converted into polygonal curves by connecting incident pixels. Medial axis, one of the most dominant skeleton representation of shapes, is defined by the set of points which has equal distances to multiple points on the 2D shape [20–22]. Since medial axis is sensitive to noise, a lot of works study robust generation of clean medial axis. There are also skeleton extraction methods making use of Delaunay triangulations of some sampling points of the shape [23].

A topology driven approach is presented for recovering lines in a clean line image for image vectorization which deals with possible combinations of curve segments meeting at a cross region [24]. In hand-written trajectory tracking, the key problem is how to deal with the segmentation concerning the intersection of trajectories. An approach of stroke extraction based on a selective searching technique is presented in [25] where angle information around the intersection region is used and distortions near intersections are ignored. Different to existing methods on skeleton segmentation, our method considers shape goodness of overall curves and thus is able to give more pleasing results.

3. Structure extraction of point clouds

Given a point cloud $S = \{s_i\}$, we process it to extract a set of polygonal curves $\{L_i\}$. The proposed algorithm consists of the following steps:

(1) decompose S into disjoint groups S_i. Outliers are removed at the same time. Each group is represented by a triangulation T (Section 3.1),
(2) compute a skeleton representation K of T (Section 3.2),
(3) extract a set of curve segment $\{Z_i\}$ excluding intersection regions (Section 3.2),
(4) pairing curve segments meeting at intersections and get a set of polylines $\{L_i\}$ (Section 3.3).

3.1. Component recognition

The first step decomposes a point cloud S into separate subgroups S_i of data points. We propose a method which jointly handle noise, outliers and disjoint curves, based on the Delauney Triangulation of data points. The set of data points S is firstly converted into a triangulation τ using Delauney triangulation.

Our method is inspired by the concept of α-shape. It is known that α-shape, as a subset of the Delauney triangulation of S, is a reasonable way to represent the shape of S. The boundary of the α-shape of S is a set of line segments Γ connecting data points. Edge $\overline{p_ip_j} \in \Gamma$ if there exists a disk of radius α touching both p_i and p_j which does not contain any other data point in S. Such a neighboring pair of data points p_i,p_j is called α-repulsive. It is easy to observe the following fact of α-shape.

- Points p_i,p_j are not α-repulsive if $\|p_i - p_j\| > 2\alpha$ (Property 1).

Therefore, an edge in τ whose length is larger than 2α does not contribute to the boundary of the α-shape of S. Moreover, the following property of Delauney triangulation is well-known.

- In a Delauney triangulation of a planar point set, there exists a connecting edge between p_i and p_j if p_j is the nearest data point to p_i (Property 2).

From property 1 and property 2, we can draw a conclusion that edge length in τ provides important information about the relationship of data points. Removing long edges in τ is effective for filtering out outliers in a noisy point cloud as well as for separating data points into disjoint curve components where each component contains a single curve or a set of intersecting curves.

After deleting those edges in τ whose length is larger than a value ρ related to sampling density of point clouds, we get a triangulation representation which may contain several disconnected components. Small isolated pieces are regarded as outliers and are deleted. Each remaining component will be represented by a single curve or a set of intersecting curves. In the following discussions, we denote each component by T

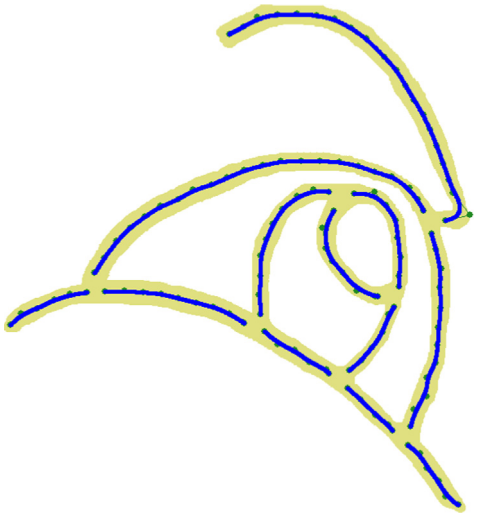

Fig. 3. Removing intersection regions decomposes the data points into segmental groups; each group of data points is fitted with a B-spline curve.

and its associating data points S. For further processing, we generate a back-white image I for S by filling each triangle in T with black color. Refer to Fig. 2 for the pipeline of curve components recognition.

The threshold ρ is important for accurate separation of data points. If ρ is too large, over segmentation might happen; if ρ is too small, originally disjoint curves might be recognized as a single curve. Both cases should be carefully avoided. Clearly, the selection of ρ should depend on the sampling density of data points.

