Computational Visual Media
https://doi.org/10.1007/s41095-020-0192-6 Vol. 6, No. 4, December 2020, 431–443
Research Article

# Simple primitive recognition via hierarchical face clustering

**Xiaolong Yang**[1,2], **Xiaohong Jia**[1,2] (✉)



**Abstract** We present a simple yet efficient algorithm for recognizing simple quadric primitives (plane, sphere, cylinder, cone) from triangular meshes. Our approach is an improved version of a previous hierarchical clustering algorithm, which performs pairwise clustering of triangle patches from bottom to top. The key contributions of our approach include a strategy for priority and fidelity consideration of the detected primitives, and a scheme for boundary smoothness between adjacent clusters. Experimental results demonstrate that the proposed method produces qualitatively and quantitatively better results than representative state-of-the-art methods on a wide range of test data.

**Keywords** quadric primitive extraction; mesh; hierarchical clustering

## 1 Introduction

Triangular meshes are one of the most popular representations of 3D shapes in computer graphics and 3D vision. In recent years, with the rapid development of 3D data acquisition techniques, it has become much easier to obtain precise geometric data. However, the obtained raw data has a large number of elements and lacks high-level information, so is difficult to use directly in downstream applications. For example, in industrial design and manufacturing applications, the original computer aided design (CAD) model might be unavailable or unsuitable for processing by current software for various reasons, leaving only a tessellated triangular mesh available. Hence the detection and recognition of high-level primitives in complex 3D data are required by many applications, such as reverse engineering [1, 2], 3D printing [3, 4], and other digital technologies.

Many algorithms exist for mesh segmentation and primitive extraction for various purposes, for example, convex decomposition [5], parametric shape extraction (planes [6], spheres and cylinders [7], developable patches [8], ellipsoids [9, 10], or general quadrics [11]), and semantic object recognition for indoor/outdoor scenes [12]. From the algorithmic point of view, segmentation algorithms can be classified into top–down decomposition (automatic or interactive) [13, 14], hierarchical bottom–up clustering, region growing [15], variational approximations [6, 7, 16], and more recent techniques based on deep learning [17], etc. However, the detection of simple primitives is an ill-posed problem. None of these methods is universally applicable in all situations.

Amongst these methods, hierarchical clustering is the simplest and most efficient algorithm for primitive detection, especially for CAD models [18, 19]. It performs pairwise clustering of adjacent clusters from bottom to top, according to designed merging criteria. In this paper, we propose an improved hierarchical clustering algorithm for extracting simple planar and quadric primitives from triangular meshes. Three major improvements are made upon previous algorithms, and an example result is shown in Fig. 1. Firstly, the primitive priority is taken into account during clustering, which is plane > cylinder > sphere > cone, where > denotes "is preferred to". Secondly, the smoothness of the boundary of each cluster is characterized by an additional regularization term. Third, fidelity is considered to avoid unnecessary merging even under acceptable fitting error, as is shown in Fig. 3.

We have conducted extensive comparisons with existing representative approaches [11, 18–20], and

1 Academy of Mathematics and Systems Science, Chinese Academy of Sciences, Beijing 100190, China.
2 University of Chinese Academy of Sciences, Beijing 100049, China. E-mail: X. Yang, yangxiaolong17@mails.ucas.ac.cn; X. Jia, xhjia@amss.ac.cn (✉).
Manuscript received: 2020-04-29; accepted: 2020-08-05

