Extracting Cycle-aware Feature Curve Networks from 3D Models

Zhengda Lu[a,b], Jianwei Guo[b,a,*], Jun Xiao[a,*], Ying Wang[a], Xiaopeng Zhang[b,a], Dong-Ming Yan[b,a]

[a]School of Artificial Intelligence, University of Chinese Academy of Sciences, Beijing 100049, China
[b]National Laboratory of Pattern Recognition, Institute of Automation, Chinese Academy of Sciences, Beijing 100190, China

Abstract

Meaningful feature curves provide high-level shape representation of the geometrical shapes and are useful in various applications. In this paper, we propose an automatic method on the basis of the quadric surface fitting technique to extract complete feature curve networks (FCNs) from 3D surface meshes, as well as finding cycles and generating a high-quality segmentation. In the initial collection of noisy and fragmented feature curves, we first fit the quadric surfaces of each curve and the corresponding neighbor vertices to filter out non-salient or noisy feature curves. Then we conduct a feature extension step to address the curve intersections and form a closed FCN. Finally, we regard circle curves as cycles in the complete FCN and segment the mesh into patches to reveal a highly structured representation of the input geometry. Experimental results demonstrate that our algorithm is more robust for FCN extraction from complex input meshes and achieves higher quality patch layouts compared with the state-of-the-art approaches. We also verify the validity of extracted feature curve cycles by applying them to surface reconstruction.

Keywords: Shape analysis, Feature curve network, Segmentation.

1. Introduction

Nowadays 3D models are commonly represented by triangular meshes, which can be easily obtained via various acquisition techniques for 3D geometry. Feature curve network (FCN) extraction is one of the fundamental problems in computing compact and faithful representations of 3D objects in geometry processing. Extracting meaningful FCNs is useful in a variety of downstream applications, such as reverse engineering [1], shape abstraction [2, 3], remeshing [4, 5], shape modeling [6, 7], and functional map computation [8], to name a few.

The feature curve here is defined as the intersection of two geometric patches on the surface and FCN reveals the shape's key structural parts. The goal of feature curve extraction is to preserve a set of connected segments that are not necessarily the original mesh edges, which convey the maximum number of essential characteristics of 3D shapes. Existing methods have addressed FCN extraction from two different aspects. Although shape approximation approaches [9, 10] can segment models to meaningful spatial parts, these methods are based on vertex clustering or boundary detection techniques to determine patches that meet certain regularity criteria. However, their greedy nature tends to create redundant boundaries within weak feature regions and obtain unsatisfactory results. Geometry-driven approaches produce detailed FCNs by extracting feature lines and fitting local geometric primitives. The generated feature curves are typically not connected in the regions where multiple surfaces intersect due to the limitation of local shape properties.

Techniques for detecting crest lines typically use local shape properties (e.g., curvature and normal voting tensor) to extract ridge and valley lines [11, 12], and then the number of detected feature curves are controlled using a filtering operation. Directly using the detected feature curves for further analysis and applications is difficult because of their distribution on the model surfaces. To form a closed and efficient FCN, user interaction is required to connect the curves by using several strokes. However, this is a challenging task for ordinary users.

In this study, we propose an automatic approach on the basis of the quadric surface fitting technique to extract complete FCNs from 3D models, as well as finding cycles and generating high-quality segmentation results. As shown in Fig. 1, we first present a valid method for filtering out non-salient feature curves by combining the curvature and surface fitting error. Thus, we can avoid determining high-frequency noise and weak features as false positives. Then, we address the curve intersections and missing curve by applying inward and outward feature extensions to create a connected 3D FCN. Finally, we can easily find cycles of connected curves that bound surface patches. Our approach can achieve high-quality FCNs and patch layouts. In summary, the main contributions of this work are the following:

- We provide an efficient method for filtering the noise curves from the initial curve network to preserve meaningful feature curves and extract complete FCNs from 3D models.
- We simultaneously find the curve cycles in the network and generate high-quality patch layouts compared with the state-of-the-art segmentation approaches.
- We obtain a compact high-level representation of 3D models that can be used for surface reconstruction.

With respect to our previous contribution [13], we have extended the following aspects:

(1) Improved feature extension by introducing the concept of node point for long curve fitting optimization. Since the extension on a long curve composed of several short curves is not satisfactory in [13], we find the node point which is the intersection of more than two adjacent surfaces on the curve, and divide it into short curves to achieve a better surface fitting and extension effect.

(2) A new method for patch layout generation by detecting the curve cycles. We find the curve cycles in the network after curve

*Corresponding author
Email addresses: jianwei.guo@nlpr.ia.ac.cn (Jianwei Guo), xiaojun@ucas.ac.cn (Jun Xiao)

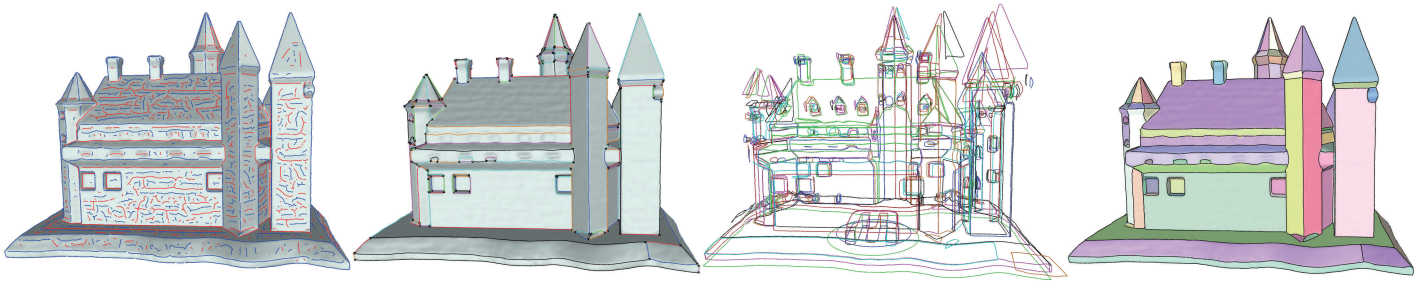

Figure 1: Our system takes as input a 3D surface mesh and fragmented feature curves and outputs a complete FCN. From the curve network, we detect cycles of connected curves that bound surface patches. From left to right: the input, extracted network, exploded view of detected cycles, and patch layout.

breaking and merging operation and we take a correlation detection based on surface fitting for small patches to solve detailed problems caused by the feature extension process. Eventually, we can generate high-quality patch layouts and feature curve network results.

(3) Extensive experiments are conducted to verify the effectiveness of our approach. We have evaluated our method on more complex 3D models with or without ground-truth segmentation results, studied the resistance to noise, and conducted an ablation study of feature curve filtering. We have also compared our method with respect to state-of-the-art methods as well as our previous work. Finally, we demonstrated the validity of the extracted feature curve cycles by applying it to surface reconstruction.

2. Related work

Our technique aims to construct shape abstraction FCNs from 3D models. We present a brief overview of the various techniques related to feature detection and FCN extraction.

Feature detection. Many studies have been conducted on detecting feature lines in 3D models. A complete survey on feature extraction is beyond the scope of this paper. Hence, we will only describe a few representatives.

Several approaches are based on the detection of ridges and valleys, which are the loci of points where the positive (negative) variation of local properties such as normal and curvatures in the direction of their maximal change attains a local maximum (minimum). These properties can be calculated through global [11] or local [12, 14] surface fitting. Ohtake et al. [11] combine multi-level implicit surface fitting and finite difference approximations to extract accurate ridges and valleys, while [12, 14] show faster computation by using the local estimation techniques to calculate the local differential information. Lai et al. [15] extract ridges and valleys via integral invariants of local neighborhoods without approximation. Since the ridges and valleys are the extreme points set of local attributes, the generated feature curves usually contain many non-salient curves and are not connected in the regions where multiple surfaces intersect.

Other types of line features have also been proposed by considering local geometry of the normal or curvature. Nomura and Hamada [16] extract feature edges between the convex and concave triangles based on normals. Weinkauf and Günther [17] extract salient edges by the topological analysis of principal curvatures. Due to the local geometry is affected where more than two geometric surfaces intersect, these methods cannot detect continuous feature curves. Kim et al. [18] classify features into corners, sharp edges, and faces based on the normal tensor voting. But they may create redundant feature lines at complex transition surfaces. Several approaches [19, 20, 21] based on symmetry detection have been proposed to extract feature curve templates and complete missing data in noisy FCNs. However, some important curves are still missed because they can only find the reoccurrences of curve templates.

Despite promising results, these methods usually can not detect features on smooth and transition regions due to the limitation of local surface properties. In contrast, we retain a maximal set of salient feature curves by combining the curve curvature and surface fitting error.

