Curve Networks for Surface Reconstruction

Yuanhao Cao, Liangliang Nan*, Peter Wonka

Abstract—Man-made objects usually exhibit descriptive curved features (i.e., curve networks). The curve network of an object conveys its high-level geometric and topological structure. We present a framework for extracting feature curve networks from unstructured point cloud data. Our framework first generates a set of initial curved segments fitting highly curved regions. We then optimize these curved segments to respect both data fitting and structural regularities. Finally, the optimized curved segments are extended and connected into curve networks using a clustering method. To facilitate effectiveness in case of severe missing data and to resolve ambiguities, we develop a user interface for completing the curve networks. Experiments on various imperfect point cloud data validate the effectiveness of our curve network extraction framework. We demonstrate the usefulness of the extracted curve networks for surface reconstruction from incomplete point clouds.

Index Terms—Curve Network, Surface Reconstruction, Feature Curve, Point Cloud, Regularity

1 Introduction

Man-made objects usually exhibit descriptive curved features. These curved features, we call *curve networks*, convey the structural characteristics of the objects. From the visual perception perspective, they serve as high-level representation of the objects. Various applications, such as non-photorealistic rendering [2], product design [3], abstraction [4], segmentation [5], reconstruction [6], and shape editing [7], have exploited and benefited from the extracted curve networks. In this work, we are interested in the problem of extracting curve networks from noisy and incomplete point clouds and using these extracted networks to guide surface reconstruction from incomplete point clouds.

In the last decades, the prevalence of various laser scanners and depth cameras (e.g., Kinect) enabled non-professional users to obtain a 3D sampling of an object in a matter of seconds. However, due to occlusions and specific material properties (e.g., transparent, reflective), obtaining a point cloud with reasonably coverage of an object remains a challenge. In practice, it is quite common that significant portions of the object are either under-sampled or completely missing in a 3D point cloud. This limits the applicability of the widely available acquisition devices and hinders the application of the large amount of existing point cloud data.

The main problem for reconstruction from partial point cloud data comes from the lack of constraints in the missing regions. Thus, the reconstruction is ill-posed as an infinite number of valid surfaces may pass these regions. To compensate for the lack of constraints, smoothness is usually exploited to fill the holes in the reconstructed surface models. However, the smoothness constraints are too local to fill holes occurring near sharp features. High-level constraints, such as symmetry, may provide an efficient completion tool [8]. However, when symmetry is not applicable, or the data is highly incomplete, it is impossible to infer a faithful 3D completion and reconstruction. Given the descriptive characteristics of curve networks, the motivation of this work is to extract such curve networks from partial point cloud data and utilize them as geometric and topological constraints to regularize the ill-posed surface reconstruction problem. To this end, we present a hybrid framework for extracting high quality curve networks from unstructured point cloud data that may have severe missing regions.

Our overall contributions are as follows:

- a novel framework that can effectively extract curve networks form partial point clouds.
- an optimization algorithm exploiting structural regularities to enhance the extracted curve networks to be regular and meanwhile respect the input point clouds.
- we demonstrate that the extracted curve networks can significantly regularize surface reconstruction from incomplete point clouds.

2 Related Work

There exists a large volume of work related to surface reconstruction in literature. In this section, we mainly review the work that are closely related to feature detection, curve based modeling, surface reconstruction from curves, and curve network extraction.

Feature detection. Quite a few techniques have been proposed for detecting curved features on polygonal models and point clouds. Lee et al. propose geometric snake [9], an interactive tool for detecting curved features from triangular meshes by extending the 2D active contour model (snakes) to 3D surfaces. The user sketches initial feature curves on the input surface, and the 3D snake iteratively snaps them to the curved features in the surface. Ohtake et al. [10] exploit implicit surface fitting to calculate extremal coefficients for extracting ridges and valleys from mesh surfaces. Kim et al. [11] utilize a variant of Moving-Least-Squares method to fit local surface patches in the neighborhood of each vertex, and then compute local curvatures based on the fitted local surface

* *Corresponding author.*

- *The authors are with the Visual Computing Center, King Abdullah University of Science and Technology, Thuwal 23955-6900, Saudi Arabia. E-mail: ppxhappy@126.com, {liangliang.nan, pwonka}@gmail.com.*

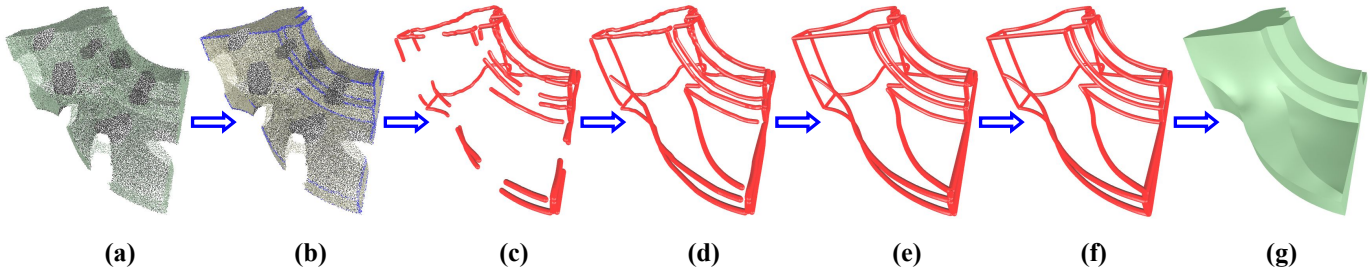

Fig. 1. Algorithm overview. (a) Input point cloud; (b) detected feature points; (c) initial curve segments; (d) improved feature curves with user guidance; (e) optimized feature curves; (f) final curve networks after completion; (g) the reconstructed surface model from the curve networks using the techniques of [1].

patches. Similarly, Yoshizawa et al. [12] extract crest lines by estimating the curvature tensor and curvature derivatives based on local polynomial fitting. In the work of Nomura and Hamada [13], the authors detect feature curves by calculating the skeleton of the feature region defined by the concavity and convexity.