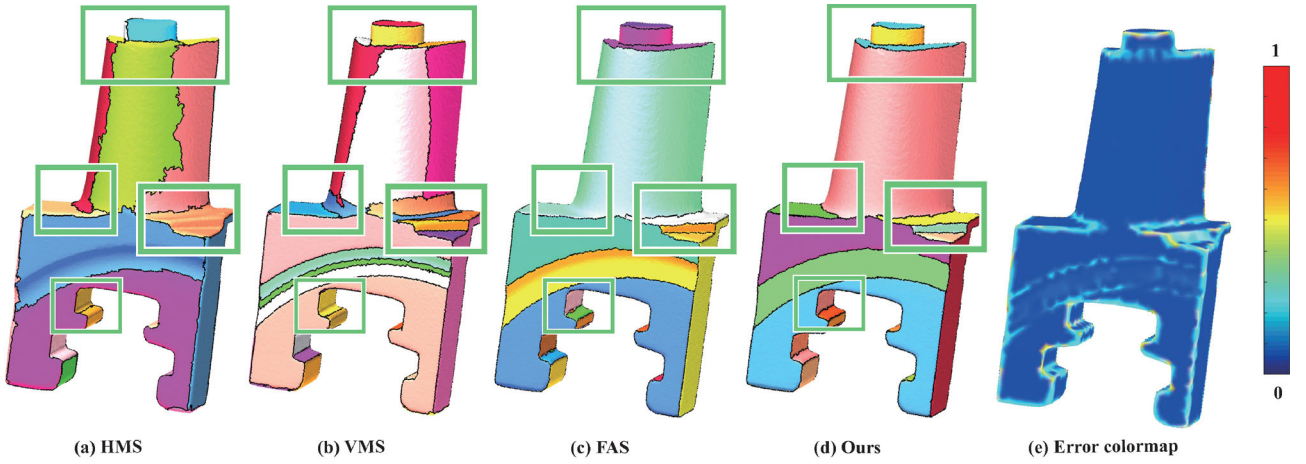



**Fig. 1** Comparisons with representative approaches: (a) *Hierarchical Mesh Segmentation* (HMS) [19], (b) *Variational Mesh Segmentation* (VMS) [11], (c) *Feature-Aligned Segmentation* (FAS) [20], and (d) our result. Colors are randomly assigned to segmented patches. (e) *Error colormap*. The color represents the distance from the point in the model to the fitted surface. Blue (value=0) indicates that the point is exactly located on the fitted surface, while red (value=1) indicates the largest distance between the fitted surface and the data point. Our result has better boundary smoothness and more meaningful extracted primitives, as highlighted.

demonstrated the advantages of our approach using a wide range of test data. The main contributions of this paper include the following:

- a simple yet efficient algorithm for primitive recognition using hierarchical face clustering;
- simultaneous consideration of both shape priority and fidelity during clustering;
- a new boundary smoothing formulation for improved boundary regularization.

## 2 Related work

Primitive extraction can be regarded as a mesh segmentation problem, which has been comprehensively studied in recent decades [21, 22]. Various criteria have been proposed for different tasks. For example, approximation fidelity and patch smoothness are the major concerns in reverse engineering and shape approximation, 3D printing imposes the printability and size constraints for each part, shape analysis and parsing segment shapes along concave lines, while scene understanding performs semantic labeling for high-level primitives. In the following, we focus on simple primitive extraction algorithms, and briefly consider major mesh segmentation approaches for different applications.

### 2.1 Shape approximation

*Region growing* is commonly used in many reverse engineering systems to extract smooth regions for scanned CAD meshes [1, 15, 23, 24]. It is also the key component of other clustering-based algorithms. Starting from a seed point, it repeatedly groups neighboring unlabeled elements with similarity of local properties, e.g., normals or curvatures. Although region growing is very efficient, it is difficult to predict the number of patches and always needs a certain amount of post-processing.

*Variational approximation* performs primitive fitting and region growing repeatedly to minimize an energy function with respect to different shape primitives. This optimization process is also known as Lloyd iteration [25]. Various metrics have been designed for extracting different primitives, e.g., planes [6], spheres and cylinders [7], ellipsoidal patches [9] and volumes [10], developable surfaces [8], and general quadric surfaces [11, 16]. This type of approach works well for clean shapes with clear structures, but is often time consuming due to its need for optimization.

*RANSAC* can be used to extract parametric primitives from raw data directly [26]; it originated in the computer vision community. However, the extracted primitives cannot be guaranteed to form connected regions and misclassification can occur. Hence, it is restricted to use as a pre-processing step for other algorithms [27].

*Hierarchical clustering* performs pairwise clustering of adjacent elements from bottom to top. A priority queue is used to determine which pair should be

clustered. A cost function is designed to measure the cost of merging a pair of clusters. For example, Garland et al. [18] measured planarity and the regularity of clusters. Attene et al. [19] extended the cost function to simple quadric primitives (sphere and cylinders) and considered the convexity of volumetric components. These approaches work very well for shapes consisting of simple primitives, but always lead to non-smooth segmentation boundaries for scanned data.