Curve network extraction. Many algorithms extract patch boundaries from segmentation results for the generation of FCN. Cohen-Steiner et al. [9] proposed the Variational Shape Approximation(VSA) to partition faces to planar patches based on Lloyd's clustering. Yan et al. [10] extend the VSA framework to the sphere, cylinder, and other quadric patches. Nieser et al. [1] grow patches around seeds where the maximal curvature of a set of neighboring faces lies below a predefined threshold and thicken the edges of the feature graph. Yamakawa et al. [22] group triangles using a region-growing technique and extract feature edges along the boundaries between the triangle groups. Mehra et al. [2] generate hierarchical abstraction curve networks by VSA and simplified by thresholding the reconstruction error based on the curve smoothness and the normal deviation of the resulting mesh approximation. De Goes et al. [3] define the concept of *exoskeletons* to abstract the 3D model and reveal meaningful perceptual parts, which are refined by employing VSA. Gehre et al. [23] extract a scale conforming FCN, which preserves the maximum set of prominent feature curves within a prescribed scale from an initial set of feature lines. However, the initial curve is discontinuous. Torrente et al. [24] adopt the Hough transform to identify and localize feature curves on 3D shapes. Sharma et al. [25] propose the ParSeNet to decompose a 3D point cloud into parametric surface patches. These patch layouts are related to the number of clusters and usually do not trace out weak features to generate a detailed FCN. Hu et al. [26] use the multiscale patch clustering to learn contextual information and segment 3D point clouds of indoor scenes into the meaningful objects. One close work to ours is that of Cao et al. [27], who take the curve length and curvature into account by filtering a dense set of feature curves for salient and long curves to generate the surface patches. However, this work, as well as other existing approaches, perform FCN extraction by preserving small scale details and thresholding curvature based on local surface properties. They are unable to get accurate feature curves within transition regions with weak features. We will show our superiority by comparing it to them.

Surfacing curve networks. Creating surfaces from 3D curve networks is an important research topic in shape modeling and

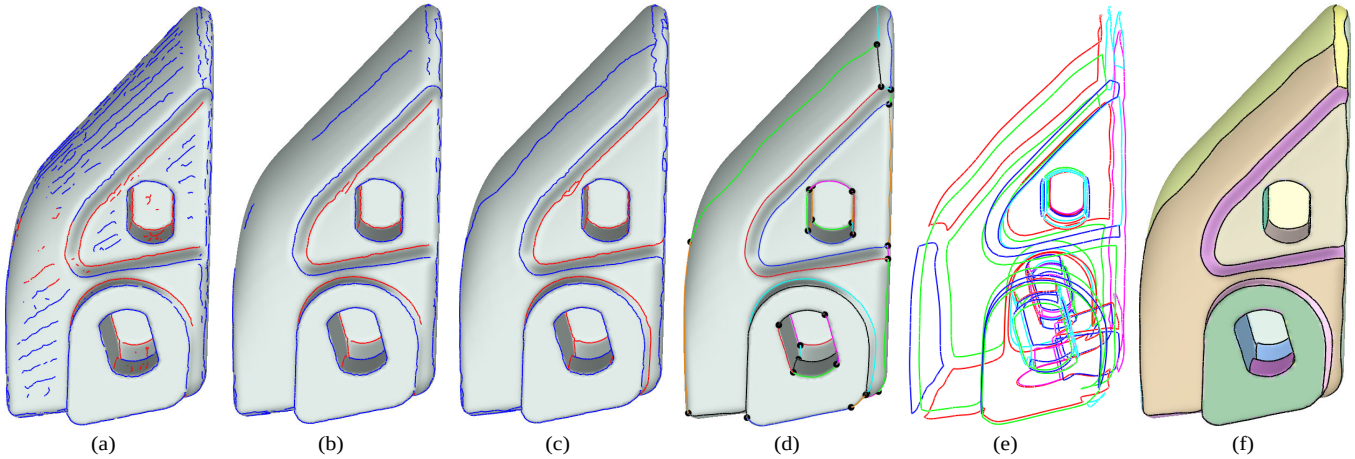

Figure 2: The pipeline of our framework. (a) input mesh and feature curves; (b) salient feature curves after filtering; (c) feature curve extension result; (d) final FCN; (e) exploded view of cycles; (f) patch layout after segmentation and boundary smoothing, where different patches are shown in different colors.

computer-aided design (CAD). This task promotes the recovery of full geometry from a set of curves spanning the surfaces. Nealen et al. [28] present a system for automatically constructing smooth surfaces by applying functional optimization with a collection of 3D curves. Liu et al. [29] produce a closed surface network that interpolates curve networks with arbitrary shapes and topologies on cross-sections. Orbay and Kara [30] propose a sketch-based modeling interface for creating smooth surfaces from curve networks. Abbasinejad et al. [31] introduce a system that supports the automatic generation of piecewise smooth surfaces from curve networks on the basis of linear algebra representation of surface patches. With similar motivation, Bessmeltsev et al. [32] present a design-driven approach for quadrangulating closed 3D curve networks. Zou et al. [33] present an algorithm for triangulating 3D spatial polygons. Zhuang et al. [34] extract a set of curve cycles in FCNs and produce smooth surfaces that interpolate the input 3D curves or cycles. Xu et al. [35] design a sketch-based modeling system (i.e., True2Form) that reconstructs 3D curve networks from typical 2D design sketches. Pan et al. [6] create freeform surfaces by exploiting the geometric constraints of curvature flow lines. Stanko et al. [36] construct triangle meshes with the solution of a new variational optimization method based on mean curvature vectors. Our results can be used as a valid input of 3D curves for these applications.

3. Overview

The input to our algorithm is a two-manifold triangular mesh $\mathcal{M}$ consisting of a set of triangle facets $\{t_i\}_{i=1}^{m}$ and vertices $\{\mathbf{v}_i\}_{i=1}^{q}$, and an initial set of fragmented feature curves, which are computed by the method of Yoshizawa et al. [12] due to its robustness. Our goal is to extract complete FCNs and generate high-quality patch layouts. Note that we first apply a remeshing procedure if the input is a low-resolution mesh, where the target edge length of the new mesh is computed on the basis of the total surface area of the input mesh (more details can be found in [37]).

The main steps of our algorithm are illustrated in Fig. 2. First, we filter out the short and unimportant feature elements on the basis of the quadric surface fitting technique. Second, we map the feature points in the remaining curves to the corresponding closest vertices in the mesh and conduct a feature extension step to form a complete FCN. Finally, we find cycles in the complete FCN and obtain patch layouts to segment the input mesh, and refine the curve network boundary by applying a smoothing operation [1]. We explain the details of each step in the next section.

4. Methodology

4.1. Feature Curve Filtering

We utilize the method in [12] to compute a set of initial curves automatically, that is, ridge and valley lines. Each feature curve c, is represented by a set of ordered feature points, $\{\mathbf{p}_0, \mathbf{p}_1, \cdots, \mathbf{p}_n\}$, on triangle edges (see Fig. 3 (a)). However, the initially generated curves typically contain many short and non-salient curves located on the geometric surfaces of the mesh due to the limitation of ridge and valley lines. We present an efficient approach to filter out these non-salient curves by combining the curve curvature and surface fitting error.

Given a curve $c = \{\mathbf{p}_0, \mathbf{p}_1, \cdots, \mathbf{p}_n\}$, the feature points are generally not the vertices of the mesh. To simplify our calculation, we map these feature points to the closest mesh vertices $\{\mathbf{v}_0, \mathbf{v}_1, \cdots, \mathbf{v}_n\}$. We consider the curvature of mapped vertices as the curvature of feature points. The curvature R for each feature curve can be computed as follows:

$$\begin{aligned} R &= \frac{\sum_{i=0}^{n} C_{RMS}(\mathbf{p}_i)}{n}, \\ C_{RMS}(\mathbf{p}_i) &= \sqrt{k_{max}^2(v_i) + k_{min}^2(v_i)}, \end{aligned} \tag{1}$$

where $C_{RMS}(\mathbf{p}_i)$ is the root mean square curvature of $\mathbf{p}_i$, and k_{max} and k_{min} are the maximal and minimal principal curvatures of the vertex $\mathbf{v}_i$ which is nearest to $\mathbf{p}_i$. We use R to filter the non-salient curves on the plane. We observe that the majority of salient curves are located in the regions where two adjacent geometric surfaces intersect. Since the R of some curve lying on the quadric surface is larger than certain intersecting curves, we cannot avoid filtering these intersecting curves when we only use the curve curvature. Therefore, we propose to use the surface fitting error, E, to determine if one curve is the intersecting curve of the adjacent surfaces. Our motivation for using fitting error is that we can fit a proper surface inside the mesh. Otherwise, we could observe a large fitting error on the local region of the intersection curve of two surfaces.

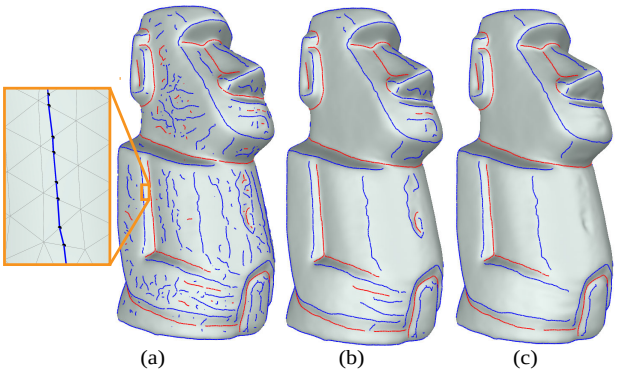

Figure 3: Illustration of our filtering process using R and E. (a) input mesh and feature curves. (b) use parameter $R = 0.005$ to filter feature curves. (c) use the parameter $E = 3 * 10^{-6}$ to filter other feature curves.

