Journal of Universal Computer Science, vol. 26, no. 9 (2020), 1265-1280
submitted: 2/12/19, accepted: 4/8/20, appeared: 28/9/20

Reconstruction of Curve Networks from Unorganized Spatial Points

Shuangbu Wang, Yu Xia, Lihua You, Jianjun Zhang
(National Centre for Computer Animation, Bournemouth University, UK
swang1@bournemouth.ac.uk, yxia@bournemouth.ac.uk
lyou@bournemouth.ac.uk, jzhang@bournemouth.ac.uk)

Abstract: Curve network reconstruction from a set of unorganized points is an important problem in reverse engineering and computer graphics. In this paper, we propose an automatic method to extract curve segments and reconstruct curve networks from unorganized spatial points. Our proposed method divides reconstruction of curve networks into two steps: 1) detecting nodes of curve segments and 2) reconstructing curve segments. For detection of nodes of curve segments, we present a principal component analysis-based algorithm to obtain candidate nodes from unorganized spatial points and a Euclidean distance-based iterative algorithm to remove peripheral nodes and find the actual nodes. For reconstruction of curve segments, we propose an extraction algorithm to obtain the points on each of curve segments. We present quite a number of examples which use our proposed method to reconstruct curve networks from unorganized spatial points. The results demonstrate the effectiveness of our proposed method and its advantages of good automation and high reconstruction efficiency.

Key Words: curve reconstruction, curve network, node detection, curve extraction
Category: I.3.7,J.6

1 Introduction

Curve reconstruction, sometimes known as curve fitting, plays an important role in reverse engineering and computer graphics. According to whether reconstructed curves pass through unorganized points or not, curve reconstruction can be separated into two types: connecting nearby points to generate a curve [Figueiredo and Miranda Gomes 1994, Amenta et al. 1998, Dey et al. 2000, Parakkat and Muthuganapathy 2016, Ohrhallinger and Mudur 2013] and finding an approximate curve from point sets [Wang et al. 2006, Khanna and Rajpal 2015, Cheng et al. 2005]. According to whether unorganized points are two dimensional or three dimensional, curve reconstruction can be also divided into 2D curve reconstruction and 3D curve reconstruction.

Although there are significant advances in 2D curve reconstruction [Dey 2006], there is a tiny amount of research in 3D curve reconstruction. Since 2D curve reconstruction is an ill-posed problem, adding a new dimension by 3D curve reconstruction will aggravate the inherent ill-posedness. Due to this problem, most of the existing work on 3D curve reconstruction only rebuilds a simple 3D curve from point data [Lee 2000, Liu et al. 2006, Kumar et al. 2004]. In order to tackle this problem, we will propose a new method to automatically reconstruct a curve network consisting of multiple 3D curve segments from unorganized spatial points, as shown in Figure 1.

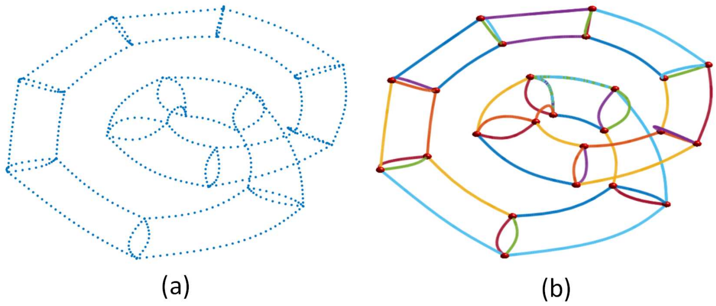

Figure 1: (a) The unorganized spatial point set of a knot model. (b) Reconstructed curve network using our method.

Curve network reconstruction is an essential step in curve network-based surfacing [Pan et al. 2015, Zou et al. 2015, Abbasinejad et al. 2011, Zhuang et al. 2013] from unorganized spatial points. It also can be used in constructing 3D models from laser range data and stereo measurements [Amenta et al. 1998]. Curve network-based surfacing, acting as an advanced modelling technique, is becoming increasingly popular [Zhuang et al. 2013]. With this method, a complicated geometric shape can be described by a complicated curve network which involves hundreds of 3D curve segments. Reconstructing such a complicated curve network from unorganized spatial points without connectivity information is very difficult. Although human may visually detect every curve segment when they look at the unorganized spatial points, the actual reconstruction process of these curve segments is very time-consuming and involves many tedious manual operations. The automatic method proposed in this paper addresses this issue. It significantly shortens the process and greatly raises the efficiency of curve network reconstruction.