3.2. Curve segment reconstruction

The next step deals with each component T to achieve disjoint curves. This task is non-trivial due to the noise near

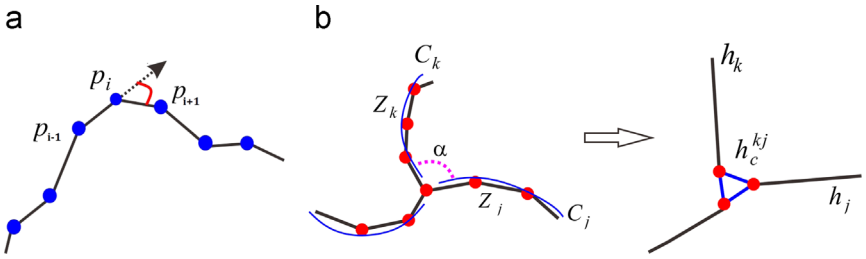

Fig. 4. Defining cost value for graph edges. (a) Turning angle of two edges in sequence. (b) A complete graph is added for each intersection vertex. The cost value of an connection edge is defined by a weighted combination of turning angle and curve energy.

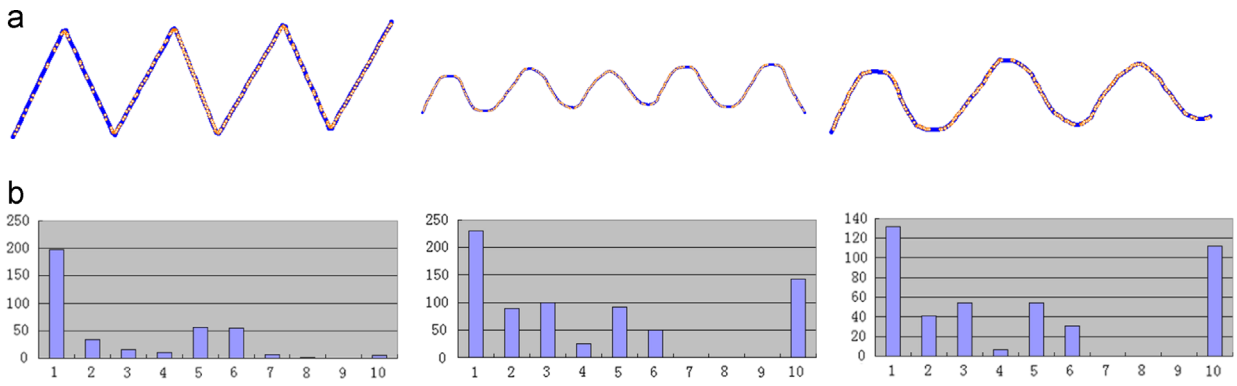

Fig. 5. Some drawing lines are shown together with curvature histograms.

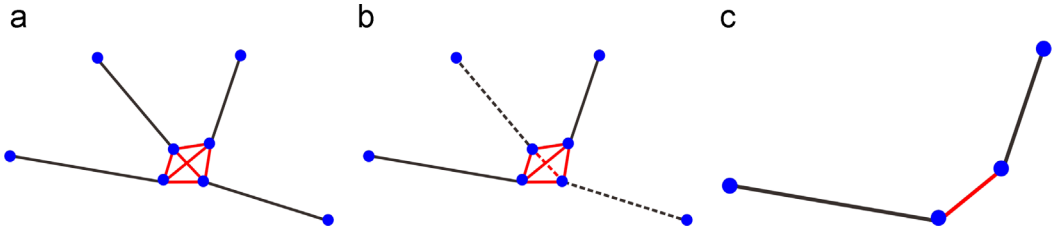

Fig. 6. Algorithm for curve extraction. (a) is a subgraph corresponding to a cross region. In (b), one path with minimal energy is found which is denoted by dashed line. After removing those nodes in the extracted path and dangling edges, we get a remaining graph in (c).

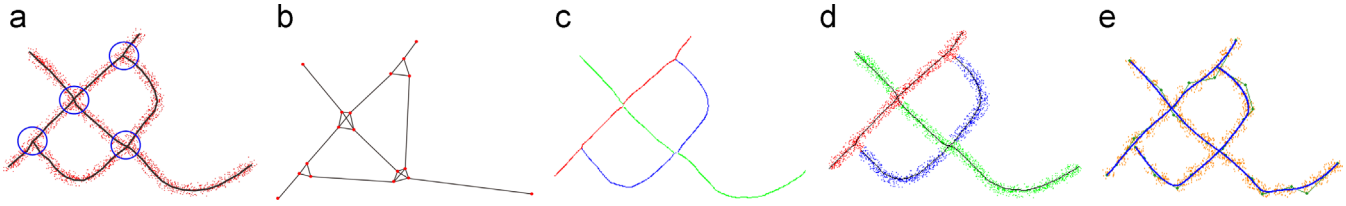

Fig. 7. The pipeline of data points segmentation and curve extraction. (a) A point cloud. The intersection regions are identified using intersection vertices of the skeleton. (b) A graph is created from the skeleton where intersection vertex of valence n is replaced by a complete graph K_n. (c) Our curve extraction algorithm gives the segmentation of the skeleton. (d) The grouping of data points using the skeleton segmentation results. (e) Fitting B-spine curves.