To handle random noise, outliers, and artifacts, Min et al. [14] present a method based on the tensor voting to extract sharp features from unstructured point clouds. Mark et al. [15] extract feature lines from point-sampled geometry by computing a minimum spanning graph of feature nodes that have high probability belonging to a feature. By fitting spline curves, Joel et al. [16] identify sharp features in a point-based model and align the spline curves with the sharp edges of the model.

Curve based modeling. There are also techniques and systems that are able to transfer 2D sketches into 3D representations. For example, ILoveSketch [17], a 3D curve sketching system that captures some of the affordances of pen and paper for professional designers, allows a designer to iterate directly on conceptual 3D models. Other systems, such as Teddy [18] and Fibermesh [19], provide simple user interfaces for designing freeform surfaces from a collection of 2D sketches. The user first creates a rough 3D model by drawing 2D strokes. Then the 3D surface models can be further edited by sketching directly on the models, where the 3D curves serve as handles for controlling the geometry. Recently, Baoxuan et al. [3] design a sketch-based modeling system (i.e., True2Form) that reconstructs 3D curve networks from typical 2D design sketches. Their strategy relies on prior knowledge to enforce structural regularities of an object.

Surfaces reconstruction from curves. The motivation of this task is to recover full geometry from a set of curves spanning in the surfaces. Orbay and Kara [20] propose a sketch-based modeling interface for creating smooth surfaces from curve networks. Based on a linear algebra representation of suface patches, Abbasinejad et al. [6] introduce a system that supports automatic generation of piecewise smooth surfaces from curve networks. With similar motivation, Bessmeltsev et al. [21] present a design-driven approach for quadrangulating closed 3D curve networks. Zou et al. [22] present an algorithm for triangulating 3D spatial polygons. To fill holes, an N-sided hole filling technique proposed by Tamás et al. [23] can interpolate the boundary curve of each hole. For partial scans with large missing parts, Nan et al. [24] propose to lift 2D image boundaries into 3D space to constrain the ill-posed surface reconstruction problem.

Curve network extraction. To extract closed curve networks, Demarsin et al. [25] propose an algorithm for extracting closed sharp feature lines from point clouds. Based on first order segmentation, they first extract candidate feature points and then represent them as a graph to recover the sharp feature lines. Then a minimum spanning tree is constructed to enclose these curved lines. In the work of Cao et al. [26], the authors extract curve networks by first detecting curved segments and then extending them to closed curve loops on surfaces.

Following the work of [26] and [3], we extract curve networks from partial and noisy point clouds through optimization. Our formulation enforces the detected curve networks to respect both data fitting and structural regularities of the objects.

3 Overview

Given a noisy and incomplete point cloud as input, our goal is to extract complete curve networks from such an imperfect input point cloud. Our framework consists of the following three key steps (an overview of our approach is shown in Fig. 1):

Curved segment generation. We first compute the surface variation at each point and extract regions of high-level variation using simple thresholding. Then, initial curved segments are generated by fitting curves to those feature points in the point cloud. For partial point clouds, we develop a simple user interface allowing a user to guide the curve fitting through loosely sketching strokes on the 3D point clouds (Sec. 4.1).

Curved segment optimization. We introduce an energy minimization formulation to optimize the feature curves. Our objective function is designed to enforce data fitting and the smoothness of the curved feature, and meanwhile to respect the structural regularities of the point cloud (Sec. 4.2).

Curve network completion. After the curved segments being optimized, we extend and connect these individual feature curves to generate complete curve networks. We use the algorithm proposed by Zhuang et al. [1] to detect cycles of surface patches from the curve networks (Sec. 4.3).

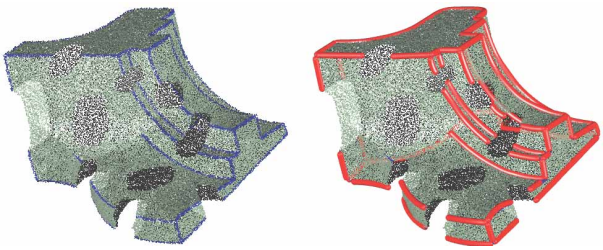

Fig. 2. Extracted feature points (left, in blue) and initial curved segments (right, in red) extracted from the point cloud.

4 METHOD

Given a noisy and incomplete point cloud as input, we first identify feature points in the point cloud. Then, we extract curved segments from these feature points via curve fitting. We also allow users to guide the curve fitting through a simple user interface for large missing regions in the point cloud.

4.1 Curved segment generation

We first detect feature points as those having high surface variations in the point cloud. Specifically, we use the method described in [15] to define the variation $\sigma(\mathbf{p})$ at a point $\mathbf{p}$ as

$$\sigma(\mathbf{p}) = \frac{\lambda_1}{\lambda_1+\lambda_2+\lambda_3}, \tag{1}$$

where $\lambda_i (1 \leq i \leq 3)$ denote the three eigenvalues (in an ascending order) of the covariance matrix defined on the neighborhood of $\mathbf{p}$. For more details on the covariance matrix setup, please refer to [15]. Since $\sigma(\mathbf{p})$ is invariant under scales at each point, so feature points can be identified by simply thresholding surface variations at each point in the input point cloud. Specifically, a point $\mathbf{p}$ is considered as a feature point if $\sigma(\mathbf{p}) > \sigma_t$, where σ_t is the threshold that is set to 0.04 in our experiment.