Due to the following reasons, automatically reconstructing curve segments from unorganized spatial points is a challenging task. First, different segmentation criteria will lead to different segmentation results for a same set of unorganized spatial points. For example, the segmentation criterion based on sharp points and the one based on nodes, where one curve segment joins others, will lead to quite different quantities of curve segments. Second, it is difficult for a computer to judge whether a curve segment is good or bad. For example, Figure 2(a) shows a set of unorganized spatial points which define the eye of a dog's head in Figure 6. Although human can easily see that there are three curve segments between two end vertices, as shown in Figure 2(b), a computer may find different curve segments such as those shown in (c) and (d) which are two typical types of wrong curve segments.

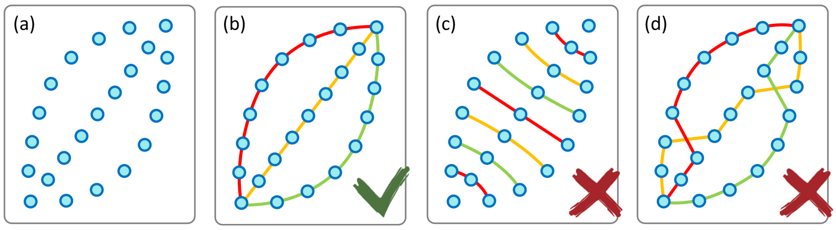

Figure 2: An example of unorganized spatial points (blue circles): (b), (c) and (d) shows three different types of curve segments obtained from the same 3D points in (a).

To solve these problems, we present a novel method to automatically obtain a correct curve network from unorganized spatial points. Our method is based on following two assumptions [Amenta et al. 1998]. (a) The input unorganized spatial point set is "clean", which means it have no self-intersections and short branches. (b) The unorganized spatial points are uniformly sampled point sets, and the sampling density is everywhere great enough to resolve the detail of the curve segments. Except for manually extracting curve segments from unorganized spatial points, we are unaware of any research activities on this topic. The contributions made in this paper are as follows.

- A principal component analysis-based algorithm to find candidate nodes from unorganized spatial points.
- A Euclidean distance-based iterative algorithm to remove peripheral nodes and find actual nodes from the candidate nodes.
- An inner point identification algorithm to extract all the inner points between two nodes of a curve segment.
- Integration of the above algorithms to develop the first method of automatically reconstructing curve network from unorganized spatial points.

The remaining parts of this paper are organized as follows. In Section 2, we briefly review related works on detection of nodes, curve reconstruction, and surfacing. Then, we give an overview of our proposed method and introduce the algorithms of automatically reconstructing curve network from unorganized spatial points in Section 3. After that, we test our method with various sets of unorganized spatial points in Section 4. Finally, we discuss and conclude our work in Section 5.

2 Related Works

Our proposed method is related to detection of nodes, curve reconstruction, and surfacing. In this section, we briefly review the most related work in these fields.

2.1 Detection of nodes

The problem of detecting the nodes among curves is an active research area, such as finding the nodes of Bézier curves [Manocha and Krishnan 1997], B-spline curves [Mørken et al. 2015] and NURBS curves [Rajab and Piegl 2014]. However, their work aims to detect nodes of planar and parametric curves. For unorganized points, [Wolin et al. 2008] proposed a simple algorithm for finding planar polyline corner. They resampled the points of the stroke and found the corners with the minimum straw distance. In particular, [Li et al. 1994] apply a distance accumulation method to detect corners on 3D space curves. Although their studied object is also spatial points, detecting curve corners is totally different from reconstructing curve segments from unorganized spatial points to be investigated in this paper. We have not found any publications which addressed this issue.

2.2 Curve reconstruction

A large body of research addresses curve reconstruction from unorganized planar points. To deal with noise-free point data, various methods have been developed by using Delaunay Triangulations such as Crust [Amenta et al. 1998] and NN Crust [Dey et al. 2000]. [Parakkat and Muthuganapathy 2016] presented a simple Delaunay triangulation based algorithm for non-parametric curve reconstruction from planar point set with different features like sharp corners, outliers and multiple objects. [Ohrhallinger and Mudur 2013] treated curve reconstruction as a minimization problem and developed an algorithm to reconstruct closed curves from a sparse point set and detect sharp corners.

Except for connecting nearby points to generate curves, fitting curves to noise point data is also an active area of research. [Wang et al. 2006] computed a planar B-spline curve to approximate an unorganized and noisy point cloud by using squared distance minimization. [Khanna and Rajpal 2015] developed an approach for curve reconstruction from a noisy point cloud based on fuzzy logic and ant colony optimization. [Cheng et al. 2005] proposed an algorithm to reconstruct polygonal closed curves from noisy samples drawn from a set of smooth closed curves.

Most of existing methods regard 3D curve reconstruction as an extended application of the algorithms of 2D curve reconstruction. [Lee 2000] investigated an algorithm to approximate a set of unorganized points with a simple curve by ordering the points with a regression line and a specified radius. The algorithm is also useful in reconstructing a 3D curve. [Kumar et al. 2004] presented an approach based on growing self-organizing maps for curve and surface reconstruction from an unorganized point cloud which can be extended to data points in 3D. [Liu et al. 2006] developed a least-squares projection algorithm which can thin a 3D point cloud and obtain a clean curve. All these research studies deal with a simple open 3D curve. The problem to be tackled in this paper is to reconstruct multiple 3D curve segments and obtain a curve network from unorganized spatial points. Existing research studies do not investigate this problem.