intersection regions as well as shape ambiguities of point clouds. For the purpose of intersecting curves reconstruction, a skeleton representation K of T is computed. This is achieved by applying an image thinning algorithm to I [19]. Due to the noise in point cloud, the obtained skeleton may contain spurious tails and crossing necks. A skeleton cleaning step is applied to remove spurious tails and merge crossing necks to get a clean and accurate skeleton representation. This skeleton cleaning problem has been well studied. Refer to [17] for more details. Refer to Fig. 2(e) for a skeleton obtained by image thinning operations.

Curve segments represented by data points are easy to detect, except for data points near intersection regions. We therefore remove those data points in local regions of intersections temporarily and return an apart set of data points D_i. The size of an intersection region is defined by a circle $X(c, r)$ where c denotes the circle center, r is circle radius. Each vertex v whose valence is larger than 2 in K is identified as a center c. The position of v is roughly where several curves intersect or self-intersections occur. r is set to 1.2 r where r is the distance from c to the closet boundary vertex of T.

Each set of data points D_i is assigned with a polyline Z_i which is easily obtained by disconnecting K at intersection vertices. Unfortunately, the shape of Z_i is zigzag and does not express curve shape well. We define a B-spline curve C_i whose control points are positions of the sequential vertices of Z_i. With C_i as an initial curve, a curve fitting procedure is applied to improve its approximation quality as well as shape fairness. The associating target data points D_i are easily found which are vertices of T. Z_i is then refined by projecting its vertices to C_i. See Fig. 3 for an example.

3.3. Pairing curve segments

Now we need to decide pairing of curves meeting at intersection regions which actually come from the same curve. We propose a graph based method to facilitate connection decision of meeting curves.

3.3.1. Connection graph construction

A graph G is built in the following way. Each polygonal curve segment Z_i has an associated edge h_c^i in G which is called a *regular edge*. Further, in order to encode all cases of curve segment combination at an intersection region, a complete graph K_n is added into G for an intersection vertex of valence n in K. In this way, a *connection edge* exists for each pair of curve segments joining at the same vertex (refer to Figs. 4(b) and 7(b)).

We then assign a weight to every edge in G to make a weighted graph. The weight of each connection edge measures the quality of merging two curve segments Z_k and Z_j. Two issues should be considered (1) shape similarity of Z_k and Z_j; (2) the turning angle between Z_k and Z_j. Let p_i, $i = 0, \ldots, M$ be some sampling points on a curve Z_k taken by a curvature adaptive sampling method [26]. Let $\angle(A, B)$ denote the angle between two vectors A and B. Shape characteristic is best evaluated by curvature. A discrete curvature value is defined at each vertex p_i by

$$\kappa(p_i) = \angle(p_{i+1} - p_i, p_i - p_{i-1}) / \|p_{i+1} - p_{i-1}\|. \quad (1)$$

We define *curvature histogram* of a curve for measuring shape characteristic using curvature values of some sampling points on a curve. Suppose the range of curvature values $[\kappa_{min}, \kappa_{max}]$ is uniformly divided into n pieces. A curvature histogram is a vector $H_k = \{h_0, h_1, \ldots, h_{n-1}\}$, $h_i = N_i/(M+1)$, where N_i is the number of sampling points in associating curvature slot (Fig. 5).

Table 1
Computational performance of our algorithm. The time numbers are measured in seconds.

Data set	Num. of points	Ske. seg. time	Points grouping time	Curve fitting time	Total time
15(a)	24,346	1.5	0.7	0.7	2.9
15(b)	19,447	1.7	0.6	0.7	3.0
15(c)	11,827	1.9	0.5	0.5	2.9
16(a)	10,637	1.5	0.6	0.5	2.6
16(b)	9436	1.3	0.5	0.4	2.2
16(c)	5066	0.9	0.3	0.3	1.5

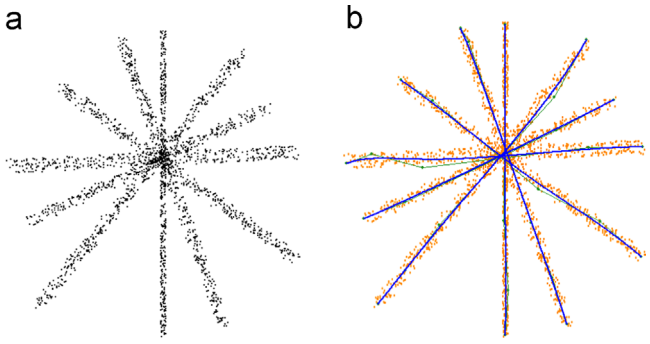

Fig. 8. An example containing large noise. (a) is the data points. (b) is the correct result using a proper size for deciding intersection region.