Hence, we adopt the concept of quadric surface fitting [10]. Since each feature curve cuts across the triangles of $\mathcal{M}$, we collect these triangles as the initial seeding set $\{t_i\}$. We then visit each triangle in $\{t_i\}$, and add their adjacent triangles to $\{t_i\}$ for three-iterations. Accordingly, we can form the local surface region, $\mathcal{P} = \{t_i\}_{i=1}^{n}$, around this feature curve. We subsequently fit the general quadric surface (including plane, sphere, cylinder, etc.) for $\mathcal{P}$. The implicit equation of the quadric surface $f(\mathbf{x}) = 0$, can be expressed as

$$f(\mathbf{x}) = \mathbf{C}^{\mathrm{T}} \cdot \mathbf{F}, \tag{2}$$

where $\mathbf{x} = [x, y, z]^{\mathrm{T}} \in \mathbb{R}^3$ is a 3D point, $\mathbf{C} = [c_0, c_1, ..., c_9]^{\mathrm{T}}$ is the vector of unknown coefficients for the quadric surface and $\mathbf{F} = [1, x, y, z, xy, xz, yz, x^2, y^2, z^2]^{\mathrm{T}}$. Both of them are ten dimensional vectors.

To compute $f(\mathbf{x})$, we minimize the fitting error between the local surface region and the quadric surface. The fitting error E is defined by combining L^2 and $L^{2,1}$ distance between a triangle t_i and $f(\mathbf{x})$:

$$E(\mathcal{P}) = \frac{1}{n}\sum_{i=1}^{n} E(t_i, f) = \frac{1}{n}\sum_{i=1}^{n} (E_d(t_i, f) + \omega E_n(t_i, f)), \tag{3}$$

where E_d measures the squared Euclidean distance (L^2), E_n measures the normal deviation ($L^{2,1}$) from t_i to f, and ω is the balance weight ($\omega = 0.5$ by default). In our implementation, E_d and E_n are approximated by the following formula respectively:

$$\begin{aligned} E_d(t_i, f) &= \int_{t_i} \frac{f(\mathbf{x})^2}{|\nabla f(\mathbf{x})|^2} \cdot d\sigma, \\ E_n(t_i, f) &= \int_{t_i} \left(\frac{\nabla f(\mathbf{x})}{|\nabla f(\mathbf{x})|} - \mathbf{n}_{t_i}\right)^2 \cdot d\sigma, \end{aligned} \tag{4}$$

where $\mathbf{n}_{t_i}$ is the unit normal vector of t_i.

The detailed optimization algorithm used to fit the quadric can be found in [10]. In short, we first compute an initial quadric surface using the algorithm [38] which only takes the L^2 component into account. Then we reduce the optimization to a linear least square problem by computing the gradient for each triangle at the barycenter and use this one point approximation of the gradient in Eq. 4. We solve the least square fitting by Cholesky decomposition.

Since the calculation of surface fitting error is time-consuming, we first compute the R measurements for each curve to filter out most of the non-salient curves located on the flat areas. Then, we compute the surface fitting error E for the other curves. If the average error E of a curve is smaller than a prespecified threshold, the curve is located inside the quadric surface and must be filtered out. The result of this step is illustrated in Fig. 3.

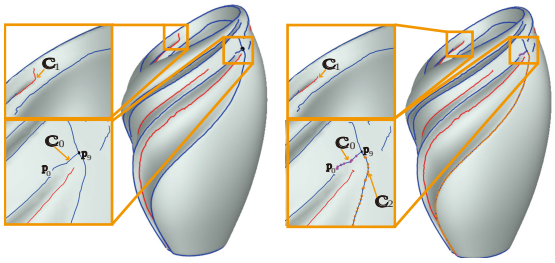

Figure 4: An example of curve optimization. Left: the input mesh with filtered feature curves. Right: result after inward extension.

4.2. Feature Curve Network Extraction

After filtering, the preserved feature curves capture the boundary information of geometric patches in the input mesh. However, these curves computed by [12] are not connected in regions where multiple surfaces intersect and cannot be directly regarded as the result of the FCN. Therefore, we use the surface fitting equation and curvature direction to extend the preserved curves and obtain the completed curve network.

Curve optimization. We observe that there exist some long feature curves that actually consist of several shorter curves. For example, in Fig. 4, the curve $c_0 = \{\mathbf{p}_0, \cdots, \mathbf{p}_9, \cdots\}$ should be split into two curves at feature point $\mathbf{p}_9$. We denote feature point $\mathbf{p}_9$ as the *node point* which is the intersection of more than two adjacent surfaces. Note that a node point is similar to the notion of a corner in some other work. The principal curvature direction of such a node point deviates from its adjacent feature points more than 30°. First, we use the feature points and their normals of a curve to fit a quadric surface and compute the average fitting error $E(c)$. If the fitting error $E(c)$ is larger than the threshold α (1.5 times filtering parameter E) or the number of feature points is more than the threshold τ ($\tau = 50$ by default), then we consider that this curve may be a long curve and contains one or more nodes. Second, we use an inward tracing scheme to find the node points. As shown in Fig. 4, starting from the endpoint $\mathbf{p}_0$, we repeatedly visit the successive feature point until we encounter the node point $\mathbf{p}_9$. We collect these feature points as similar points for the endpoint $\mathbf{p}_0$ and form a new curve. However, if the number of $\{\mathbf{p}_0, \mathbf{p}_1, \cdots, \mathbf{p}_i\}$ is too small (e.g., $i < 5$), such as curve c_1 shown in Fig. 4, then we regard these points as noise and delete them. By tracing inward from another endpoint of c_0 in the same way, we finally split this long curve into two real curves, c_0 and c_2, see Fig. 4 (right)). At the same time, since an original curve may have three endpoints in a T-shaped curve, we first find a common point for these curves from the three endpoints, and then divide the original curve into three curves from the common point.

Outward extension. We extend each curve outwards from its two endpoints to complete the FCN. By taking the endpoint $\mathbf{p}_0$ as an example (see Fig. 5), we aim to find the optimal mesh vertex to which the curve could extend. To this end, we compute a candidate set of mesh vertices, C_v, which is the

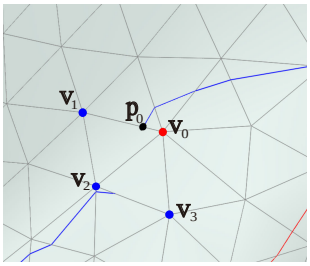

Figure 5: Illustration of computing the set of candidate vertices for the outward extension.

subset of the one-ring neighboring vertices of $\mathbf{v}_0$ ($\mathbf{v}_0$ is the closest mesh vertex of $\mathbf{p}_0$). We consider each neighbor $\mathbf{v}_k$ of $\mathbf{v}_0$, if the angle between the vector $\mathbf{v}_k - \mathbf{v}_0$ and the curve direction at $\mathbf{p}_0$ is less than 90°. Then we add $\mathbf{v}_k$ to the set C_v. In this way, each $\mathbf{v}_k$ in C_v does not deviate from the curve direction too much or turn back to the endpoint $\mathbf{p}_0$.

After building the candidate set, we define the extension cost value $F(\mathbf{v}_k)$ for each vertex $\mathbf{v}_k \subseteq \{C_v\}$ to measure the possibility of $\mathbf{v}_k$ being selected for the extension. Thus, we add the candidate vertex $\mathbf{v}_k$ with the minimal extension cost to the feature curve. We use both the curvature direction and curve-fitting error to calculate the extension cost function $F(\mathbf{v}_k)$ as follows:

$$F(\mathbf{v}_k) = E(\mathbf{v}_k) + \lambda G(t_{min}(\mathbf{v}_k), t_{min}(\mathbf{v}_0)) \quad (5)$$

where $E(\mathbf{v}_k)$ is the curve fitting error defined above, G is the function to calculate the minimal principal curvature directions angle between $\mathbf{v}_k$ and $\mathbf{v}_0$, and λ is the balance weight that normalizes E and G to the same scale.

To accelerate the extension speed, we collect the candidate vertices of the curves and use a global priority queue to maintain them (the priority of each candidate is set to their extension cost value). At each time, we pop the candidate $\mathbf{v}_k$ with the least cost from the queue and connect it to the corresponding curve endpoint if the fitting error of $\mathbf{v}_k$ is less than the threshold β (1.3 times the average curve fitting error that is computed before the outward extension). If $\mathbf{v}_k$ has been mapped to another curve, then two feature curves are connected by $\mathbf{v}_k$. In this case, the current curve will not extend in this direction anymore. The curve extension process is then performed by repeatedly popping candidates with the least cost. Once a candidate is popped, the priority queue is updated by updating the candidate vertex set and their cost values. Finally, we extract a complete FCN when the priority queue becomes empty.

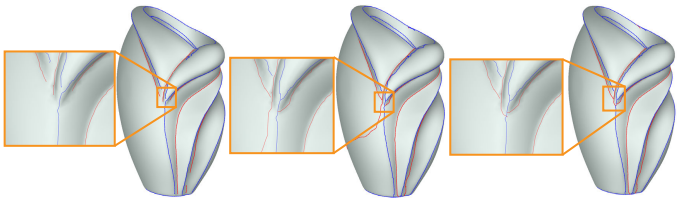

Figure 6: Illustration of our improved curve extension algorithm. Left: the initial feature curves; middle: the extension result of [27]; and right: our result.

A similar extension strategy has been used in [27]. However, they only use the curvature and principal direction. Thus this method is unsuitable for the smooth regions with unclear features. Fig. 6 shows the comparison of our improved curve extension result and those in [27].

Breaking and merging curves. This feature extension step generates many *joining points* that connect several original curves. If one joining point is not a node point and it connects two curves, then we must consider the following cases shown in Fig. 7 (a): (1) The joining point $\mathbf{p}_i$ is one endpoint of one curve c_2 but is the middle point of another curve c_1. We then break the curve c_1 into two curves; (2) The joining point $\mathbf{p}_0$ is the endpoint for both curves c_1 and c_3 and the corresponding curvature directions are similar. Finally, we combine these two curves into one feature curve (see Fig. 7 (b)).