2.3 Surfacing

Surfacing is an important part of the design process. [Abbasinejad et al. 2011] developed a system to generate piecewise-smooth patches from unorganized 3D curves using a greedy algorithm.[Zhuang et al. 2013] proposed a novel routing system for finding cycles in a 3D curve network and creating surfaces. [Zou et al. 2015] investigated an algorithm to obtain a triangulation of multiple and non-planar 3D polygons. In order to make better surface shapes, [Pan et al. 2015] presented a surfacing method of curve networks by aligning the principal curvature directions with flow line. Since the aim of this paper is to reconstruct curve segments from unorganized spatial points, we only apply existing mature surfacing techniques to visualize and test our reconstructed curve network.

3 Algorithms

In this section, we first overview our proposed method. Then we investigate how to detect candidate nodes, determine the actual nodes, and use the actual nodes to reconstruct curve segments from unorganized spatial points.

3.1 Overview of proposed method

To avoid the poor results as shown in Figure 2(c), the correct end points of curve segments need to be detected at first. Generally, the two end points of a curve segment are two nodes in a curve network. Thus, we need to determine nodes from unorganized spatial points.

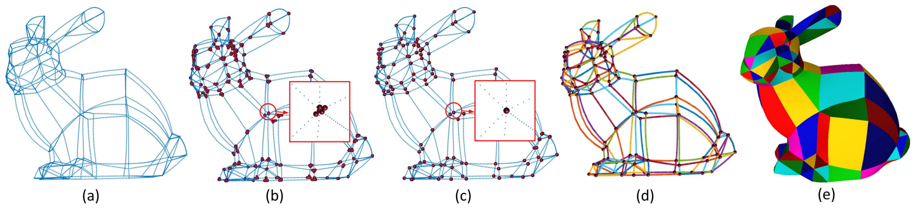

Figure 3: Pipeline of the proposed method. (a) An input set of unorganized spatial points (b) Detected and grouped candidate nodes shown in the square. (c) Obtained actual node in the square after iterative computations. (d) Reconstructed curve segments where different curve segments are shown in different colours. (e) The surface model after surfacing.

Our method includes three steps. The first step is detection of nodes which is elaborated in Subsection 3.2. In this step, we first propose a principal component analysis-based algorithm to find the candidate nodes and divide them into groups according to l^2 distance. Then we remove peripheral nodes in each group by iterative computations and finally obtain the actual nodes. The second step is curve reconstruction which is detailed in Subsection 3.3. In this step, we propose four inner point identification criteria: nearest point, point removal, minimum angle, and reverse search, and use them to extract all inner points and obtain a curve segment between two nearest nodes. The last step is surfacing. It is used to verify and visualize the result of the reconstructed curve segments. In this step, the curve network which has been reconstructed with our proposed method is input into a surfacing tool to produce a complete surface model.

The pipeline of our proposed method is illustrated in Figure 3. First, a set of unorganized spatial points of a bunny model shown in Figure 3(a) is input into our system. Then, the principal component analysis-based algorithm is used to detect candidate nodes from the unorganized spatial points which are grouped as depicted in Figure 3(b). Next, an iterative algorithm is proposed to remove the peripheral nodes and find the actual nodes indicated in Figure 3(c). After that, the inner point identification algorithm is introduced to find all the inner points and generate a network of curve segments presented in Figure 3(d). Finally, the network of curve segments is changed into a 3D surface model given in Figure 3(e) by surfacing for visualization and verification of the reconstruction result.

3.2 Node detection

A node in a curve network is a joining point of several curve segments. Unlike curve corners, nodes might not have sharp features which can be detected by using geometrical information. In order to identify these nodes, we introduce curvature variation based on the observation: the surrounding points of a node have a higher curvature variation than adjacent continuous points on a same curve because the surrounding points of the node, which lie on different connected curves, have different curvatures. Inspired by the work of [Pauly et al. 2003] where principal component analysis was used to estimate surface variation, in this paper, we apply principal component analysis to find the points with high curvature variation, and named this algorithm as a principal component analysis-based algorithm.

Given a set of unorganized spatial points P in R^3, we take an arbitrary point $P_i \in P$ and and let A be the set of its k neighbour points. The covariance matrix of A is defined as

$$Cov(A) = \frac{1}{k}([A] - P_i)^T([A] - P_i) \tag{1}$$

The point variation is given as

$$\delta(P_i) = \frac{\lambda_0}{\lambda_0 + \lambda_1 + \lambda_2} \tag{2}$$

where $\lambda_0 \leq \lambda_1 \leq \lambda_2$ are the eigenvalues of $Cov(A)$.