### 2.2 Shape decomposition

Part salience and minima rules [28] are widely used in many shape analysis tasks; the concavity of shapes is the main cue used to guide the segmentation. This type of approach is also known as part-based segmentation [29]. Katz and Tal [13] proposed a novel hierarchical mesh decomposition algorithm based on fuzzy clustering and graphcut. Lai et al. [30] solved the shape decomposition problem using a random walk formulation. Lafarge et al. [31] used a *Markov random field* (MRF) to label the vertices of the mesh. Lien et al. [5] explored an alternative partitioning strategy that decomposes a given model into approximately convex pieces for applications such as collision detection. Chen et al. [32] described a benchmark for evaluation of 3D mesh segmentation algorithms, which revealed the underlying theoretical concepts and classified segmentation algorithms. Due to the difficulties of automatic segmentation, some approaches allow user interaction to assist the segmentation process [14, 33, 34]. Instead of segmentation in Euclidean space, some approaches first transform the input mesh into the frequency domain and apply spectral clustering of the shape [35, 36].

Instead of using minima rules, feature lines can also be used for segmentation. Such algorithms first extract ridges and valleys [37] from input meshes, and then remove small features by filtering and extend the major ones to form closed feature loops [20, 38]. The enclosed regions are extracted as the final segmentation.

Mesh decomposition is also required in 3D printing applications: due to size and printability constraints, an input shape has to be decomposed into small pieces [3] or sub-components that can readily be printed [4, 39] or packed [40].

More recently, machine learning techniques were introduced in the geometry processing community [41, 42]. Guo et al. [43] presented a novel approach for 3D mesh labeling using deep convolutional neural networks (CNNs); it proved to be more robust. On this basis, Kalogerakis et al. [41] combined image-based fully convolutional networks (FCNs) and surface-based conditional random fields (CRFs) to yield coherent segmentations of 3D shapes. Simultaneously, Charles et al. [44] designed a novel type of neural network directly applicable to point clouds. It is invariant under rigid transformations of point positions in the input, and showed strong performance. Xu et al. [45] proposed a 3D shape representation learning approach, a directionally convolutional network (DCN), to extend convolution operations from images to the surface meshes of 3D shapes. However, such approaches still cannot provide an exact segmentation for surface approximation purposes.

### 2.3 Semantic segmentation

Apart from the above-mentioned low-level segmentation tasks, many techniques have been proposed recently for high-level primitive segmentation for indoor or outdoor scenes. For example, Kim et al. [12] exploited the special structure of indoor environments to accelerate 3D acquisition and recognition with a low-end hand-held scanner. Nan et al. [46] also presented an algorithm for recognition and reconstruction of scanned 3D indoor scenes, reinforcing classification by a template fitting step to provide a scene reconstruction. Dai et al. [47] designed an easy-to-use and scalable RGB-D capture system that achieved good performance on several 3D scene understanding tasks, including 3D object classification, semantic voxel labeling, and CAD model retrieval. Nguyen et al. [48] built a robust annotation tool that effectively and conveniently enabled segmentation and annotation of massive 3D data.

### 2.4 Our approach

There are many other mesh segmentation techniques that are not directly related to our work. The reader is referred to survey papers for more details [21, 22]. Our approach falls into the category of low-level primitive extraction by hierarchical clustering. Instead of considering only fitting errors, we also take shape priority and boundary regularity into account, which lead to better segmentation results than representative competing counterparts.

## 3 Overview

Let $\mathcal{M} = \{\mathcal{T}, \mathcal{V}\}$ denote the input mesh surface, where $\mathcal{T} = \{t_1, \ldots, t_{n_f}\}$ is the set of triangles, and $\mathcal{V} = \{v_1, \ldots, v_{n_v}\}$ is the set of vertices. Each $t_i$ has three vertices $\{v_{i,j}\}_{j=1}^{3}$. Our goal is to partition $\mathcal{M}$ into a set of non-overlapping components, or clustering regions, denoted $\mathcal{R} = \{\mathcal{R}_i\}_{i=1}^{n}$ such that each region $\mathcal{R}_i$ can be approximated by a best-fitting simple primitive $\{\mathcal{P}_i\}$ (a plane, cylinder, sphere, or cone). Hence $\mathcal{R}_i$ consists of a set of connected triangles $\{t_{i,k}\}_{k=1}^{n_i}$ such that $\mathcal{T} = \bigcup_{i=1}^{n} R_i$. We assign a cluster id $\mathcal{C}_i$ to each region $\mathcal{R}_i$, and also assign this cluster id $\mathcal{C}_i$ to every triangle $t_{i,k} (k = 1, \ldots, n_i)$ inside this region.