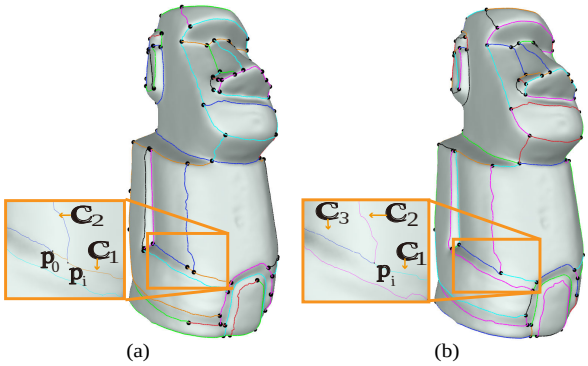

Figure 7: Curve merging: (a) initial curve extension result. (b) complete FCN after curves breaking and merging.

4.3. Patch Generation via Cycles Detection

A set of curve cycles are easily extracted from the complete FCN, then the input model can be segmented into different patches by the cycles.

Connectivity recovery. Due to the mesh irregularity, the connecting line of two adjacent feature points on the ridge and valley lines may not be located on a triangle. Therefore, when we map feature points $\{\mathbf{p}_i\}$ to the closest mesh vertices $\{\mathbf{v}_i\}$, the mapped vertex set and their edges between these vertices construct an undirected graph that may have more than one connected component, thus we cannot find the vertex set surrounded by the feature lines. In this step, we first examine and recover the connectivity of each mapped graph.

For each mapped graph, we use the inward tracing scheme again. First, we put the vertex $\mathbf{v}_0$ mapped by the endpoint $\mathbf{p}_0$ into a queue. Second, we iteratively pop out the first vertex in the queue and put its adjacent vertex $\mathbf{v}_k \subseteq \{\mathbf{v}_i\}$ which are not visited into the queue each time. Finally, if all vertices in $\{\mathbf{v}_i\}$ are visited when the queue is empty, the mapped graph is connected. Otherwise, we need to recover the connectivity of this graph. If the traversed $\mathbf{v}_j$ is not adjacent to the other remaining vertices $\{\mathbf{v}_r\}$, then we find the nearest vertex $\mathbf{v}_p \subseteq \{\mathbf{v}_r\}$ and add the vertices in the shortest path between $\mathbf{v}_j$ and $\mathbf{v}_p$ to $\{\mathbf{v}_i\}$. This step is repeated until the last mapped vertex $\mathbf{v}_n$.

Cycles detection and patch generation. Given that the complete FCN captures the boundary information of geometric patches in the input mesh, we generate patches by detecting the

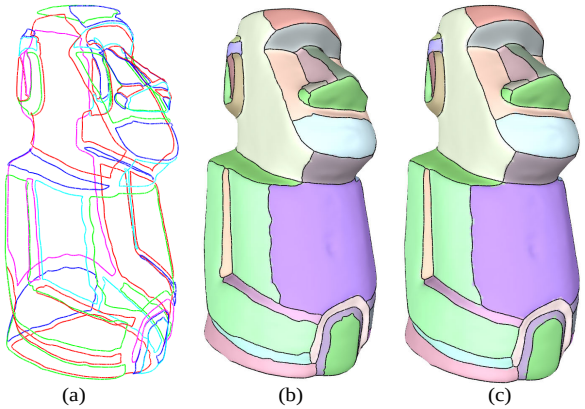

Figure 8: Patches generation: (a) the result of curve cycles. (b) primary patches with many zigzags on boundaries. (c) final patch layout after boundary smoothing.

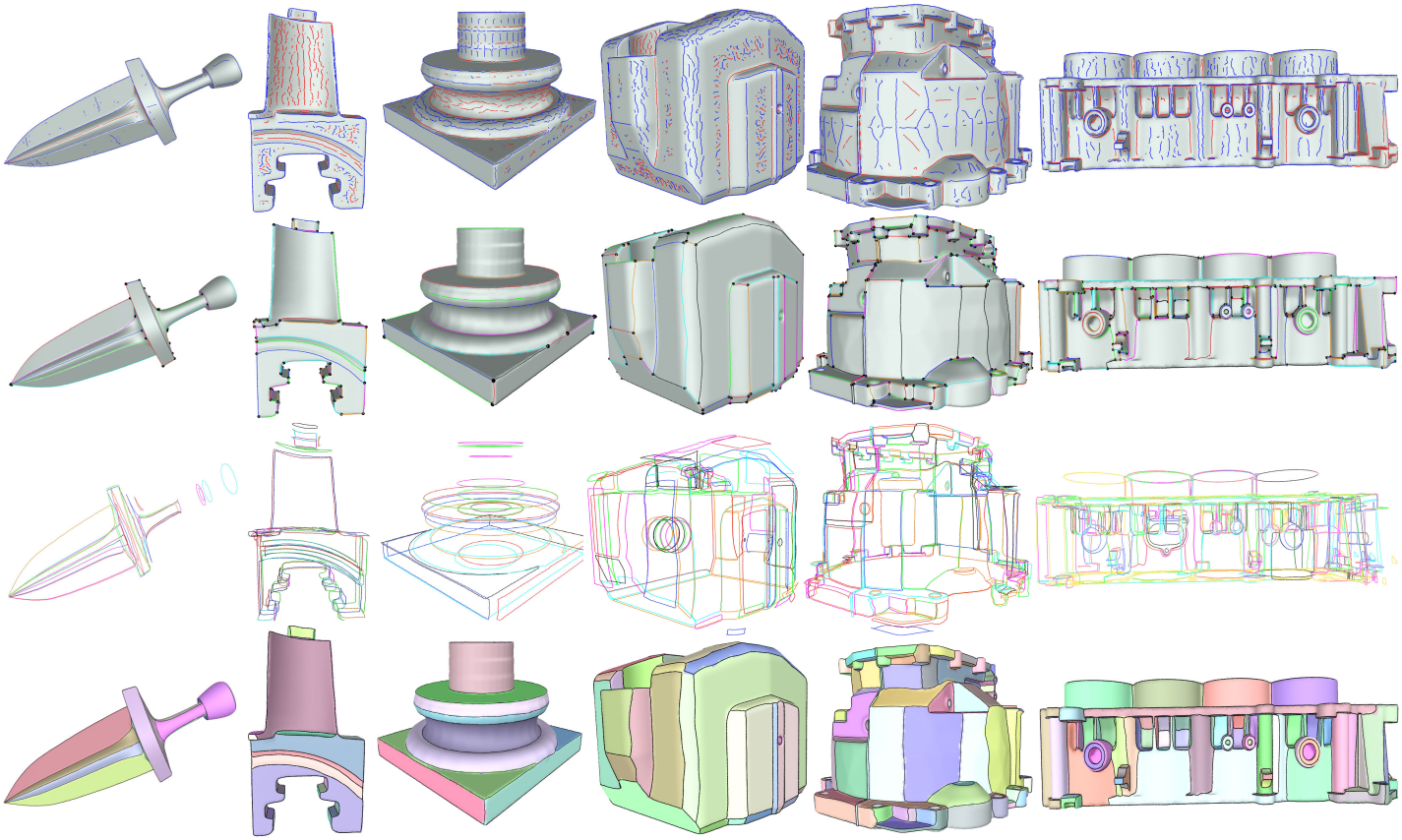

Figure 9: A gallery of examples generated by our framework. For each model, we show the input noisy feature lines, complete FCNs, exploded view of curve cycles, and patch layouts.

curve cycles from FCN. Here a curve cycle is defined as a continuous set of curves, which can be used as the boundary of a patch.

In order to find the curve cycles, we construct a graph, $\mathcal{G} = (V_g, U_g)$, whose nodes V_g are the endpoints of the feature curves and edges U_g are the feature curves in FCN. In general, a node's shortest loop L, which consists of a set of curves, is the boundary of a geometric patch in the input mesh, and a feature curve appears twice in all patch boundaries. Therefore, we iteratively find the shortest loop L in $\mathcal{G}$ to generate patches and delete the edge in $\mathcal{G}$ while the corresponding feature curve appears twice in the generated patches. Note that for sphere and cylinder surfaces, the curve on these surfaces may be a circle curve without node points. We directly regard such a circle curve as a cycle.

Since the vertex set V mapped by a curve cycle L_c is connected, the input mesh is divided into two parts and we can easily obtain the patch surrounded by L_c. For example, we add the vertex v_0, an adjacent vertex of V that has not been assigned to a patch, into a queue. Then we iteratively pop out the first vertex of the queue, place it into the vertex set S, and put its adjacent vertex v_k which does not belong to V into the queue. If v_k lies on the curve $c_i \notin L_c$, these vertices in S are located outside L_c and we clear the queue and S, then we add another unvisited vertex v_j adjacent to V to the queue and continue traversal. Finally, we obtain the new patch whose vertex set S is surrounded by L_c. Notably, there are patches with two or more boundary cycles, such as the cylinder patch. For such patches, we cannot directly find it by a curve cycle. Therefore, we first find each patch surrounded by a curve cycle, and then the undetected patches are the patches with two or more curve cycles. Here to find the undetected patch, we start from a curve cycle to traverse the adjacent mesh vertices. When encountering the adjacent curve cycles, the vertices surrounded by these curve cycles are the vertices of a patch with two or more curve cycles.