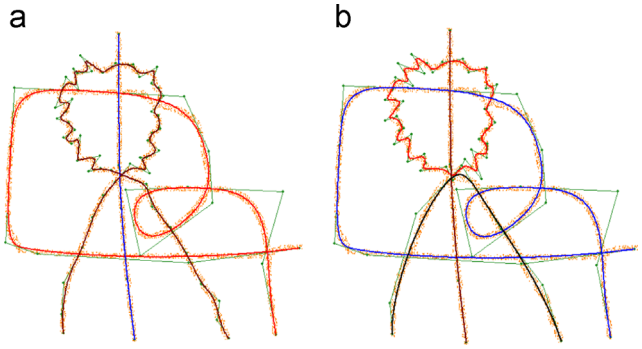

Fig. 9. This example demonstrates the affection of λ_1 and λ_2 in Eq. (2). (a) A result using $\lambda_1 = 10$, $\lambda_2 = 1.0$. (b) A result using $\lambda_1 = 0.1$, $\lambda_2 = 1.0$.

The connection edge of curve segments Z_k and Z_j is then assigned a cost value

$$\omega(h_c^{kj}) = \lambda_1 \alpha_{kj} + \lambda_2 \angle(H_k, H_j), \quad (2)$$

where α_{kj} is the turning angle of curve segments Z_k and Z_j (Fig. 4(b)); λ_1 and λ_2 are coefficients for balancing local turning and shape similarity of curve segments. The weight of a regular edge is defined by its average curvature

$$\omega\{h_r\} = \frac{\sum \kappa(p_i)}{M+1}.$$

3.3.2. Skeleton segmentation algorithm

We then extract paths in G for curve reconstruction. This can be done by finding paths in each subgraph g_k in G relating to an intersection region. g_k is obtained by extending from a connection subgraph to all incident edges until the extension distance is large enough or other connection edges are met. Refer to Fig. 6(a) for a subgraph of the example in Fig. 7. An algorithm (Algorithm 1) is applied to g_k to find the 'best' path in sequence, making use of the Dijkstra algorithm.

Algorithm 1. Polyline extraction.

Data: a graph g_k
Result: disjoint polylines
find all paths $\chi = \{l_k\}$ between each pair of end nodes in g_k using Dijkstra shortest path algorithm.
While g_k *is not empty* **do**
  If *there are more than* 2 *end nodes in* g_k **then**
  |find the path l_k which has minimal energy among χ;
  end
  else
  |return the final path.
  end
  store this path and remove edges and nodes of l_k from g_k;
  remove dangling edges from g_k and rename the remaining graph as g_k;
end

This procedure is illustrated in Fig. 6. In the second step, we remove all edges of the extracted path from g_k and the remaining subgraph is still a connected graph, unless the extracted path is the final path. Performing the algorithm for all intersections, we obtain a set of paths extracted from the whole set of data points. Using previously stored correspondence relationship of polygonal curves in K and paths in G, we obtain a set of polygonal curves L_i where each polygonal curve corresponds to a parametric curve to recover.

4. Point grouping and curve fitting

For curve fitting, we also need to segment the set of data points, in consistent with the skeleton segmentation result. This is achieved by associating each data point to the nearest skeleton curve. Ideally, this distance should be measured along the orthogonal direction to the skeleton curve. In practice, we perform dilation operations for all skeleton curves simultaneously. For each skeleton X_i, we apply the dilation algorithm to give an expanded skeleton image. The visited pixel q_i by skeleton i is marked with id i and the distance measurement d_i (the steps of dilation) to X_i is also stored in q_i. The dilation procedure stops when the expanding step is big enough to cover the width of point cloud. If a data point is visited by multiple skeletons, it is assigned to the skeleton with the minimal distance.

Once each pixel is correctly assigned to its nearest skeleton curve, it is simple to group the data points by checking associating covering pixels. Refer to Fig. 7(c) for the segmentation result of a skeleton, Fig. 7(d) for the segmentation result of data points. Note that a data point near intersection region may be assigned to multiple skeleton curves.

Once the cloud point has been classified into a number of groups, we fit each group of data points with a B-spline curve using the Squared Distance Minimization (SDM) method. The curvature based positions of nodes are good candidates for initial control points of B-spline curves. Refer to Fig. 7(e) for fitting B-spline curves obtained using SDM.