Curve segment extraction. From the identified feature points, we generate polylines using a modified version of [27] to fit the points with high variations.

To handle noisy point clouds, we propose an additional termination condition to prevent the polyline growing into multiple feature regions at sharp corners. Given an endpoint $\mathbf{p}_k$ and it direct neighbor $\mathbf{p}_{k-1}$ in the polyline, we extend $\mathbf{p}_k$ to $\mathbf{p}_{k+1}$ if the angle between $\overline{\mathbf{p}_{k-1}\mathbf{p}_k}$ and $\overline{\mathbf{p}_k\mathbf{p}_{k+1}}$ is smaller than 30°. In our implementation, we sort the feature points according to their variations and choose the point with highest variation as the seed point for propagating a polyline. After one polyline propagation is terminated, all the feature points within the neighborhoods $\mathbf{N}_{s_{max}}$ to the polyline are removed. We then choose a new seed point from the remaining feature points and propagate another polyline. We repeat this process until no feature points are left. Fig. 2 illustrate the feature points (left) and the extracted polylines (right) in the point cloud.

User guidance and symmetry. For point clouds with large missing regions, our automatic curve segment generation may fail to extract good polylines. So we develop a user interface that allows users to guide the feature curve extraction process by simple clicking and sketching directly on the partial 3D point clouds (please refer to the accompanying video). Fig. 3 shows two examples of how users' guidance can help to obtain more complete and smooth feature curves.

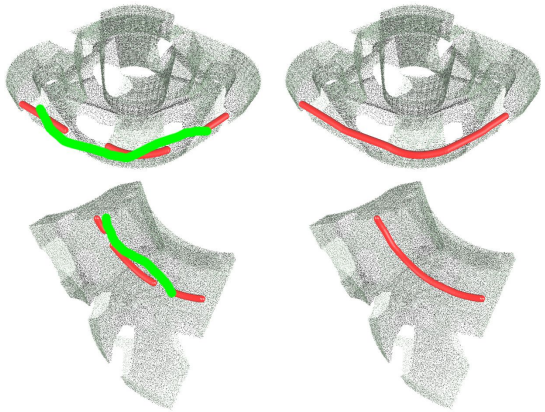

Fig. 3. Two discontinuous feature curves (left column) are smoothly connected by simple user strokes, yielding more complete feature curves (right column). Feature curves are in red and user strokes are in green.

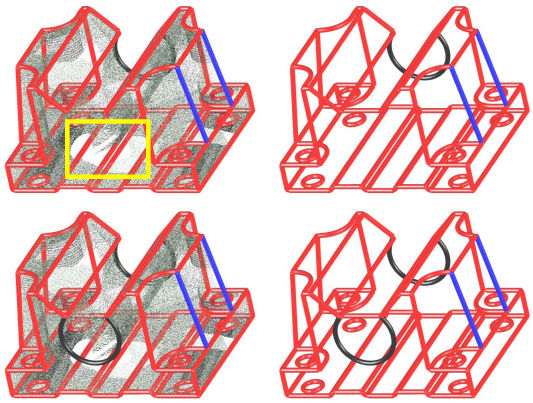

Fig. 4. A missing feature curve in the region marked by the yellow rectangle is generated from its mirror image (the black curve in the upper right subfigure) using reflective symmetry information. Left: feature curves overlaid on the point clouds. Right: feature curves only.

Since reflective symmetry is common for man-made objects, we allow the user to indicate if symmetry is applicable for the objects represented by point clouds. For partial point clouds, automatic symmetry detection is usually not reliable. Thus, we rely on the extracted feature curves (even though they are not complete) to determine the symmetry plane which in turn completes the feature curves based on reflective symmetry. Fig. 4 shows an example of completing the initial feature curves by symmetry.

4.2 Curved segment optimization

The initial curved segments extracted in the previous step have two problems. On the one hand, they are extracted

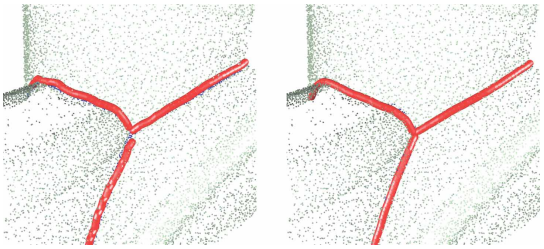

Fig. 5. Feature curve optimization without shape prior constraints. Left: The initial curved segments extracted in the previous step. Right: The optimized feature curves, which are smooth and are aligned well with the feature points (in blue).

from local sampled points, thus they are not accurate due to noise and outliers. On the other hand, they are not smooth enough for further reconstruction due to the nature of the polyline growing process. Thus, we propose to exploit prior knowledge and global structural regularities of the object to optimize the entire feature curves altogether. Our optimization encourages obtaining smooth, globally regular feature curves, and meanwhile respecting the input point cloud.