Since the eigenvector with λ_0 defines the least-squares plane through the set A of neighbour points, it approximates the normal of the surface which is formed by the set A. The point variation is to measure the deviation variation along the surface normal compared with the tangent plane at point P_i. Therefore, the point variation gives an estimation of curvature. We introduce a threshold ε for the point variation. If $\delta(P_i) > \varepsilon$, P_i is the point with high curvature variation and is selected as a candidate node. After calculating all input unorganized spatial points, we obtain a set of candidate nodes.

Since candidate nodes consist of actual intersections and their own surrounding points, they are presented in the form of different groups in the set of unorganized spatial points as shown in Figure 3(b). To obtain the actual node in each group, all groups of candidate nodes need to be detected at first. To this end, we introduce a user's specified distance τ_S. If the Euclidean distances between a random candidate nodes and the rest of candidate nodes are less than τ_S, these candidate nodes are divided into a same group and the nodes in the group is denoted as K_i.

After dividing the obtained candidate nodes into different groups, we use the following algorithm called the Euclidean distance-based iterative algorithm to remove peripheral nodes and find the actual nodes from the candidate nodes in each group. The algorithm is applicable to the nodes with different valences, such as valences of 3, 4, 5 and 6.

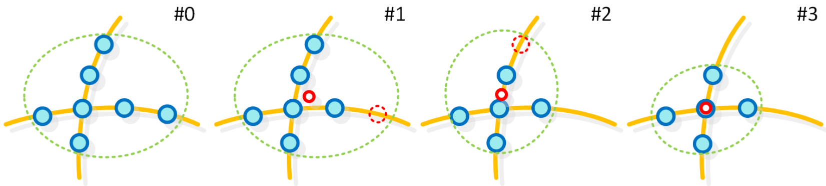

Figure 4: An example of the iterative process for finding an actual node: a group of candidate nodes (blue circles) are surrounded by green dashes. The peripheral node (red dash circle) from a centroid (red circle) is removed in the first iteration (#1) and second iteration (#2) and the actual node is obtained in the third iteration (#3).

Algorithm 1 Nodes Detection

Require: unorganized spatial points P
Ensure: nodes by matrix C

```
for each point P_i ∈ P do
    compute l^2 distance d(P_i, P_rest) and store k nearest points from P_i in set A_i
    calculate the λ_i of Cov(A_i)
    if δ(P_i) > ε then
        P_i is a candidate node and stored in set I
    end if
end for
while I ≠ φ do
    for a random candidate node I_i ∈ I, compute d(I_i, I_rest)
    if d(I_i, I_rest) ≤ τ_S then
        store these candidate nodes in group K and remove them from I
        compute the amount n of candidate nodes in K
        while n > 4 do
            Find the centroid of K and remove the peripheral candidate node from the centroid, and n = n − 1
        end while
        store the nearest candidate node from the centroid of K in matrix C as an optimal node, and empty K
    end if
end while
```

The algorithm consists of two steps. If one group has n nodes K_i $(i = 1, 2, ..., n)$, we first calculate the centroid E_n of the n nodes with $E_n = \frac{1}{n}\sum_{i=1}^{n} K_i$.

The second step is to calculate the l^2 distance between the centroid and each of the n nodes and remove a node according to $\arg\max_{i=1,2,...,n} \ d(E_n, K_i)$ where $d(\cdot)$ is the standard l^2 distance and K_i is the ith intersection in the group. The centroid of the remaining $n-1$ intersections is calculated again in the first step where n is replaced by $n-1$, and a new node according to $\arg\max_{i=1,2,...,n-1} \ d(E_{n-1}, K_i)$ is found in second step by changing n into $n-1$ and removed.

The above process is repeated until four nodes are left and the nearest node from the last centroid is an actual point (see Figure 4). The pseudo-code for the above process of detecting candidate nodes and iteratively removing the peripheral nodes to find the actual nodes is given in Algorithm 1.

3.3 Curve reconstruction

We reconstruct a smooth curve segments by connecting neighbour points from a node to another node through the following inner point identification algorithm.