5. Experiments and discussion

5.1. Computational time

Our approach is implemented with C++ using Visual Studio 2012 platform. Our program runs on a desktop

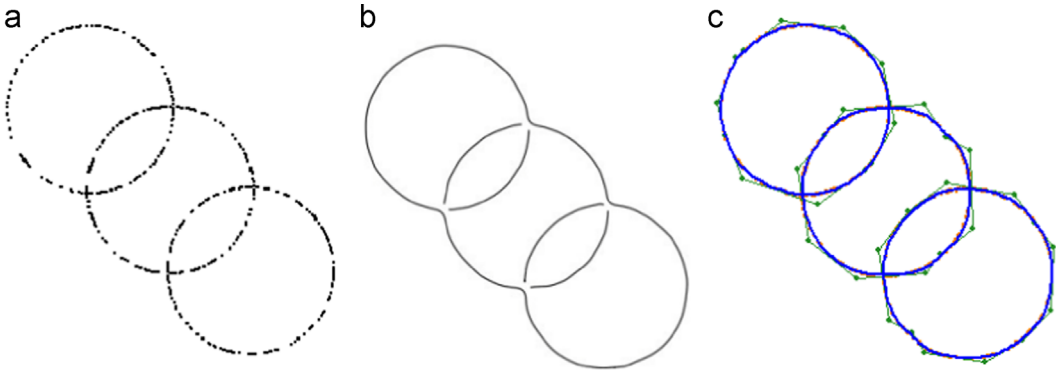

Fig. 10. A data set with crossing regions from [9]. (a) Data points. (b) Result from [9]. (c) Result of our method.

computer with 3.0 GHz CPU and 2 GB memory. The overall computing time for each example is about 1–2 s, which includes the computation time for triangulation, skeleton extraction, segmentation and curve fitting. The number of data points ranges from 5000 to 25,000. Refer to Table 1 for more details on computation time of some examples in this paper.

5.2. Examples and comparisons

Curve extraction problems appear in various application such as reverse engineering, image processing and pattern recognition, where a point cloud with complicated shape is processed to obtain multiple B-spline curves to represent the given point cloud.

To demonstrate the effectiveness of our method, we compare it with three existing methods. Examples in Figs. 10 and 11 are two data sets taken from [9]. Fig. 10 illustrates that the method in [9] gives incorrect grouping of point clouds. Fig. 11 shows one deficiency of the method in [9] that the reconstructed curve loses some end geometries (the red circle region). Fig. 12(a) shows that the method in [8] fails to recover an intersection region. Fig. 12(c) and (d) shows that the result of our method is more smooth and pleasing than the result of [8]. Fig. 13 gives an comparison of our method with the method in [5]. Notice that

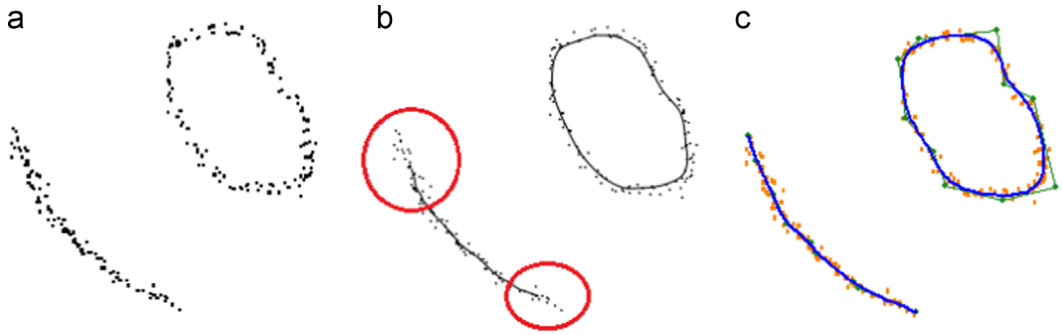

Fig. 11. An example from [9]. (a) Data points. (b) Result from [9]. Red circles indicate the regions where the reconstructed curve fails to cover. (c) is the result of our method. (For interpretation of the references to color in this figure caption, the reader is referred to the web version of this paper.)