Our segmentation algorithm is an improved version of the well-known *hierarchical face clustering* (HFC) approach [19]. There, at the beginning, each triangle $t_i$ is considered to be a clustering region, with cluster id set to the index of the triangle. A cost function is defined for pairs of adjacent clusters, which measures the error on fitting a single primitive to the two clusters. All pairs of adjacent clusters are fed into a priority queue, such that the pair with smallest fitting error are at the head of the queue. On each iteration, the pair at the head of the queue is removed, and the two clusters in this pair are merged into a new cluster. The cost values of all pairs of clusters affected by the merging operation are updated in the queue. The algorithm terminates when all clusters are well represented, for example, stopping when the total error increases as the number of clusters decreases, or the error of some cluster exceeds a user-specified threshold.

Figure 2 illustrates an example of the hierarchical clustering process. Our cost function for merging neighboring clusters also considers both primitive priority and boundary smoothness in a unified framework, as detailed in the next section.

## 4 Cost function

In this section, we present the details of the cost function used for merging two adjacent clusters. The following principles are key to the design of our cost function: (i) try to find a simple primitive with the highest priority that best fits the merged clusters, (ii) take boundary smoothness into consideration before and after merging, and (iii) consider fidelity to avoid unnecessary merging. The cost function for merging the $i$-th and $j$-th clusters is defined as follows:

$$E_{\text{clustering}}^{(i,j)} = E_{\text{pri}}^{(i,j)} + \beta E_{\text{smth}}^{(i,j)} \tag{1}$$

The meaning of each term above is described next.

### 4.1 Fitting energy

We first consider use of a simple primitive $\mathcal{P}$ to approximate the merged $i$-th and $j$-th clusters, leading to a fitting energy defined as

$$E_{\text{pri}}^{(i,j)} = \min_{\mathcal{P}} \alpha^{\mathcal{P}} E_{\text{fit}}^{\mathcal{P}} \tag{2}$$

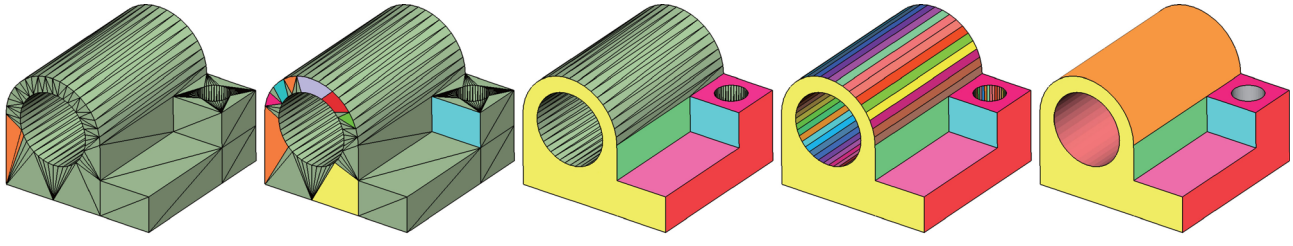
where the term $E_{\text{fit}}^{\mathcal{P}}$ evaluates the approximation of the merged clusters by primitive $\mathcal{P}$, and the parameter $\alpha^{\mathcal{P}}$ takes the priority and fidelity of primitive approximation into account. Note that both terms are defined with respect to the $i$-th and $j$-th cluster, but we omit $i, j$ here for brevity. These two terms are analysed further below.