Let $\mathbf{B}_1, \mathbf{B}_2, ..., \mathbf{B}_m$ be the m initial curved segments, and each curve $\mathbf{B}_i$ is represented by n_i sequential discrete points $\mathbf{b}_{i,1}, \mathbf{b}_{i,2}, ..., \mathbf{b}_{i,n_i}$. The optimized position of points $\mathbf{b}_{i,j}$ is represented by $\overline{\mathbf{b}}_{i,j}$. We define the following energy terms for curve optimization:

- **Data fitting**. We use the data fitting term to ensure the optimized curved segments stay as close as possible to the 3D feature points detected from the point cloud. Mathematically, this term is define as the sum of the squared distance between a point $\overline{\mathbf{b}}_{i,j}$ in the feature curve and the 3D point in the input point cloud:

$$E_{alignment} = \sum_{i=1}^{m} \sum_{j=1}^{n_i} \|\overline{\mathbf{b}}_{i,j} - \mathbf{p}_{i,j}\|^2, \quad (2)$$

  where $\mathbf{p}_{i,j}$ is the variation weighted average of all the neighboring points of $\mathbf{b}_{i,j}$.
- **Smoothness**. This term encourages the curved segments to deform to be smooth. We define the non-smoothness F_{smooth} of the curved segments as following:

$$E_{smooth} = \sum_{i=1}^{m} \sum_{j=2}^{n_i-1} \|\overline{\mathbf{b}}_{i,j-1} - 2\overline{\mathbf{b}}_{i,j} + \overline{\mathbf{b}}_{i,j+1}\|^2. \quad (3)$$

- **Fidelity**. This term prevents the curved segments from deviating too much from their initial locations. It is defined as the sum of the squared distance of a point in the feature and its initial position:

$$E_{fidelity} = \sum_{i=1}^{m} \sum_{j=1}^{n_i} \|\overline{\mathbf{b}}_{i,j} - \mathbf{b}_{i,j}\|^2. \quad (4)$$

Then our objective function E is defined as the weighted sum of the above individual energy terms:

$$E = \omega_1 \cdot E_{fidelity} + \omega_2 \cdot E_{alignment} + \omega_3 \cdot E_{smooth}. \quad (5)$$

By minimizing the above energy function, the quality of the initially detected curved segments are improved. Fig. 5 shows an example of the optimization result without structural regularity constraints. As can be seen from this figure, the curved segments have been smoothed and are now better aligned with the detected 3D feature points in the point cloud.

We observe structural regularities (such as straight line segments, co-planarity, and symmetry) are common features in man-made objects. Thus, we exploit these high-level structural regularities to further improve the quality of the feature curves. In this work, the following structural regularities are detected and enhanced:

- **Linearity**. To straighten feature curves representing s-traight line segments, this term measures how well the interior points of a curved segment can be represented by the linear combination of its two endpoints:

$$E_{line} = \sum_{i \in \mathbf{S}_{line}} \sum_{j=1}^{n_i} \|t_{i,j} \cdot \overline{\mathbf{b}}_{i,1} + (1 - t_{i,j}) \cdot \overline{\mathbf{b}}_{i,n_i} - \overline{\mathbf{b}}_{i,j}\|^2, \quad (6)$$

  where $\mathbf{S}_{line}$ are the feature curves detected as straight lines. Parameter $t_{i,j}$ is the weight that can be computed by minimizing $\|t_{i,j} \cdot \mathbf{b}_{i,1} + (1 - t_{i,j}) \cdot \mathbf{b}_{i,n_i} - \mathbf{b}_{i,j}\|^2$ in the initial feature curves.
- **Circularity**. For a closed curve $\mathbf{B}_i$ representing a circle, the circularity term is defined to measure how far the closed curve is from being a perfect circle. We measure the non-circularity as the sum of the difference between the squared length of the circle's diameter r_i and the squared length of the segment from a point in the feature curve to the circle center $\mathbf{c}_i$:

$$E_{circle} = \sum_{i \in \mathbf{S}_{circle}} \sum_{j=1}^{n_i} (\|\overline{\mathbf{b}}_{i,j} - \mathbf{c}_i\|^2 - r_i^2). \quad (7)$$

  where $\mathbf{S}_{circle}$ are the feature curves detected as circles.
- **Co-planarity**. For all the curved segments $\mathbf{S}_{coplanar}$ that are supposed to be lying in the same plane, we first compute a plane $\mathbf{C}$ by least-squares fitting of the feature points. Then this term is defined as the sum of the squared distance between a point in the feature point and the plane:

$$E_{coplanar} = \sum_{i \in \mathbf{S}_{coplanar}} \sum_{j=1}^{n_i} (dist(\overline{\mathbf{b}}_{i,j}, \mathbf{C}))^2, \quad (8)$$

  where $dist(\overline{\mathbf{b}}_{i,j}, \mathbf{C})$ measures the distance from a point $\overline{\mathbf{b}}_{i,j}$ to the plane $\mathbf{C}$.
- **Symmetry and parallelism**. Given pairs of curves that are detected to be symmetric or parallel, for simple formulation and computation, we first perform a resampling step to ensure that the two curves are represented by the same number of points. Then, the symmetry constraint is defined as:

$$E_{symmetry} = \sum_{k} ((\frac{\overline{\mathbf{b}}_{i,k} + \overline{\mathbf{b}}_{j,k}}{2} - \mathbf{C}_{i,j}) \cdot \mathbf{n})^2, \quad (9)$$

  where $\mathbf{C}_{i,j} = \sum_k (\mathbf{b}_{i,k} + \mathbf{b}_{j,k})/2$ and $\mathbf{n}$ is the normalized vector of $\sum_k (\mathbf{b}_{i,k} - \mathbf{b}_{j,k})$.