This algorithm consists of four inner point identification criteria. They work together to find the right points on a same curve segment. The first criterion is the nearest point. It requires the point P_i to be identified as the nearest neighbour point to point P_{i-1} which has already been identified. This criterion ensures that every point will not be missed and the reconstructed curve segment is an original and real curve segment of the set of unorganized spatial points. The second criterion is the point removal. It means that once an inner point P_i is used to produce a curve segment, the point is removed immediately and will not be used in the following inner point identification processing. This criterion is useful because an inner point should not be identified more than once to produce a wrong curve segment, and this criterion can effectively prevent the mistakes occurring during the process of identifying inner points of a curve segment. The third criterion is the minimum angle. With this criterion, the vector angle θ_{P_0} determined by three continuous points excluding the node must be greater than an obtuse angle θ. This criterion can prevent us finding a wrong point from other curve segments (e.g. Figure 2(d)). The three criteria can be mathematically represented as

$$\begin{aligned} &\arg\min_{P_i \in P} \ d(P_{i-1}, P_i) \\ &s.t. \ \ \theta_{P_{i-1}} \geq \theta, P_i \notin P_{used} \end{aligned} \tag{3}$$

where P_i is an arbitrary inner point and P_{used} is a group of already identified points. $\theta_{P_{i-1}} = \angle(\overrightarrow{P_{i-2}P_{i-1}}, \overrightarrow{P_{i-1}P_i})$ where P_{i-2} and P_{i-1} are identified points, and θ is the users specified obtuse angle which can be set between 100° and 150°.

Algorithm 2 Curve Reconstruction

Require: : nodes C and unorganized spatial points P
Ensure: curve segments by matrix F

for each node $C_i \in C$ **do**
  remove all candidate nodes from P, and rank P in ascending sort order according to $d(C_i, P)$
  Initialize $j = 1$ and $F_1^{i,j} = C_i$ $\{F_k^{i,j}$ is kth point of jth curve staring from $C_i\}$
  while $d(P_j, C_i) < \tau_S$ **do**
    initialize $k = 2$
    if $P_j \notin F^i$ **then**
      $F_k^{i,j} = P_j$ and $k = k + 1$
    else
      $j = j + 1$, continue
    end if
    while $F_k^{i,j} = P_j \notin C$ **do**
      find the nearest point P_n in P from $F_{k-1}^{i,j}$
      if $P_n \notin C$ and $\theta_{F_{k-1}^{i,j}} \geq \theta$ **then**
        $F_k^{i,j} = P_n$,$k = k + 1$ and remove P_n from P
      else if $P_n \notin C$ and $\theta_{F_{k-1}^{i,j}} \leq \theta$ **then**
        then find next nearest point P_n
      else
        $F_k^{i,j} = P_n \in C$ and $k = k + 1$
      end if
    end while
    $j = j + 1$
  end while
end for
for each curve $F^{i,j} \in F$ **do**
  if $F^{i,j} \neq \phi$ **then**
    find the same curve F^{i_s,j_s} and replace the points next to the ending node of $F^{i,j}$ with the points next to the starting node of F^{i_s,j_s}, and remove F^{i_s,j_s} from F
  end if
end for

If two curve segments meet at a same node and are close together as shown in Figure 5(a), the second criterion may not work properly, i. e., the point on another curve may be identified and selected when the identification process of a curve segment is near finish as shown in Figure 5(b). In order to avoid this problem, we introduce the fourth criterion, i. e., the reverse search. It requires a same curve segment to be searched from two opposite directions, and the point next to the ending node identified during the search from the right to the left shown by the bottom one in Figure 5(c) should be replaced by the point next to the starting node identified during the search from the reverse direction, i. e. from the left to the right shown by the top one in Figure 5(c), to obtain the correct curve segment shown in Figure 5(d).

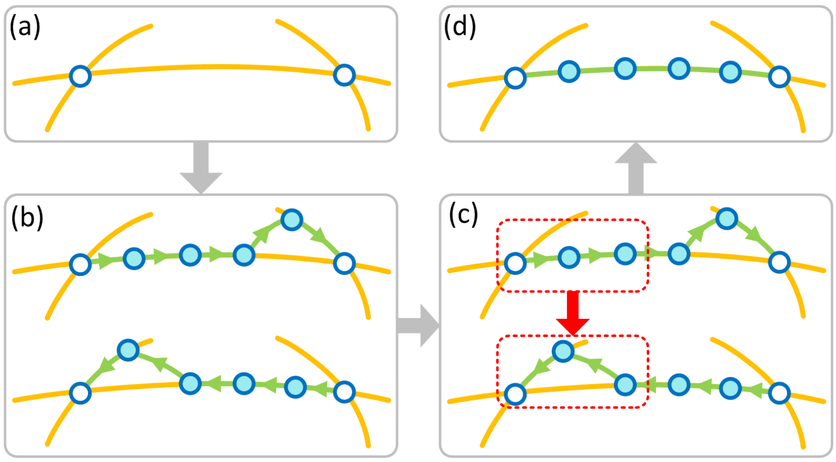

Figure 5: An example of the reverse search process. (a) Two nodes; (b) Two curve segments between two same nodes (the green line means the growing direction of the curve, and the top curve is from left to right and the bottom curve is from right to left); (c) The point next to the ending node of the bottom curve is replaced by the point next to the starting node of the top curve; (d) Final curve segment.

The pseudo-code for the inner point identification algorithm is given in Algorithm 2. It is combined with Algorithm 1 to automatically reconstruct curve network from the unorganized spatial points.