Since our filtering algorithm retains more salient curves and leads to finer segmentation in the transition area, it is necessary to calculate the relationship between these patches and merge similar adjacent patches. To do that, we assume that the patch with vertices less than 5% of the total mesh vertices may need to be merged. We first find adjacent patches based on the curve cycle in the original patch P_i and add patches whose number of vertices exceeds P_i to obtain an adjacent patch set $\{P_j, \cdots, P_k\}$. Then we traverse each adjacent patch in the set, and expand from the shared curves with P_i to find the same number of triangles with P_i, then form a new local region with P_i to fit a quadric surface and calculate the fitting error. When the smallest fitting error with adjacent patch P_k is smaller than 1.1 times the fitting error of P_i, we consider that P_i and P_k belong to the same patch and merge them. Otherwise, P_i is independent and will not be merged. Finally, we generate patches for the input mesh. The examples of cycles and patches are shown in Figs. 8 (a) and 8 (b).

Smoothing boundary. Since the patch boundaries are typically not smooth due to mesh discretization, we need a post-processing step to smooth the boundaries. We apply the iterative energy minimization method introduced in [1] to smooth the boundary curves as follows:

$$E(c) = \int_c \left(\frac{\langle \dot{c}, X \rangle}{\| \dot{c} \| \| X \|} \right)^2 ds \tag{6}$$

where X is the field of maximal principal curvature directions. By minimizing this energy, the boundary curve c is aligned to the field of principal curvature directions. An example of a smoothed patch layout is illustrated in Fig. 8 (c).

5. Experimental Results

In this section, we conduct a number of experiments on various input models to demonstrate the efficacy of our approach. First, we generate meaningful complete FCNs, cycles, and surface segmentation results from dense and discrete feature curves. Second, we evaluate the quality and validity of the filtering parameters of R and E. We also test our algorithm on noisy data to further demonstrate the robustness of our approach. Third, we compare our approach with state-of-the-art methods using qualitative and quantitative evaluations. Finally, we prove the usefulness of the extracted FCN by reconstructing the surfaces. We implement our algorithm in C++ by using a desktop computer equipped with an Intel i7-7700k processor with 4.2 GHz and 16 GB of RAM.

5.1. Evaluation

FCN extraction. We evaluate our method on a wide range of 3D models with different complexities, ranging from CAD components to building models. Fig. 9 shows the results of extracting complete FCNs from surface meshes. The input initial curves (ridges and valleys) are automatically computed by [12], and they are fragmented and discretely distributed on the surfaces. With the majority of parameters set to default, we only tune the R and E to achieve effective filtering results and the final patch layouts. With our approach, we can determine both sharp and smooth feature curves that capture the main geometric characteristics, such as mesh boundaries and fading features. We then obtain the complete FCNs by extending and connecting the preserved curves, as shown in the second row of Fig. 9. The last two rows show our detected cycles of closed curves and the resulting surface segmentation of the input mesh. Thus we decompose the surface into structural parts.

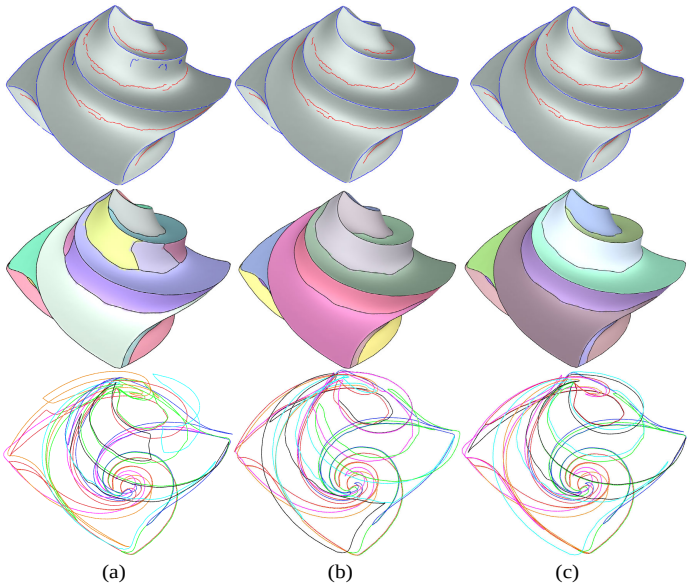

Figure 10: Feature curve filtering result (top) and the corresponding patch layouts (middle) and cycles (bottom). (a) $R = 0.004$ without E. (b) $R = 0.007$ without E. (c) $R = 0.004$ and $E = 1.6 * 10^{-7}$.

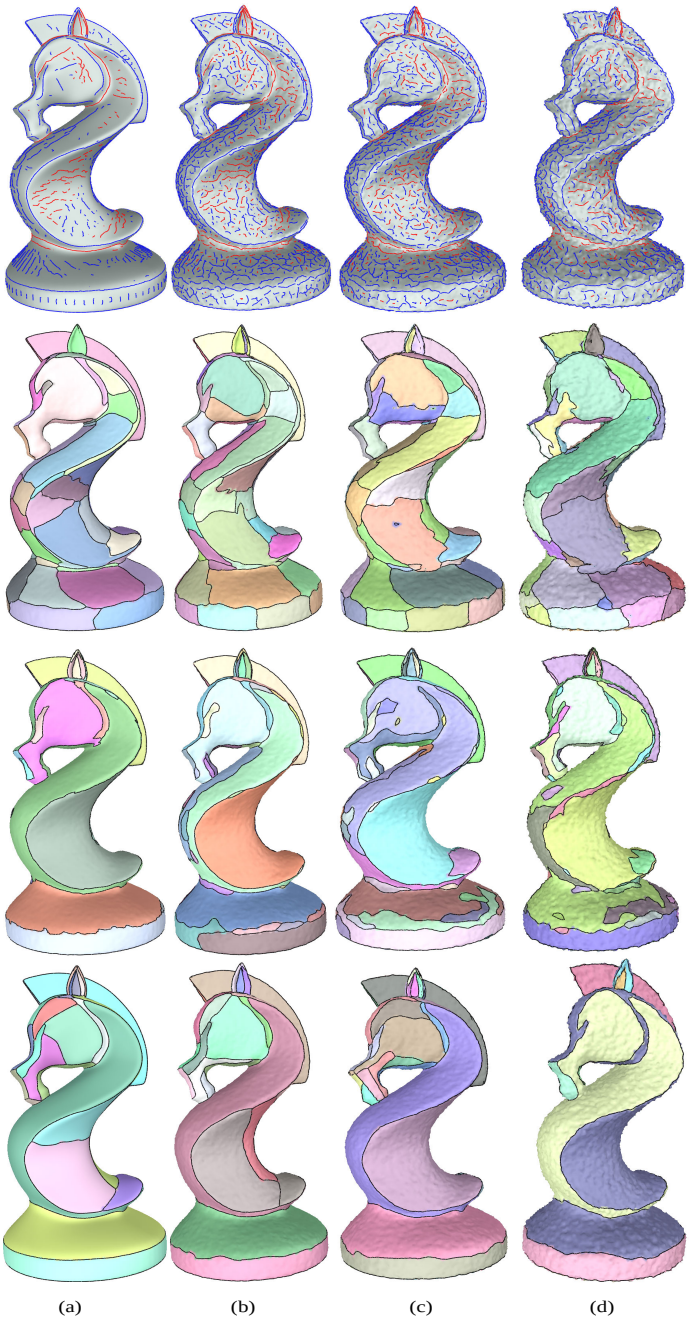

Figure 11: Comparison of models with different noise levels. From top to bottom: noisy models, results of VSA [9], results of FASCC [39], our results.

Feature curve filtering. The feature curve filtering algorithm described in Sec. 4.1 aims to obtain meaningful and salient feature curves from noisy and fragmented input. Parameters R and E in the filtering process have a direct effect on the result of filtering FCNs and surface segmentation. As shown in Fig. 10, we compare the influence of R and E with the result of curve filtering, patch layouts, and cycles. Although a low R parameter reserves some additional curves on quadric surfaces, a higher R parameter filters out several salient curves and results in unsatisfactory segmentation. By contrast, the parameter E removes the additional curves after filtering with a low R and achieves more satisfactory segmentation. For different input models, we tune both R and E parameters to obtain effective feature curve filtering results.

Robustness against noise. The robustness of our method is

demonstrated in Fig. 11 by comparing the difference in patch layouts between several synthetic noisy models with the original model. We also compare these to VSA [9] and FASCC [39]. We select a clean model and generate the patch layout using these algorithms (see Fig. 11 (a)). Then we randomly add Gaussian noise to mesh vertices with a standard deviation of 0.2%, 0.3%, and 0.6% (see Fig. 11 (b), 11 (c), 11 (d)) of the length of the bounding box diagonal. The initial curves computed with the noisy data are more crooked and discontinuous compared with those computed with the clean model. We successfully generate similar patch layouts for the noisy models by adjusting the parameters as shown in Table 1. Although the patch boundaries are more jagged when the noise intensity increases, we can still produce better results than VSA [9] and FASCC [39].

$\mid D \mid$	0	0.2	0.3	0.6
R	0.0001	0.008	0.01	0.016
E	$3 * 10^{-8}$	$5.8 * 10^{-7}$	$1.15 * 10^{-6}$	$4.1 * 10^{-6}$

Table 1: The parameter configurations for our method on noisy models. $\mid D \mid$ is the standard deviation of the noise, $\mid R \mid$ is the curve curvature filtering parameter, $\mid E \mid$ is the surface fitting error filtering parameter.