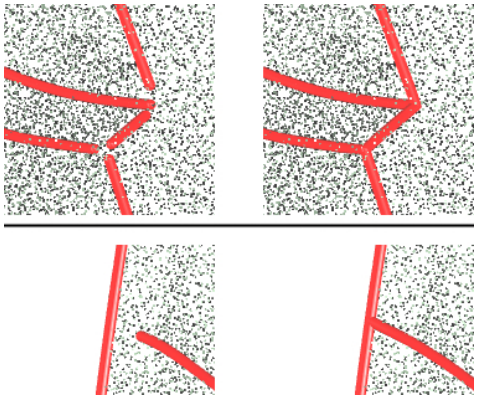

Fig. 6. Two types of feature curve completion. Top row: extending several feature curves to be connected at their endpoints. Bottom row: extending a feature curve to the interior of another one. Left column shows the initial curved segments, and right column are the extended feature curves.

Similarly, the parallelism constraint is given by:

$$E_{parallel} = \sum_k \|\bar{\mathbf{b}}_{i,k} - \bar{\mathbf{b}}_{j,k} + \mathbf{d}_{i,j}\|^2, \tag{10}$$

where $\mathbf{d}_{i,j} = \sum_k (\mathbf{b}_{j,k} - \mathbf{b}_{i,k})$.

In our implementation, we formulate the above structural regularities as soft constraints, and use Lagrange multipliers to enhance these regularities by minimizing the augmented energy function using the L-BFGS algorithm [28].

4.3 Curve network completion

After optimizing the curve segments, we now have regularized feature curves. Unfortunately, they are disconnected and can not be used for surface reconstruction. In this step, we propose a method to extend and close these feature curves to obtain well connected curve networks. This step is important for two purposes: 1) sharp corners of an object can be recovered after connecting several endpoints of the feature curves; 2) only closed curve networks can be used to detect surface patches for surface reconstruction.

Actually, it is not a difficult task to connect several endpoints of different feature curves if their endpoints are close to each other. Given several endpoints $\mathbf{p}_1, \mathbf{p}_2, ..., \mathbf{p}_k$ with corresponding tangent direction $\mathbf{t}_1, \mathbf{t}_2, ..., \mathbf{t}_k$, the connecting point (corner) $\mathbf{p}$ can determined by minimizing the following function,

$$\sum_{i=1}^{i=k} (\mathbf{p} - \mathbf{p}_i) \cdot \mathbf{t}_i. \tag{11}$$

The top row of Fig. 6 shows an example for automatically connecting several endpoints of different feature curves. In order to connect an endpoint $\mathbf{p}_1$ of a feature curve to the interior of another curve $\mathbf{B}_i$, let $\mathbf{t}_1$ denote the tangent direction at $\mathbf{p}_1$, we extend $\mathbf{p}_1$ to the point $\mathbf{p}$ on curve $\mathbf{B}_i$ minimizing $(\mathbf{p} - \mathbf{p}_1) \cdot \mathbf{t}_1$. The bottom row of Fig. 6 shows such an example.

For endpoints that are far away from each other (this is typically true for point clouds with significant missing regions), the main task for the feature curve completion problem is to determine which curves can be connected together. We solve this problem based on a clustering method with a cost function defined as below. For each endpoint $\mathbf{p}_i$ of a feature curve, we define the cost of connecting it with other point $\mathbf{q}_i$ from another curved segment as:

$$F(\mathbf{p}_i, \mathbf{q}_i) = \frac{dist(\mathbf{p}_i, \mathbf{q}_i)/s_{max}}{2 + \cos(\theta(\mathbf{p}_i, \mathbf{q}_i))}, \tag{12}$$

where $dist(\mathbf{p}_i, \mathbf{q}_i)$ is the distance between points $\mathbf{p}_i$ and $\mathbf{q}_i$. If the point $\mathbf{q}_i$ is also an endpoint, then $\theta(\mathbf{p}_i, \mathbf{q}_i)$ is the angle between the tangent directions at these two points. If the point $\mathbf{q}_i$ is a sample point in the interior of another feature curve, then $\theta(\mathbf{p}_i, \mathbf{q}_i)$ is the angle between the tangent direction of $\mathbf{p}_i$ and direction of the vector $\mathbf{q}_i - \mathbf{p}_i$.

For each endpoint $\mathbf{p}_i$, a point $\mathbf{p}_j$ with minimum cost $F(\mathbf{p}_i, \mathbf{p}_j) < \lambda$ from other feature curves are progressively clustered together. The threshold λ is set to 0.9 in our experiment. By clustering each endpoint with the best points on other curves, we get initial clusters $K_1, K_2, ..., K_m$, where K_i is composed of points $\mathbf{p}_{i,1}, \mathbf{p}_{i,2}, ..., \mathbf{p}_{i,n_i}$. We define another cost to merge two clusters such that two endpoints $\mathbf{p}_{i,s}$ in K_i and $\mathbf{p}_{j,t}$ in K_j are merged:

$$E(K_i, K_i) = \min_{\mathbf{p}_{i,s} \in K_i, \mathbf{p}_{j,t} \in K_j} F(\mathbf{p}_{i,s}, \mathbf{p}_{j,t}). \tag{13}$$

We iteratively merge the closest clusters as long as the cost defined by Equation 13 is smaller than the threshold λ. We continue this process until no more cluster pairs can be merged. Fig. 7 shows one of the feature curve completion results using the proposed clustering method.