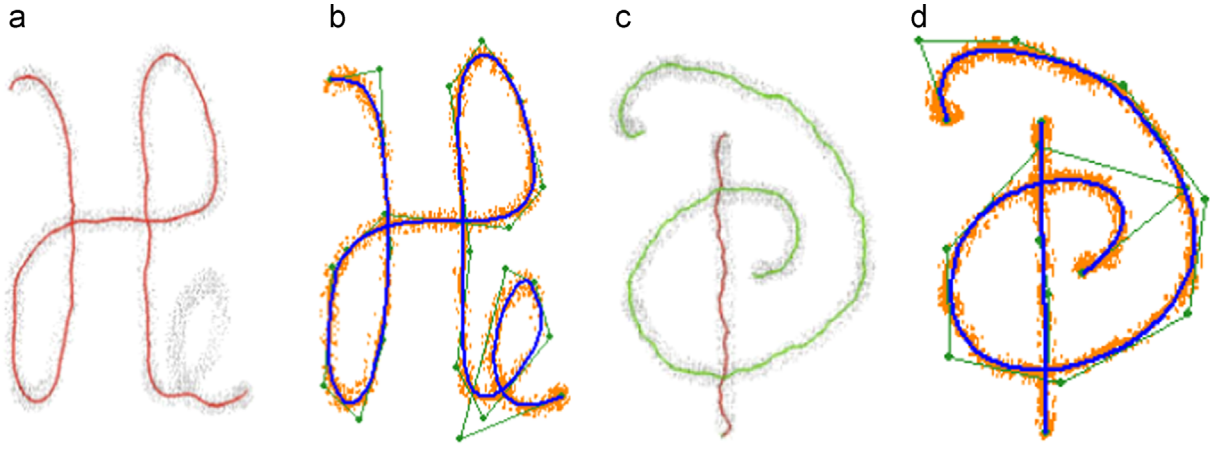

Fig. 12. An experiment compares our method with the method in [8]. (a) and (c) are results of [8]. (b) and (d) are results of our method.

Fig. 13. An example of cartoon image from [5]. (a) The original cartoon image. (b) Result from [5]. (c) Fitting B-spline curves obtained with our method.

the shapes in Fig. 13(b) are basically correct, while many details are lost. The result obtained by our method in Fig. 13(c) successfully preserves more details such as the ears.

Figs. 14–16 show more results of our method. The data points in Fig. 14 are obtained by intersecting 3D models with parallel planes. The fitting curves and reconstructed surfaces are shown. Fig. 15 gives some results of extracting B-spline curves from noisy point clouds with complex shapes. Our algorithm correctly recovers the topological structure of point clouds with B-spline curves. The results are reasonable segmentation of the data points and associating fitting B-spline curves. The reconstruction procedure is fully automatic.

A picture composed of simple line drawings is useful for imagination education for children. Our method can be applied to extract meaningful curve strokes from such images. Fig. 16 shows some segmentation results of simple line drawing pictures. Each character is segmented into a set of drawing lines rendered with different colors. The vectorization of images by fitting B-spline curves is also shown.

5.3. Discussion and limitations

In our algorithm, data points near intersection regions need to be removed. It is important to decide a proper size for intersection region. For most cases, the size of point cloud width gives satisfactory results, as demonstrated by the examples we have already shown. Our method for determining intersection regions works well for point cloud with large noise. Fig. 8 shows an experiment.

Eq. (2) is important for deciding curve segment combination. For all examples in Figs. 10–12 and 14–16, we set $\lambda_2 = 1$ and $\lambda_1 = 10$ and get pleasing results. However, in some cases, there are ambiguities in shapes and the reasonable solution is not unique. Refer to Fig. 9 for an example. In this case, we may adjust the values of λ_1 and λ_2 to give different results.

Our method depends on the skeleton of data points. If the skeleton results obtained using the image thinning algorithm is not correct for expressing the topology of data points, our curve extraction algorithm will consequently give wrong segmentation results and fitting curves. This happens when some curves are very close to each other as shown in Fig. 17.

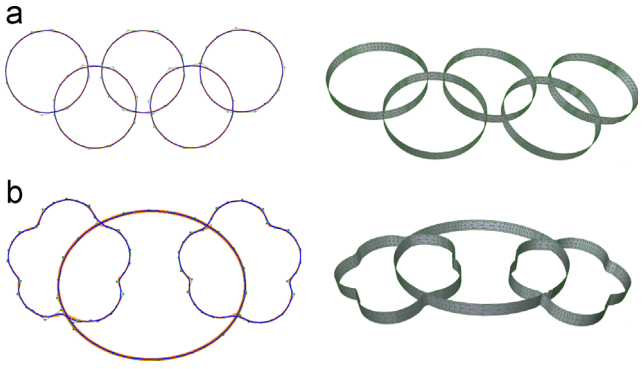

Fig. 14. Point clouds are generated by cutting 3D models with planar slices. The shown 3D surfaces are constructed using the segmentation result and fitting B-spline curves of our method.

6. Conclusion

We have described a novel method for reconstructing curves from a planar point cloud. We introduce a weighted graph representation of point cloud which is able to evaluate all segmentation results of data points by a shape evaluation functional. Curvature histogram is proposed for defining shape characteristic of a curve segment. Guided by this weighted graph and associated energy functional, we are able to divide the given data points into meaningful subgroups for curve fitting. Our algorithm has the virtue that it considers the shape goodness of all possible grouping results and is capable of finding curves having good shape if some assumptions on fitting curves are made. We demonstrated the performances of our proposed method by applying it to the problems of stroke recognition in simple line drawings and multiple curve reconstruction from planar point cloud with complicated shape.