5.2. Comparison

We compare our method with previous approaches using different mesh segmentation strategies, including VSA [9], QSE [10], FGL [1], Papercafts [40], FASCC [39] and DFN [27]. Fig. 12 and Fig. 13 respectively illustrate the comparison results on four examples with ground-truth segmentation and three other complex models without ground-truth. We manually segment the model in Fig. 12 according to their geometric patches ("Toy" and "Sword") and the ground-truth segmentation results ("Lamp" and "Vase") in a 3D mesh segmentation benchmark [41]. We tune the parameters of all compared methods to obtain fair and satisfactory results. The numerical statistics on the patch layouts are presented in Table 2 and Table 3.

First, both VSA [9] and QSE [10] use primitive-fitting and cluster facets in the best-fitting regions to reduce the fitting error. Their fitting errors are generally smaller than those of other methods (Table 2). VSA [9] only utilizes planes for fitting and often create additional boundaries in large curvature variation regions such as sphere and cylinder (Fig. 12 (b)). QSE [10] applies quadric surface fitting and improves the quadric surface segmentation results, but their segmentation results are unstable and may miss some essential boundaries (see Fig. 12 (c)). FGL [1] adopts the region-growing algorithm to obtain the patch layout and generate a set of curves, while Papercafts [40] considers the pruned feature lines as boundaries and merges the small patches. Figs. 13 (c) and 13 (d) demonstrate that FGL [1] and Papercafts [40], respectively, miss many important feature curves, obtain unsatisfactory patch layouts, and produce additional patches due to curvature variation in the quadric surface. Second, FASCC [39] merges a watershed over-segmentation and provides an interactive interface to improve the patch layout. However, the result of mesh segmentation is affected by the quality of input FCN. As shown in Fig. 13 (e), their method fails to bridge feature curves and further separates in the blending regions.

Finally, since our method is partially motivated by DFN [27], which includes feature filtering and curve extension, we compare our curve network results to those of DFN [27] (see Fig. 12 (g) and Fig. 13 (f)). Although both our method and DFN [27] directly extend and connect pruned feature curves, DFN [27] typically removes more curves by only using the curve length and curvature without cycle information in the network. By contrast, we preserve additional salient curves to produce a satisfactory curve network by combining the curvature and surface fitting error. DFN [27] uses VSA [9] to further generate the patch layout and solve this problem. However, VSA [9] is unsuitable for quadric surfaces. We detect cycles in the network, fit the quadric surface again for each patch, and compute the hybrid distance between the fitting and input surfaces by equation (3) to measure the patch quality. Thus, we can generate additional reasonable patch layouts, and our patches have smaller fitting errors compared with DFN (see Table 2 and Table 3).

Comparison to our conference work [13]. As shown in Fig. 14 (a), our new approach can find the node points in the long curve

Models	$\mid f \mid$	Methods	$\mid C \mid$	$\mid P \mid$	$\mid E \mid$	T(s)
Toy	59996	Ground-truth	47	26	0.0802	0
		VSA [9]	136	50	0.1125	1.776
		QSE [10]	116	45	0.0242	5.294
		FGL [1]	30	14	0.1646	0.169
		Papercafts [40]	51	**26**	0.1014	0.315
		FASCC [39]	29	16	**0.0872**	3.579
		DFN [27]	30	17	0.1211	**0.128**
		Our	**40**	23	0.0671	0.319
Sword	10042	Ground-truth	56	23	0.0057	0
		VSA [9]	75	28	**0.0055**	0.656
		QSE [10]	35	18	0.0166	1.516
		FGL [1]	33	17	0.0137	0.582
		Papercafts [40]	38	16	0.0147	0.486
		FASCC [39]	35	17	0.0123	1.417
		DFN [27]	42	18	0.0190	**0.079**
		Our	**50**	**21**	0.0064	0.316
Lamp	27100	Ground-truth	60	20	0.0329	0
		VSA [9]	102	32	**0.0374**	0.895
		QSE [10]	91	30	0.0189	8.783
		FGL [1]	27	9	0.1090	0.184
		Papercafts [40]	48	**20**	0.1505	0.31
		FASCC [39]	54	16	0.1549	1.371
		DFN [27]	84	18	0.0439	**0.117**
		Our	**64**	23	0.0502	0.326
Vase	40000	Ground-truth	79	32	0.0136	0
		VSA [9]	93	**32**	0.0224	1.264
		QSE [10]	**68**	26	**0.0119**	6.393
		FGL [1]	3	4	0.1106	0.198
		Papercafts [40]	44	21	0.0401	0.353
		FASCC [39]	51	22	0.0283	1.607
		DFN [27]	51	18	0.0236	**0.145**
		Our	63	25	0.0197	0.439

Table 2: Evaluation and comparison of models with ground-truth. $\mid f \mid$ is the number of triangles in each model, $\mid C \mid$ is the number of preserved curves in the final network, $\mid P \mid$ is the number of generated patches, and $\mid E \mid$ is the hybrid distance between the fitting surfaces and the input mesh, and T is the running time, respectively. The best result of each measurement closest to the ground-truth is marked in **bold** font.

Models	$\mid f \mid$	Methods	$\mid C \mid$	$\mid P \mid$	$\mid E \mid$	T(s)
Anouk	99790	VSA [9]	178	64	0.0614	5.445
		QSE [10]	82	39	0.0585	20.745
		FGL [1]	367	134	0.0358	2.083
		Papercafts [40]	460	164	0.0317	2.829
		FASCC [39]	435	155	0.0344	123.251
		DFN [27]	509	186	0.0218	**1.39**
		Our	**591**	**206**	**0.0203**	6.45
Bell	100464	VSA [9]	**2892**	**988**	**0.1597**	10.982
		QSE [10]	273	101	0.9660	36.983
		FGL [1]	1603	573	0.2638	9.239
		Papercafts [40]	2284	793	0.2559	10.194
		FASCC [39]	1871	653	0.2535	20.623
		DFN [27]	1059	368	0.4459	**2.21**
		Our	1918	660	0.3116	7.781
House	195728	VSA [9]	**1448**	**491**	**0.0650**	10.131
		QSE [10]	285	106	0.1836	19.013
		FGL [1]	205	91	0.2295	1.391
		Papercafts [40]	804	306	0.0953	0.758
		FASCC [39]	699	263	0.1547	34.918
		DFN [27]	443	158	0.0766	**1.906**
		Our	1067	377	0.0674	5.219

Table 3: Evaluation and comparison of models without ground-truth. The largest $\mid C \mid$ and $\mid P \mid$ and smallest $\mid E \mid$ and T of each measurement are marked in **bold** font.

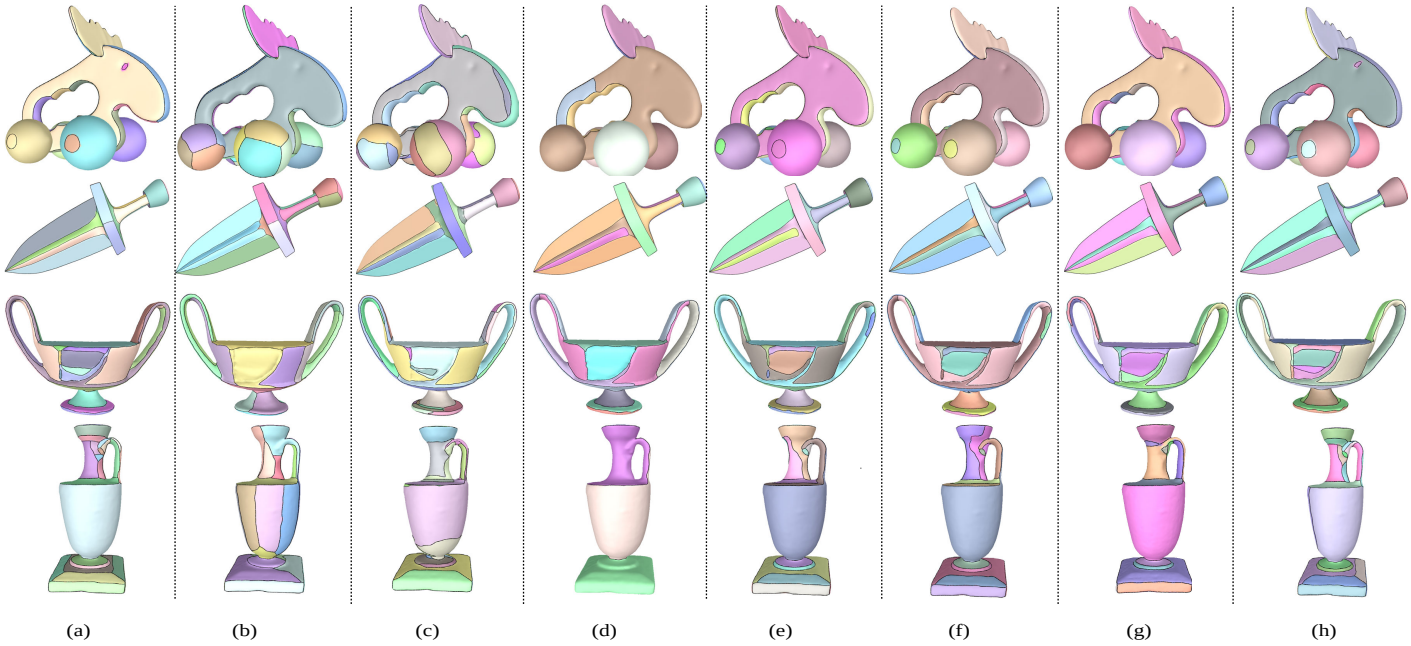

Figure 12: Comparison with previous methods on models of "Toy", "Sword", "Lamp" and "Vase" with ground-truth segmentation. From left to right: ground-truth segmentation (a), VSA [9] (b), QSE [10] (c), FGL [1] (d), Papercafts [40] (e), FASCC [39] (f), DFN [27] (g), and our method (h), respectively.