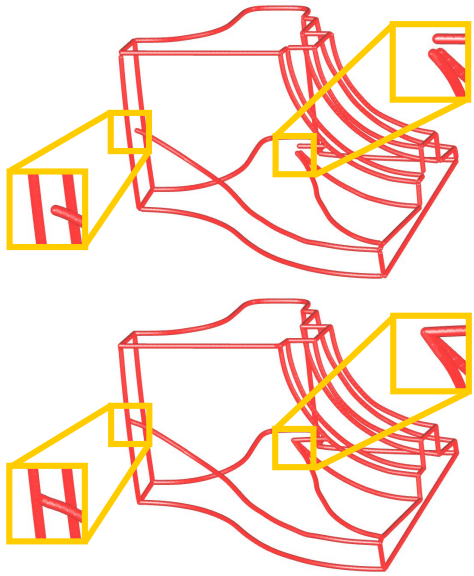

Fig. 7. Result of connecting several endpoints of curves

5 Results and Discussion

We tested our algorithm on a large set of noisy point clouds (both real laser scans and synthetic data) with large missing regions in the neighborhood of the feature curves.

Curve network results. Fig. 8 shows the curve network extraction and completion results for three free-form surface

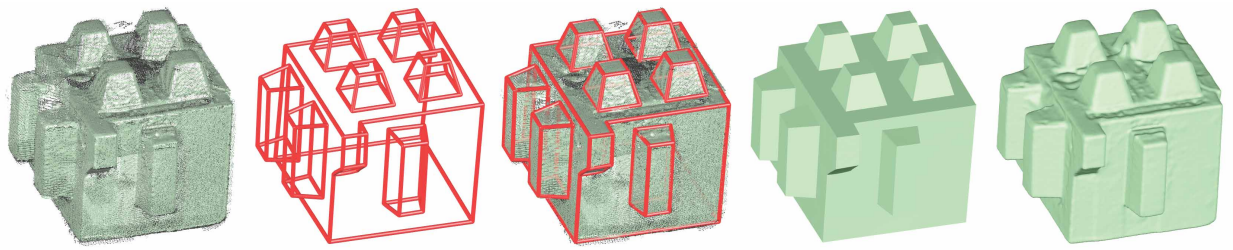

Fig. 9. A mechanical part reconstructed from the curve network extracted from its noisy laser scan. From left to right: input point cloud, extracted curve network, the curve network overlaid on the point cloud, reconstructed surface model from the curve networks using the algorithm proposed by Zhuang et al. [1], and the surface model reconstructed using the Screened Poisson method [29].

Fig. 8. Curve networks extracted from point clouds of objects with free-form surfaces.

objects. As can be seen from this figure, although large holes occur at the feature regions of the point clouds, our method can still produce satisfactory curve networks.

Fig. 9 shows the processing of a mechanical part. Despite the noise in the input laser scan, our method successfully extracts the curve network from the point cloud. We can also see that the sharp features are faithfully preserved in the reconstructed surface model using our curve network as input.

In Fig. 10, we demonstrate a collection of curve networks and the corresponding surface models reconstructed from these curve networks. Similar to Figures 8 and 9, the examples shown here are all partial point clouds, but are mechanical parts. The first column are noisy point clouds with missing data near the feature regions. The second column are the curve networks extracted from the point clouds by our approach. The third column are the point clouds overlaid on the extracted curve networks. It can be seen clearly from this column, the extracted curve networks coincide well with the curved features of the point clouds. The fourth column shows the reconstructed surface models from the curve networks using the algorithm proposed by Zhuang et al. [1]. Although some regions in the point clouds are missing (especially those at/near the feature regions), the surface models are faithfully reconstructed from the extracted curve networks.

As a comparison, we show the reconstruction results from these point clouds using the Screened Poisson reconstruction method [29] in the last column. It is obvious that the Poisson method can not reconstruct faithful surface models to fill large holes in the point clouds, and sharp features in the objects are usually smoothed. In contrast, the reconstruction from our extracted curve networks successfully recover these sharp features.

Limitations. One limitation of our method is that it is difficult to extract very small features from the point clouds. It is also difficult to extract the feature curves that are very close to each other. In such a case, our method may not be able to separate these feature curves.

Another limitation is that we still can not handle very large missing regions. Although we developed a simple user interface to guide the feature curve completion for the regions with missing data. We found that it is still too difficult to characterize such curve features by a simple interpolation of the points.

6 Conclusions and Future Work

In this paper, we presented a framework for extracting curve networks from noisy and partial point cloud data. Our method can automatically extract and complete most of the feature curves. We exploited structural regularities to enhance the extracted curve networks to be regular and meanwhile respect the input point clouds. To resolve ambiguities for point clouds with large missing regions, we developed a simple user interface that allows the user to guide the feature extraction and completion. Experiments on various imperfect point clouds validated the effectiveness of our curve networks extraction framework. The reconstructed surface models from our curve networks confirmed that the problem of reconstruction from partial point clouds can be significantly regularized by using the curve networks extracted using our method.

In the future, we plan to exploit the extracted curve networks for further editing of the reconstructed surface models.