4 Results

We implemented our algorithms in MATLAB. In order to help visualize the results of curve reconstruction, we use the surfacing tool of [Zhuang et al. 2013] to create surface models from the reconstructed curve networks of unorganized spatial points with our proposed curve reconstruction method. We demonstrate our method on a large number of unorganized spatial point sets sampled from curve networks provided by surfacing work [Pan et al. 2015, Zhuang et al. 2013]. All the experiments were carried out on a 3.5 GHz PC with 32 GB memory.

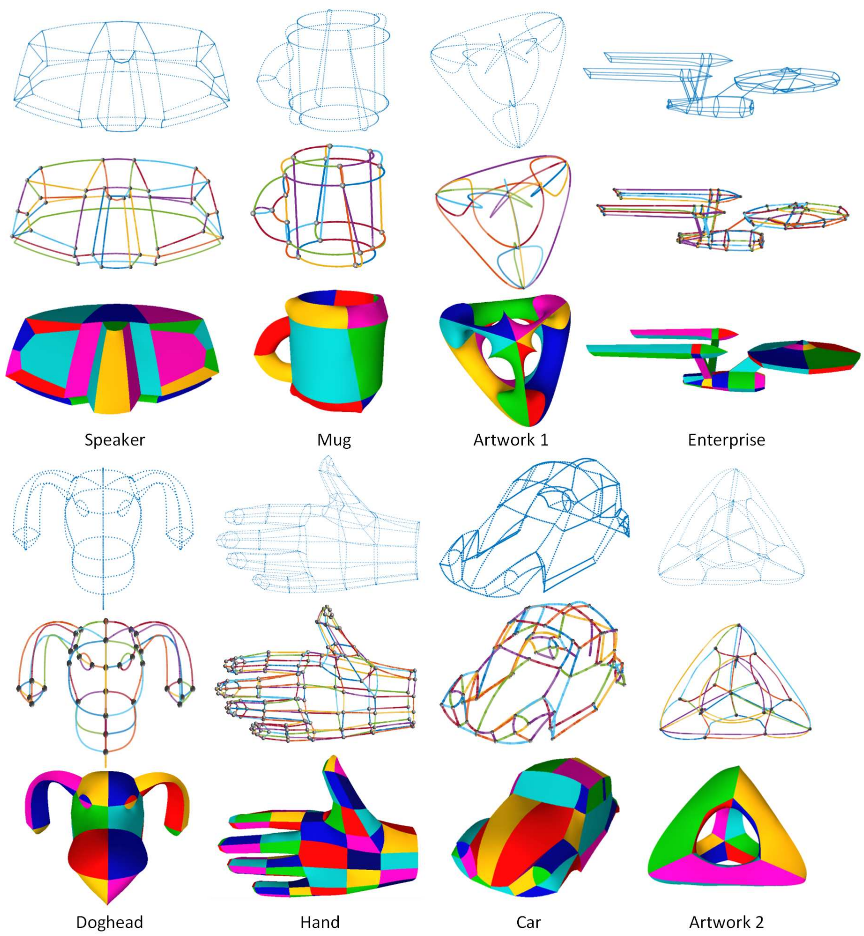

Figure 6: Test results of examples. Each example includes unorganized spatial points (top), curve network (middle) and surface model (bottom).

Our method successfully reconstructed curve networks from different unorganized spatial point sets. Figure 6 shows 8 models among the original test samples. Table 1 shows the running time, the numbers of nodes (N_n), curve segments (N_c) and data points (N_d) ,and the applicable values of ε, τ_S and θ for the various spatial point sets used in Figure 1, Figure 2 and Figure 6. As can be derived from the reconstruction algorithm above, the increase in runtime is related to the model complexity, sampled interspace and the number of curve segments. The results indicate that our proposed automatic reconstruction method is highly efficient.

Table 1: Statistical data of test examples.

Model	N_n	N_c	N_d	ε	τ_S	$\theta(°)$	Time(s)
Speaker	38	64	6884	2e-7	0.02	120	14
Doghead	33	67	14864	1e-4	0.01	120	55
Hand	150	281	25947	2e-7	0.01	120	111
Bunny	123	270	17638	1e-6	0.02	150	49
Enterprise	110	225	33721	1e-5	0.005	120	180
Mug	20	42	16777	1e-4	0.01	120	40
Car	75	132	15021	1e-7	0.01	150	82
Artwork 1	8	24	13668	1e-5	0.01	150	60
Artwork 2	20	48	18530	1e-5	0.01	150	44
Knot	24	48	2918	1e-4	0.1	120	3