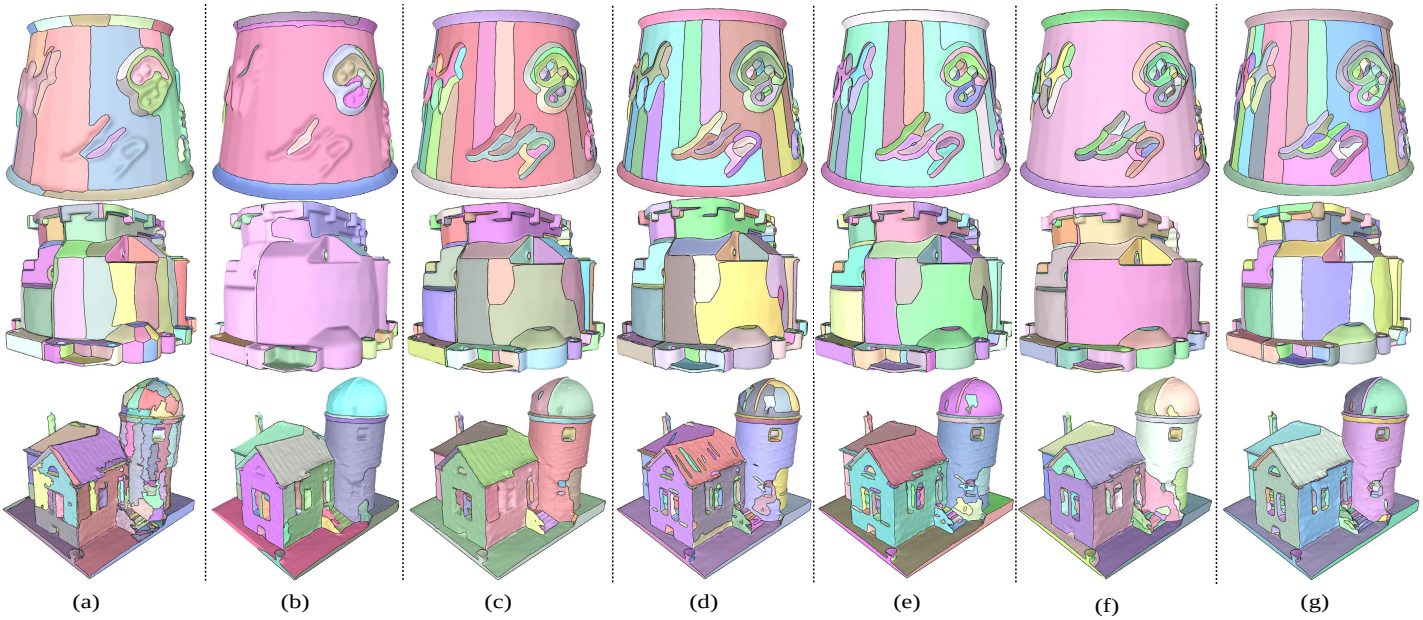

Figure 13: Comparison with previous methods on models of "Anouk", "Bell" and "House" without ground-truth segmentation. From left to right: results of VSA [9] (a), QSE [10] (b), FGL [1] (c), Papercafts [40] (d), FASCC [39] (e), DFN [27] (f), and our method (g), respectively.

to divide it into short curves and refit them for the curve extensions, so we can extend the curve along their direction and get a better patch layout. Meanwhile, in [13], some ridges and valley lines exist in the transition regions, which cannot be effectively filtered, resulting in some undesired small patches. In this paper, after expanding these curves, we can avoid them through our merging operation, see in Fig. 14 (b) and (c).

5.3. Application: Surface Reconstruction

To further verify the validity of FCN extracted by our method, we generate surfaces from the extracted cycles by using the algorithm introduced in [34]. Fig. 15 shows two results of the reconstructed surfaces. Fig. 15 (a) demonstrates several transition regions where two quadric surfaces intersect in the input model. We can extract the mesh boundary and form the curve loop for each quadric surface. We then directly take the curve loop in Fig. 15 (b) as the input of [34]. Finally, the reconstructed surfaces are illustrated in Fig. 15 (c), and the blending areas are removed from the model. Note the building in Fig. 15(c) has additional creases that are not at all visible in the input model. The reason is that we only take the obtained curve cycles as the input to [34] without other constraints, and [34] may generate additional creases on the reconstructed surface.

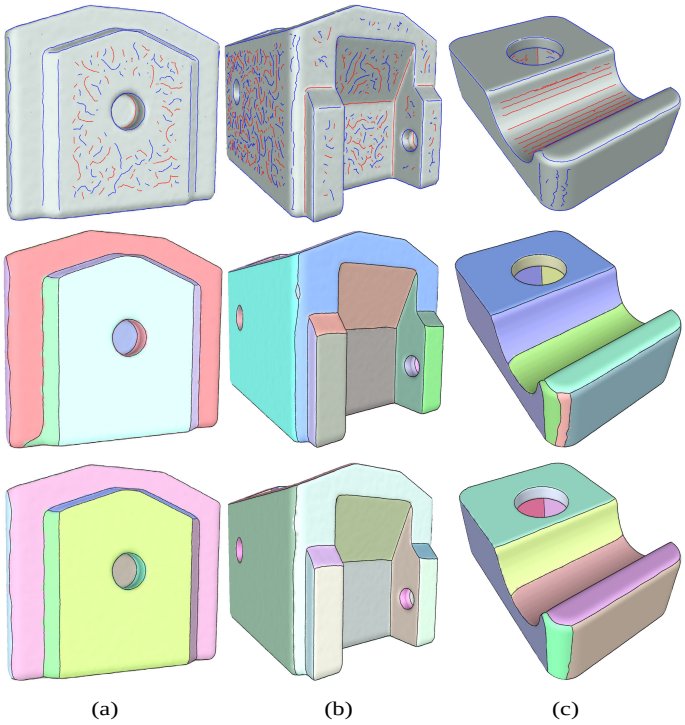

Figure 14: Compared with our conference work [13]. From top to bottom: input models with feature curves, patch layouts of [13], and our results.

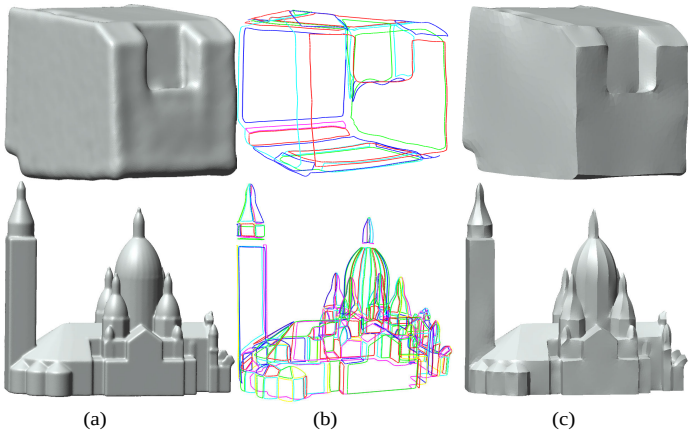

Figure 15: Surface reconstruction: (a) input models with smoothing areas; (b) extracted cycles; (c) reconstructed surface result using the algorithm of [34].

5.4. Limitations

We successfully extract complete FCNs from various 3D models. Given that our feature filtering and extension operations only function on individual curves, we disregard the distance or relationships between different curves. Thus, our results miss the global information of shapes, such as symmetry, parallelism, and coplanarity. In our method, the validity of FCN from meshes with low sampling resolution is uncertain because of its poor sensitivity to mesh scales or triangle shapes. Fig. 16 (a) shows that the input resolution is too low to obtain acceptable ridge and valley lines especially near the boundary features and the patch layout is also unsatisfactory. We can get a satisfying result after subdividing the mesh (Fig. 16 (b)). Furthermore, our method may produce insignificant results for organic shapes because of the lacking semantic information (see Fig. 16 (c)).

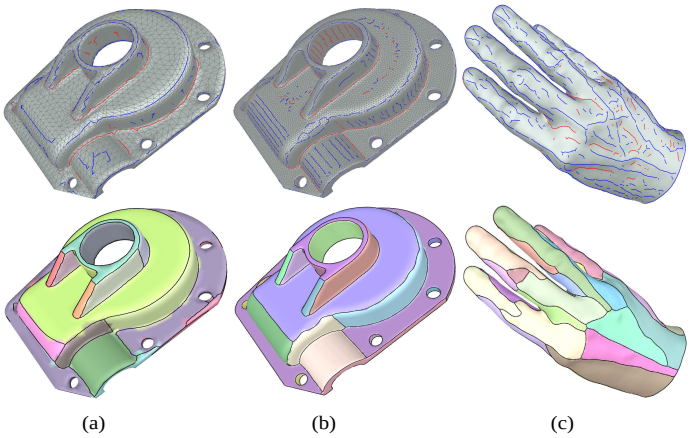

Figure 16: (a) A low-resolution mesh and the corresponding patch layout; (b) a high-resolution mesh and the corresponding patch layout; (c) a hand model and the patch layout. Here (a) and (c) show the unsatisfactory results of our method.