For further work, we will try to apply our algorithm to some practical problems in image processing such as image segmentation and image vectorization. We are also interested in extracting strokes in hand-written characters and studying the semantic information of digital maps.

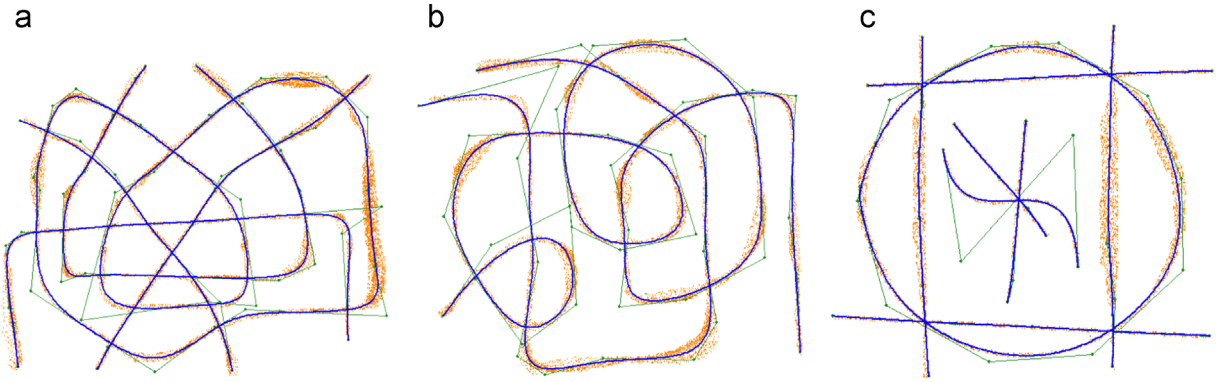

Fig. 15. Example of curve structure reconstruction from planar point clouds. Data points are rendered by yellow points. Fitting B-spline curves are rendered together with their control polygons. The example in (a) contains 24,346 data points. The example in (b) contains 19,447 data points. The example in (c) contains 11,827 data points. (For interpretation of the references to color in this figure caption, the reader is referred to the web version of this paper.)

Fig. 16. Segmentation and vectorization of several simple line drawing images. We use different color to render extracted line stokes. (a) Results of segmentation; (b) fitting B-spline curves as a vectorization of the image. The number of data points (pixels) are 10,637, 9436 and 5066 from left to right respectively. (For interpretation of the references to color in this figure caption, the reader is referred to the web version of this paper.)

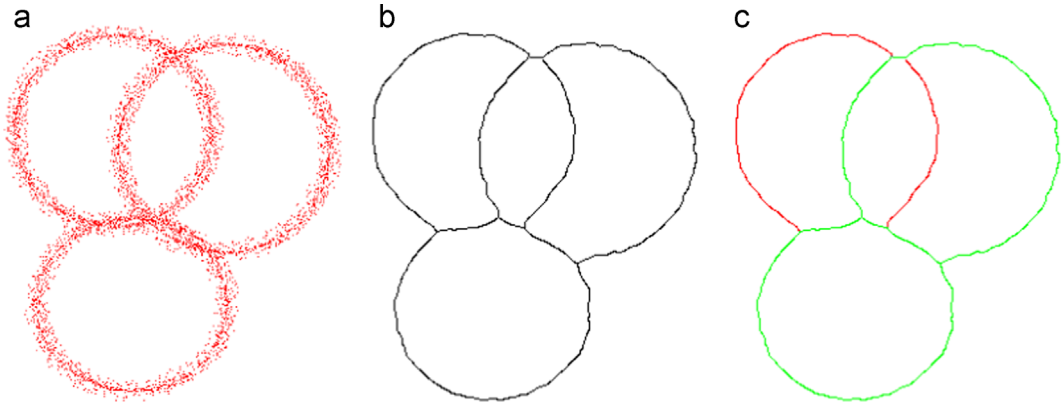

Fig. 17. An example of three crossing circles. (a) Data points. The skeleton result is shown in (b) which does not express three crossing circles. As a result, a wrong segmentation result is obtained (c).

Conflict of interest

The authors do not have any conflict of interest.

Acknowledgments

This work is supported by National Natural Science Foundation of China (Grant nos. 61202275, 61173086).

References

[1] de Goes F, Cohen-Steiner D, Alliez P, Desbrun M. An optimal transport approach to robust reconstruction and simplification of 2d shapes. *Computer Graphics Forum* 2011;**30**(5)1593–602.