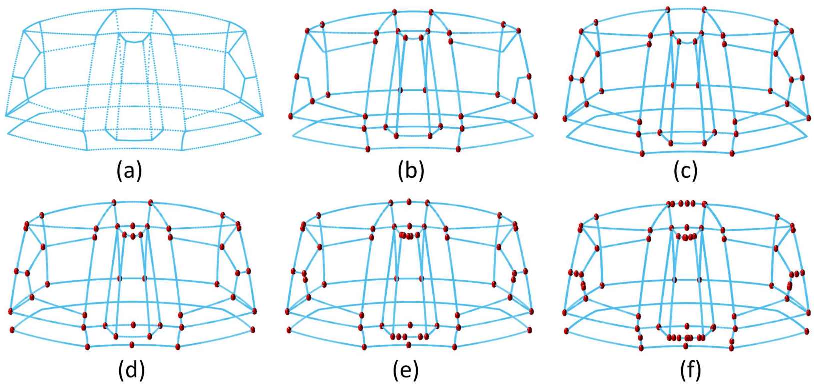

Figure 7: The reconstructed curve networks with different threshold ε (the red points represents nodes): (a) unorganized spatial point set. (b) $\varepsilon = 2e-6$. (c) $\varepsilon = 2e-7$. (d) $\varepsilon = 1e-8$. (e) $\varepsilon = 5e-9$. (f) $\varepsilon = 1e-9$.

The threshold ε in Table 1 ensures that the reconstructed curve network exactly satisfies the actual structure of the original unorganized spatial point set. Figure 7 shows the reconstructed curve networks of the speaker model with different threshold ε. When ε has a large value, the reconstruction is unsuccessful and the curve network misses several curve segments (Figure 7(b)). When the value of ε is reduced, the number of nodes will increase which leads to an increase in curve segments (Figure 7(d-f)). Although the quantity of curve segments is different from the actual curve network which is shown in Figure 7(c), these increased curve segments have no influence on the actual structure, and the reconstructed results (Figure 7(d-f)) are acceptable and can be used to generate the surface model.

Algorithm 2 of our proposed method is applicable to reconstruction of 2D curves. In order to demonstrate the applicability of Algorithm 2 in 2D curve reconstruction, here we compare it with the method of Euclidean minimal spanning trees (EMST) which can correctly reconstruct differentiable arcs from sufficiently dense samples as proved in [Figueiredo and Miranda Gomes 1994]. Figure 8 shows the comparison of curve reconstructions from 2D points between Algorithm 2 of our proposed method and the EMST method. As shown in the figure, Algorithm 2 of our proposed method reconstructs the two curves correctly. In contrast, the EMST method gives wrong curve reconstruction results.

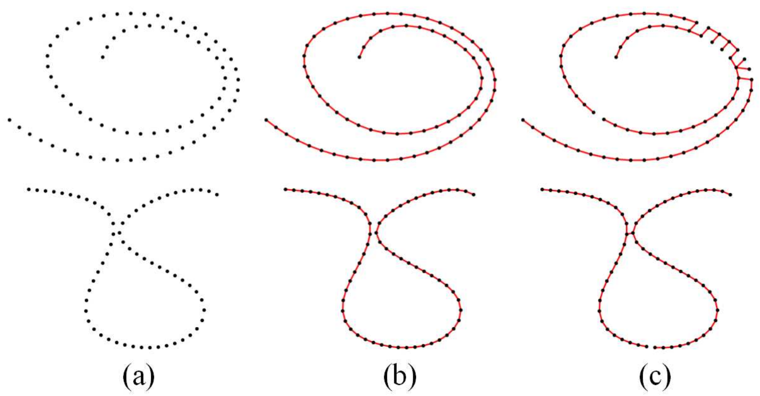

Figure 8: Comparison of curve reconstructions from 2D points (a) between our method (b) and EMST method (c).

5 Conclusions

To the best of our knowledge, we have proposed the first automatic method for reconstructing curve networks from unorganized spatial points. Our key insight is to divide the reconstruction process into two parts: node detection and inner point identification. We first limit the search scope of curve segments by using the principal component analysis-based algorithm to detect all candidate nodes. Then we employ the Euclidean distance-based iterative algorithm to remove peripheral nodes and obtain the actual nodes. Finally we find all inner points between two nodes and extract the curve segments based on our proposed inner point identification algorithm. The test results show that our method can automatically and effectively process various unorganized spatial point sets and obtain the networks of curve segments with good automation and high extraction efficiency.

Despite its advantages, there are a several limitations in our method. First, determination of the parameters ε, τ_S and θ is based on trial-and-error, and could be obtained through optimization analysis. Second, the unorganized spatial points are uniformly sampled which partly limits further applications of our method. We have found that applying Delaunay triangulations can solve the problem which will be investigated in our future work.

Acknowledgements

This research is supported by the PDE-GIR project which has received funding from the European Union's Horizon 2020 research and innovation programme under the Marie Skłodowska-Curie grant agreement No 778035.