6. Conclusion and Future Work

In this paper, we propose a new method to address the problem of extracting cycle-aware FCNs from 3D models. We successfully solve this problem by utilizing operations of feature curve filtering, feature curve extension, and cycle detection. The proposed algorithm, which is robust to noisy data, can detect both strong and weak features and connect them to valid and complete FCNs. We demonstrate our approach on a variety of examples, including simple mechanical parts to complex 3D building models.

In future investigations, we can integrate global shape constraints into our framework. Furthermore, we can extend our approach to other shape analysis applications, such as finding repeated feature instances in the curve network as well as fitting and editing a subdivision or spline surface by manipulating certain feature curves.

Acknowledgements

This work is partially funded by National Key R&D Program (2018YFB2100602), NSFC (61802406, 61772523), Beijing science and technology project (Z181100003818019), the Youth Innovation Promotion Association of CAS (Y201935), Beijing Natural Science Foundation (L182059), Open Project Program of State Key Laboratory of Virtual Reality Technology and Systems Beihang University (VRLAB2019B02), CCF–Tencent Open Research Fund (RAGR20190105), and the Key Research Program of Frontier Sciences CAS (QYZDY-SSW-SYS004).

[1] M. Nieser, C. Schulz, K. Polthier, Patch layout from feature graphs, Computer-Aided Design 42 (3) (2010) 213–220.

[2] R. Mehra, Q. Zhou, J. Long, A. Sheffer, A. Gooch, N. J. Mitra, Abstraction of man-made shapes, in: ACM Transactions on Graphics (Proc. SIGGRAPH Asia), 2009, pp. 1–10.

[3] F. De Goes, S. Goldenstein, M. Desbrun, L. Velho, Exoskeleton: Curve network abstraction for 3d shapes, Computers & Graphics 35 (1) (2011) 112–121.

[4] R. Ling, J. Huang, B. Jüttler, F. Sun, H. Bao, W. Wang, Spectral quadrangulation with feature curve alignment and element size control, ACM Transactions on Graphics 34 (1) (2014) 1–11.

[5] D. Khan, D.-M. Yan, F. Ding, Y. Zhuang, X. Zhang, Surface remeshing with robust user-guided segmentation, Computational Visual Media 4 (2) (2018) 113–122.

[6] H. Pan, Y. Liu, A. Sheffer, N. Vining, C.-J. Li, W. Wang, Flow aligned surfacing of curve networks, ACM Transactions on Graphics 34 (4) (2015) 1–10.

[7] G. Gori, A. Sheffer, N. Vining, E. Rosales, N. Carr, T. Ju, Flowrep: Descriptive curve networks for free-form design shapes, ACM Transactions on Graphics 36 (4) (2017) 1–14.

[8] A. Gehre, M. Bronstein, L. Kobbelt, J. Solomon, Interactive curve constrained functional maps, in: Computer Graphics Forum, Vol. 37, Wiley Online Library, 2018, pp. 1–12.

[9] D. Cohen-Steiner, P. Alliez, M. Desbrun, Variational shape approximation, in: ACM Transactions on Graphics (Proc. SIGGRAPH), 2004, pp. 905–914.

[10] D.-M. Yan, W. Wang, Y. Liu, Z. Yang, Variational mesh segmentation via quadric surface fitting, Computer-Aided Design 44 (11) (2012) 1072–1082.

[11] Y. Ohtake, A. Belyaev, H.-P. Seidel, Ridge-valley lines on meshes via implicit surface fitting, in: ACM Transactions on Graphics (Proc. SIGGRAPH), 2004, pp. 609–612.

[12] S. Yoshizawa, A. Belyaev, H.-P. Seidel, Fast and robust detection of crest lines on meshes, in: Proceedings of the 2005 ACM symposium on Solid and physical modeling, 2005, pp. 227–232.

[13] Z. Lu, J. Guo, J. Xiao, Y. Wang, X. Zhang, D.-M. Yan, Feature curve network extraction via quadric surface fitting, in: Pacific Graphics Short Papers, 2019.

[14] S.-K. Kim, C.-H. Kim, Finding ridges and valleys in a discrete surface using a modified mls approximation, Computer-Aided Design 38 (2) (2006) 173–180.

[15] Y.-K. Lai, Q.-Y. Zhou, S.-M. Hu, J. Wallner, H. Pottmann, Robust feature classification and editing, IEEE Transactions on Visualization and Computer Graphics 13 (1) (2006) 34–45.

[16] M. Nomura, N. Hamada, Feature edge extraction from 3d triangular meshes using a thinning algorithm, in: Vision Geometry X, Vol. 4476, International Society for Optics and Photonics, 2001, pp. 34–41.

[17] T. Weinkauf, D. Günther, Separatrix persistence: Extraction of salient edges on surfaces using topological methods, in: Computer Graphics Forum, Vol. 28, Wiley Online Library, 2009, pp. 1519–1528.

[18] H. S. Kim, H. K. Choi, K. H. Lee, Feature detection of triangular meshes based on tensor voting theory, Computer-Aided Design 41 (1) (2009) 47–58.

[19] H. Liu, N. Dai, B. Zhong, T. Li, J. Wang, Extract feature curves on noisy triangular meshes, Graphical Models 93 (2017) 1–13.

[20] M. Sunkel, S. Jansen, M. Wand, E. Eisemann, H.-P. Seidel, Learning line features in 3d geometry, in: Computer Graphics Forum, Vol. 30, Wiley Online Library, 2011, pp. 267–276.

[21] A. Gehre, I. Lim, L. Kobbelt, Feature curve co-completion in noisy data, in: Computer Graphics Forum, Vol. 37, Wiley Online Library, 2018, pp. 1–12.

[22] S. Yamakawa, K. Shimada, Feature edge extraction via angle-based edge collapsing and recovery, Journal of Computing and Information Science in Engineering 18 (2) (2018) 021001–021001–18.

[23] A. Gehre, I. Lim, L. Kobbelt, Adapting feature curve networks to a prescribed scale, in: Computer Graphics Forum, Vol. 35, Wiley Online Library, 2016, pp. 319–330.

[24] M.-L. Torrente, S. Biasotti, B. Falcidieno, Recognition of feature curves on 3d shapes using an algebraic approach to hough transforms, Pattern Recognition 73 (2018) 111–130.

[25] G. Sharma, D. Liu, E. Kalogerakis, S. Maji, S. Chaudhuri, R. Měch, Parsenet: A parametric surface fitting network for 3d point clouds, arXiv preprint arXiv:2003.12181.

[26] S. Hu, J. Cai, Y. Lai, Semantic labeling and instance segmentation of 3d point clouds using patch context analysis and multiscale processing, IEEE Trans. Vis. Comput. Graph. 26 (7) (2020) 2485–2498.

[27] Y. Cao, D.-M. Yan, P. Wonka, Patch layout generation by detecting feature networks, Computers & Graphics 46 (2015) 275–282.

[28] A. Nealen, T. Igarashi, O. Sorkine, M. Alexa, Fibermesh: designing freeform surfaces with 3d curves, in: ACM Transactions on Graphics (Proc. SIGGRAPH), 2007, pp. 41–es.

[29] L. Liu, C. Bajaj, J. O. Deasy, D. A. Low, T. Ju, Surface reconstruction from non-parallel curve networks, in: Computer Graphics Forum, Vol. 27, Wiley Online Library, 2008, pp. 155–163.

[30] G. Orbay, 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, 2011, pp. 71–78.

[31] F. Abbasinejad, P. Joshi, N. Amenta, Surface patches from unorganized space curves, in: Computer Graphics Forum, Vol. 30, Wiley Online Library, 2011, pp. 1379–1387.

[32] M. Bessmeltsev, C. Wang, A. Sheffer, K. Singh, Design-driven quadrangulation of closed 3d curves, ACM Transactions on Graphics 31 (6) (2012) 1–11.

[33] M. Zou, T. Ju, N. Carr, An algorithm for triangulating multiple 3d polygons, in: Computer Graphics Forum, Vol. 32, Wiley Online Library, 2013, pp. 157–166.

[34] Y. Zhuang, M. Zou, N. Carr, T. Ju, A general and efficient method for finding cycles in 3d curve networks, ACM Transactions on Graphics 32 (6) (2013) 1–10.

[35] B. Xu, W. Chang, A. Sheffer, A. Bousseau, J. McCrae, K. Singh, True2form: 3d curve networks from 2d sketches via selective regularization, ACM Transactions on Graphics 33 (4) (2014) 131:1–131:13.

[36] T. Stanko, S. Hahmann, G.-P. Bonneau, N. Saguin-Sprynski, Surfacing curve networks with normal control, Computers & Graphics 60 (2016) 1–8.

[37] Y. Wang, D.-M. Yan, X. Liu, C. Tang, J. Guo, X. Zhang, P. Wonka, Isotropic surface remeshing without large and small angles, IEEE Transactions on Visualization and Computer Graphics 25 (7) (2018) 2430–2442.

[38] D.-M. Yan, Y. Liu, W. Wang, Quadric surface extraction by variational shape approximation, in: International conference on geometric modeling and processing, Springer, 2006, pp. 73–86.

[39] Y. Zhuang, H. Dou, N. Carr, T. Ju, Feature-aligned segmentation using correlation clustering, Computational Visual Media 3 (2) (2017) 147–160.

[40] J. Mitani, H. Suzuki, Making papercraft toys from meshes using strip-based approximate unfolding, ACM Transactions on Graphics 23 (3) (2004) 259–263.

[41] X. Chen, A. Golovinskiy, T. Funkhouser, A benchmark for 3d mesh segmentation, ACM Transactions on Graphics 28 (3) (2009) 1–12.