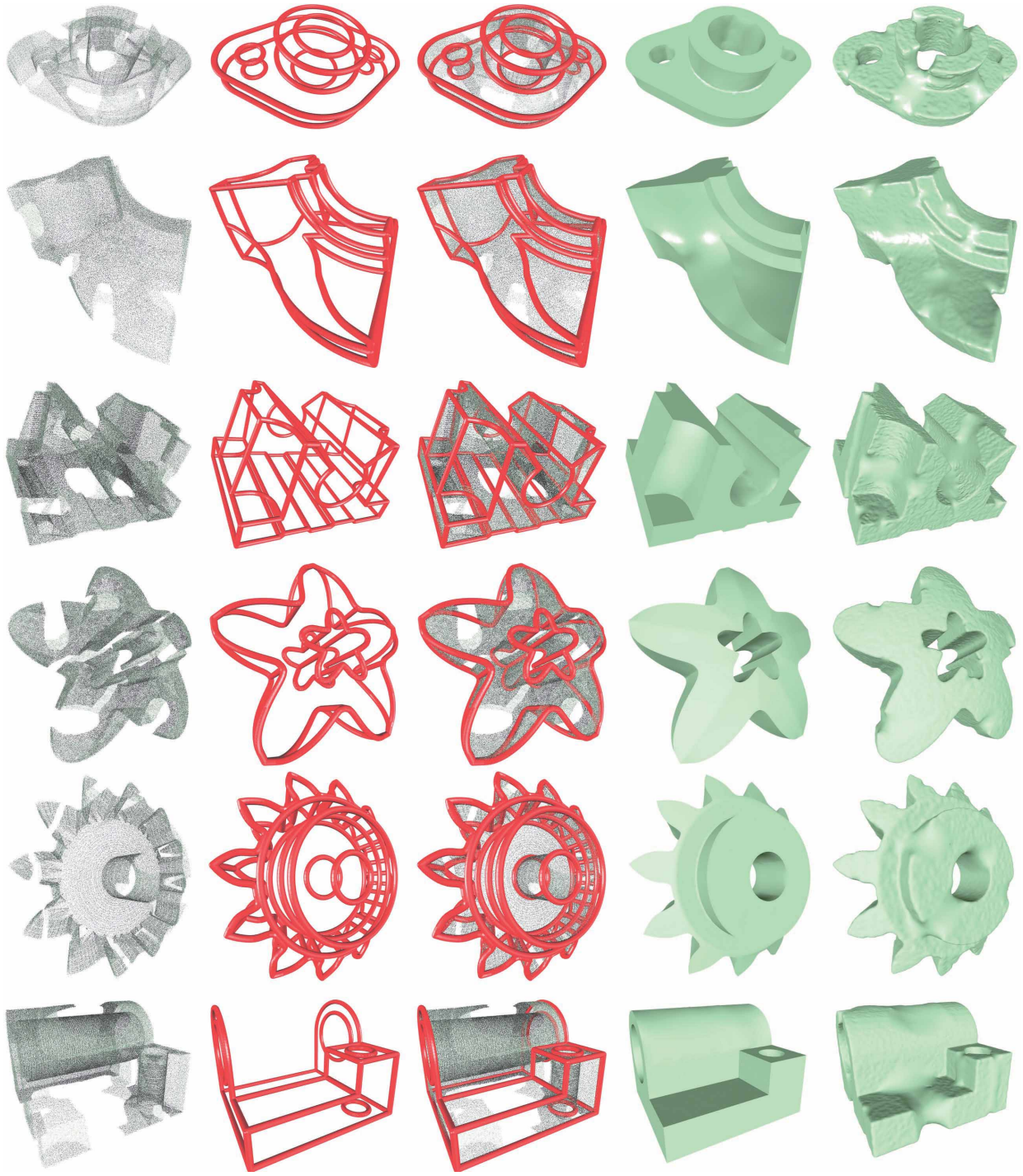

Fig. 10. Curve network extraction and surface reconstruction from a set of synthetic data. From left to right: input point clouds, extracted curve networks, curve networks overlaid on the point clouds, reconstructed surface models from the curve networks using the algorithm proposed by Zhuang et al. [1], surface models reconstructed using the Screened Poisson method [29].

ACKNOWLEDGEMENTS

We gratefully acknowledge the support of NVIDIA Corporation with the donation of the Quadro K52000 GPU used for this research. This work was supported by the KAUST Visual Computing Center.

REFERENCES

[1] Y. Zhuang, M. Zou, N. Carr, and T. Ju, "A general and efficient method for finding cycles in 3d curve networks," *ACM Trans. Graph.*, vol. 32, no. 6, pp. 180:1–180:10, Nov. 2013.

[2] T. Saito and T. Takahashi, "Comprehensible rendering of 3-d shapes," in *Proceedings of the 17th Annual Conference on Computer Graphics and Interactive Techniques*, ser. SIGGRAPH '90. New York, NY, USA: ACM, 1990, pp. 197–206.

[3] B. Xu, W. Chang, A. Sheffer, A. Bousseau, J. McCrae, and K. Singh, "True2form: 3d curve networks from 2d sketches via selective regularization," *Transactions on Graphics (Proc. SIGGRAPH 2014)*, vol. 33, no. 4, 2014.

[4] R. Mehra, Q. Zhou, J. Long, A. Sheffer, A. Gooch, and N. J. Mitra, "Abstraction of man-made shapes," in *ACM Transactions on Graphics (TOG)*, vol. 28, no. 5. ACM, 2009, p. 137.

[5] M. Nieser, C. Schulz, and K. Polthier, "Patch layout from feature graphs," *Comput. Aided Des.*, vol. 42, no. 3, pp. 213–220, Mar. 2010.

[6] F. Abbasinejad, P. Joshi, and N. Amenta, "Surface patches from unorganized space curves," *Computer Graphics Forum (Symp. on Geometry Proc.)*, vol. 30, no. 5, pp. 1379–1387, 2011.