[2] Dey TK, Wenger R. Reconstructing curves with sharp corners. *Computational Geometry* 2001;**19**(2–3)89–99.

[3] Amenta N, Bern M, Eppstein D. The crust and the β-skeleton: *combinatorial curve reconstruction*. *Graphical Models and Image Processing* 1998;**60**(2)125–35.

[4] Dey TK, Kumar P. A simple provable algorithm for curve reconstruction. In: SODA; 1999; p. 893–4.
[5] Wang J, Yu Z, Zhang W, Wei M, Tan C, Dai, N, et al. Robust reconstruction of 2d curves from scattered noisy point data. *Computer-Aided Design* 2014;**50**:27–40.
[6] Ruiz O, Vanegas C, Cadavid C. Ellipse-based principal component analysis for self-intersecting curve reconstruction from noisy point sets. *The Visual Computer* 2011;**27**(3)211–26.
[7] Zhu D, Bo P, Zhou Y, Zhang C, Wang K. Fitting multiple curves to point clouds with complicated topological structures. In: The 13th International Conference on Computer-Aided Design and Computer Graphics; 2013; p. 60–7.
[8] Einbeck J, Tutz G, Evers L. Local principal curves. *Statistics and Computing* 2005;**15**(4)301–13.
[9] Goshtasby AA. Grouping and parameterizing irregularly spaced points for curve fitting. *ACM Transactions on Graphics (TOG)* 2000;**19**(3) 185–203.
[10] Furferi R, Governi L, Palai M, Volpe Y. From unordered point cloud to weighted b-spline-a novel pca-based method. In: Proceedings of the 2011 American Conference on Applied Mathematics and The 5th WSEAS International Conference on Computer Engineering and Applications, World Scientific and Engineering Academy and Society (WSEAS), Stevens Point, Wisconsin, USA; 2011; p. 146–51.
[11] Yan H. Fuzzy curve-tracing algorithm. *IEEE Transactions on Cybernetics, Systems, Man, and Cybernetics, Part B: Cybernetics* 2001(5): 768–80.
[12] Zhao Y-d, Cao J-j, Su Z-x, Li Z-y. Efficient reconstruction of non-simple curves. *Journal of Zhejiang University—SCIENCE C* 2011;**12**(7) 523–32 http://dx.doi.org/10.1631/jzus.C1000308.
[13] Levin D. The approximation power of moving least-squares. *Mathematics of Computation of the American Mathematical Society* 1998;**67**(224) 1517–31.
[14] Lee I-K. Curve reconstruction from unorganized points. *Computer Aided Geometric Design* 2000;**17**(2)161–77.
[15] Pottmann H, Leopoldseder S, Hofer M. Approximation with active b-spline curves and surfaces. In: Proceedings of the 10th Pacific Conference on Computer Graphics and Applications. Washington, DC, USA; IEEE Computer Society; 2002; p. 8–18.
[16] Wang W, Pottmann H, Liu Y. Fitting b-spline curves to point clouds by curvature-based squared distance minimization. *ACM Transactions on Graphics (TOG)* 2006;**25**(2)214–38.
[17] Melhi M, Ipson SS, Booth W. A novel triangulation procedure for thinning hand-written text. *Pattern Recognition Letters* 2001;**22**(10) 1059–71.
[18] Song Y. Boundary fitting for 2d curve reconstruction. *The Visual Computer* 2010;**26**(3)187–204.
[19] Zhang T, Suen CY. A fast parallel algorithm for thinning digital patterns. *Communications of the ACM* 1984;**27**(3)236–9.
[20] Dey TK, Zhao W. Approximate medial axis as a voronoi subcomplex. *Computer-Aided Design* 2004;**36**(2)195–202.
[21] Zhu Y, Sun F, Choi Y-K, Jüttler B, Wang W. Spline approximation to medial axis. CoRR abs/1307.0118.
[22] Aichholzer O, Aigner W, Aurenhammer F, Hackl T, Jüttler B, Rabl M. Medial axis computation for planar free-form shapes. *Computer-Aided Design* 2009;**41**(5)339–49.
[23] Ramel J-Y, Vincent N, Emptoz H. A structural representation for understanding line-drawing images. *International Journal on Document Analysis and Recognition* 2000;**3**(2)58–66.
[24] Noris G, Hornung A, Sumner RW, Simmons M, Gross M. Topology-driven vectorization of clean line drawings. *ACM Transactions on Graphics (TOG)* 2013;**32**(1)4.
[25] Zou JJ, Yan H. Extracting strokes from static line images based on selective searching. *Pattern Recognition* 1999;**32**(6)935–46.
[26] Zitnick CL. Handwriting beautification using token means. *ACM Transactions on Graphics* 2013;**32**(4)53:1–8.