References

[Abbasinejad et al. 2011] Abbasinejad, F., Joshi, P., Amenta, N.: Surface patches from unorganized space curves. In Computer Graphics Forum, 30(5), (2011), 1379-1387.

[Amenta et al. 1998] Amenta, N., Bern, M., Eppstein, D. : The crust and the β-skeleton: Combinatorial curve reconstruction. Graphical models and image processing, 60(2),(1998), 125-135.

[Cheng et al. 2005] Cheng, S. W., Funke, S., Golin, M., Kumar, P., Poon, S. H., Ramos, E.: Curve reconstruction from noisy samples. Computational Geometry, 31(1-2), (2005), 63-100.

[Dey 2006] Dey, T.K.: Curve and surface reconstruction: algorithms with mathematical analysis (Vol. 23). Cambridge University Press, (2006).

[Dey et al. 2000] Dey, T. K., Mehlhorn, K., Ramos, E. A.: Curve reconstruction: Connecting dots with good reason. Computational Geometry, 15(4), (2000), 229-244.

[Figueiredo and Miranda Gomes 1994] De Figueiredo, L.H. and de Miranda Gomes, J.: Computational morphology of curves. The Visual Computer, 11(2), (1994), 105-112.

[Khanna and Rajpal 2015] Khanna, K., Rajpal, N.: Reconstruction of curves from point clouds using fuzzy logic and ant colony optimization. Neurocomputing, 161, (2015), 72-80.

[Kumar et al. 2004] Kumar, G. S., Kalra, P. K., Dhande, S. G.: Curve and surface reconstruction from points: an approach based on self-organizing maps. Applied Soft Computing, 5(1), (2004), 55-66.

[Lee 2000] Lee, I. K.: Curve reconstruction from unorganized points. Computer Aided Geometric Design, 17(2), (2000), 161-177.

[Li et al. 1994] Li, S. Z., Wang, H., Ang, T. H., Bey, K. M.: Detection of corners on 3D space curves. In Intelligent Robots and Computer Vision XIII: 3D Vision, Product Inspection, and Active Vision, 2354, (1994), 339-346.

[Liu et al. 2006] Liu, Y. S., Paul, J. C., Yong, J. H., Yu, P. Q., Zhang, H., Sun, J. G., Ramani, K.: Automatic least-squares projection of points onto point clouds with applications in reverse engineering. Computer-Aided Design, 38(12), (2006), 1251-1263.

[Manocha and Krishnan 1997] Manocha, D., Krishnan, S.: Algebraic pruning: a fast technique for curve and surface intersection. Computer Aided Geometric Design, 14(9), (1997), 823-845.

[Mørken et al. 2015] Mørken, K., Reimers, M., Schulz, C.: Computing intersections of planar spline curves using knot insertion. Computer Aided Geometric Design, 26(3), (2009), 351-366.

[Ohrhallinger and Mudur 2013] Ohrhallinger, S., Mudur, S.: An efficient algorithm for determining an aesthetic shape connecting unorganized 2d points. In Computer Graphics Forum 32(8), (2013), 72-88.

[Pan et al. 2015] Pan, H., Liu, Y., Sheffer, A., Vining, N., Li, C. J., Wang, W. Flow aligned surfacing of curve networks. ACM Transactions on Graphics (TOG), 34(4), (2015), 127.

[Parakkat and Muthuganapathy 2016] Parakkat, A. D., Muthuganapathy, R.: Crawl through neighbors: A simple curve reconstruction algorithm. In Computer Graphics Forum 35(5), (2016), 177-186.

[Pauly et al. 2003] Pauly, M., Keiser, R., Gross, M.: Multi-scale feature extraction on point-sampled surfaces. In Computer graphics forum, 22(3), 2003(September), 281-289.

[Rajab and Piegl 2014] Rajab, K., Piegl, L. A.: A Knowledge-Guided Approach to Line NURBS Curve Intersection. Computer-Aided Design and Applications, 11(1), (2014), 1-9.

[Wang et al. 2006] Wang, W., Pottmann, H., Liu, Y.: Fitting B-spline curves to point clouds by curvature-based squared distance minimization. ACM Transactions on Graphics (ToG), 25(2), (2006), 214-238.

[Wolin et al. 2008] Wolin, A., Eoff, B., Hammond, T.: ShortStraw: A Simple and Effective Corner Finder for Polylines. In SBM, (2008), 33-40.

[Zhuang et al. 2013] Zhuang, Y., Zou, M., Carr, N., Ju, T.: A general and efficient method for finding cycles in 3D curve networks. ACM Transactions on Graphics (TOG), 32(6), (2013), 180.

[Zou et al. 2015] Zou, M., Ju, T., Carr, N.: An algorithm for triangulating multiple 3D polygons. In Proceedings of the Eleventh Eurographics/ACMSIGGRAPH Symposium on Geometry Processing, (2013), 157-166.