[7] R. Gal, O. Sorkine, N. J. Mitra, and D. Cohen-Or, "iwires: an analyze-and-edit approach to shape manipulation," in *ACM Transactions on Graphics (TOG)*, vol. 28, no. 3. ACM, 2009, p. 33.

[8] A. J. Law and D. G. Aliaga, "Single viewpoint model completion of symmetric objects for digital inspection," *Computer Vision and Image Understanding*, vol. 115, no. 5, pp. 603–610, 2011.

[9] Y. Lee and S. Lee, "Geometric snakes for triangular meshes," *Computer Graphics Forum (EUROGRAPHICS)*, vol. 21, no. 3, pp. 299–320, 2002.

[10] Y. Ohtake, A. Belyaev, and H.-P. Seidel, "Ridge-valley lines on meshes via implicit surface fitting," *ACM TOG (SIGGRAPH)*, vol. 23, no. 3, pp. 609–612, 2004.

[11] S.-K. Kim and C.-H. Kim, "Finding ridges and valleys in a discrete surface using a modified mls approximation." *Computer-Aided Design*, vol. 38, no. 2, pp. 173–180, 2006.

[12] S. Yoshizawa, A. Belyaev, and H.-P. Seidel, "Fast and robust detection of crest lines on meshes," in *Proc. of ACM Symposium on Solid and Physical Modeling*, 2005, pp. 227–232.

[13] M. Nomura and N. Hamada, "Feature edge extraction from 3D triangular meshes using a thinning algorithm," in *Society of Photo-Optical Instrumentation Engineers (SPIE) Conference Series*, vol. 4476, 2001, pp. 34–41.

[14] M. K. Park, S. J. Lee, and K. H. Lee, "Multi-scale tensor voting for feature extraction from unstructured point clouds," *Graphical Models*, vol. 74, no. 4, pp. 197–208, 2012.

[15] R. K. Mark Pauly and M. Gross, "Multi-scale feature extraction on point-sampled surfaces." *Computer Graphics Forum*, vol. 22, no. 3, pp. 281–289, Sep. 2003.

[16] J. D. II, T. Ochotta, L. K. Ha, and C. T. Silva, "Spline-based feature curves from point-sampled geometry," *Visual Comput*, vol. 24, pp. 449–462, 2008.

[17] S.-H. Bae, R. Balakrishnan, and K. Singh, "Ilovesketch: As-natural-as-possible sketching system for creating 3d curve models," in *Proceedings of the 21st Annual ACM Symposium on User Interface Software and Technology*, ser. UIST '08. New York, NY, USA: ACM, 2008, pp. 151–160.

[18] T. Igarashi, S. Matsuoka, and H. Tanaka, "Teddy: A sketching interface for 3d freeform design," in *Proceedings of the 26th Annual Conference on Computer Graphics and Interactive Techniques*, ser. SIGGRAPH '99. New York, NY, USA: ACM Press/Addison-Wesley Publishing Co., 1999, pp. 409–416.

[19] A. Nealen, T. Igarashi, O. Sorkine, and M. Alexa, "Fibermesh: Designing freeform surfaces with 3d curves," *ACM Trans. Graph.*, vol. 26, no. 3, Jul. 2007.

[20] G. Orbay and L. B. Kara, "Sketch-based modeling of smooth surfaces using adaptive curve networks," in *Proceedings of the Eighth Eurographics Symposium on Sketch-Based Interfaces and Modeling*, ser. SBIM '11, 2011, pp. 71–78.

[21] M. Bessmeltsev, C. Wang, A. Sheffer, and K. Singh, "Design-driven quadrangulation of closed 3d curves," *ACM TOG (SIGGRAPH Asia)*, vol. 31, no. 5, 2012.

[22] M. Zou, T. Ju, and N. Carr, "An Algorithm for Triangulating Multiple 3D Polygons," *Computer Graphics Forum*, vol. 32, no. 5, pp. 157–166, 2013.

[23] T. Várady, A. Rockwood, and P. Salvi, "Transfinite surface interpolation over irregular n-sided domains," *Computer-Aided Design*, vol. 43, no. 11, pp. 1330–1340, 2011.

[24] L. Nan, A. Sharf, and B. Chen, "2d-d lifting for shape reconstruction," *Computer Graphics Forum*, vol. 33, no. 7, pp. 249–258, 2014.

[25] K. Demarsin, D. Vanderstraeten, T. Volodine, and D. Roose, "Detection of closed sharp edges in point clouds using normal estimation and graph theory," *Computer-Aided Design*, vol. 39, no. 4, pp. 276–283, 2007.

[26] Y. Cao, D.-M. Yan, and P. Wonka, "Patch layout generation by detecting feature networks," *Computer & Graphics*, vol. 46, pp. 275–282, 2015.

[27] J. Daniels II, T. Ochotta, L. K. Ha, and C. T. Silva, "Spline-based feature curves from point-sampled geometry," *The Visual Computer*, vol. 24, no. 6, pp. 449–462, 2008. [Online]. Available: http://dx.doi.org/10.1007/s00371-008-0223-2

[28] D. C. Liu and J. Nocedal, "On the limited memory bfgs method for large scale optimization," *Mathematical programming*, vol. 45, no. 1-3, pp. 503–528, 1989.

[29] M. Kazhdan and H. Hoppe, "Screened poisson surface reconstruction," *ACM Trans. Graph.*, vol. 32, no. 3, pp. 29:1–29:13, Jul. 2013.