#### *4.1.1 Fitting by a fixed primitive*

The approximation error of the merged clusters by a fixed primitive $\mathcal{P}$ can be evaluated by

$$E_{\text{fit}}^{\mathcal{P}} = \frac{1}{m} \sum_{k=1}^{m} \text{dist}(\boldsymbol{p}_k, \mathcal{P})^2 \tag{3}$$

where $\{\boldsymbol{p}_k\}_{k=1}^{m}$ includes the vertices, barycenters of



(a) (b) (c) (d) (e)

**Fig. 2** Clustering process for the Joint model with 445 faces. (a) Grey-green: input mesh. Orange: pair of faces with smallest clustering cost. (b) After reducing to 412 clusters. Planes are formed due to their higher priority. (c) At 166 clusters. All large planes have been extracted. (d) At 65 clusters. Cylinders are detected. (e) At 12 clusters; the final result corresponding to the ground truth. Clusters are randomly coloured.

triangles, and midpoints of all edges belonging to the $i$-th and $j$-th clusters, and dist is the Euclidean distance from data point $\boldsymbol{p}_k$ to the fitted primitive $\mathcal{P}$. Note that since our primitives are all simple quadrics, the distance function dist$(\cdot)$ has an explicit representation. Readers are referred to Refs. [11, 19, 49] for more details.

#### *4.1.2 Priority and fidelity*

Since the target primitive $\mathcal{P}$ can be a plane, cylinder, sphere or cone, we adopt a parameter $\alpha^{\mathcal{P}}$ to control the priority and fidelity of the choice, defined as

$$\alpha^{\mathcal{P}} = \alpha^{\mathcal{P}}_{\text{pri}} \alpha^{\mathcal{P}}_{\text{fide}} \tag{4}$$

Here, $\alpha^{\mathcal{P}}_{\text{pri}}$ is based on the priority of the primitive type, i.e., plane > cylinder > sphere > cone, and is set to

$$\alpha^{\mathcal{P}}_{\text{pri}} = \begin{cases} 0.8, & \text{if } \mathcal{P} \text{ is a plane} \\ 0.9, & \text{if } \mathcal{P} \text{ is a cylinder} \\ 1.0, & \text{if } \mathcal{P} \text{ is a sphere or cone} \\ \infty, & \text{if } \mathcal{P} \text{ is an undesired case} \end{cases} \tag{5}$$

The term $\alpha^{\mathcal{P}}_{\text{fide}}$ controls the fidelity of the merging, and is defined as

$$\alpha^{\mathcal{P}}_{\text{fide}} = \begin{cases} \infty, & \text{if } \alpha^{(i)}_{\text{pri}} + \alpha^{(j)}_{\text{pri}} < 2\alpha^{\mathcal{P}}_{\text{pri}} \text{ and} \\ & |\text{Area}(i) - \text{Area}(j)| \geqslant 3\text{Area}(i \bigcup j)/4 \\ 1, & \text{otherwise} \end{cases} \tag{6}$$

where $\alpha^{(i)}_{\text{pri}}$ is the priority value of the current primitive approximating the $i$-th cluster. Equation (6) works as follows. Suppose that the current $i$-th and $j$-th clusters are approximated by different types of primitives, e.g., a plane and a cylinder, and the area of the $i$-th cluster is far bigger than that of the $j$-th cluster. Then $\alpha^{\mathcal{P}}$ ensures we choose a plane to approximate the merged area rather than a cylinder. See Fig. 3(b) for an illustration.

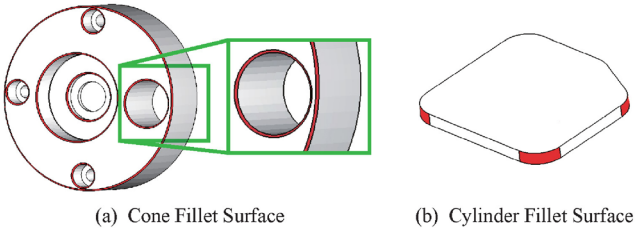



**Fig. 3** Fillet surfaces (red). (a) Approximate conical fillet surfaces between plane and cylinder clusters (white). The area of the planes and cylinders is much larger than that of the fillet part. (b) Approximate cylindrical fillet surfaces between plane clusters. The area of the planar part is much larger than that of the fillet part. We avoid merging such fillet surfaces clusters.

We show how $E^{\mathcal{P}}_{\text{fit}}$ and $\alpha^{\mathcal{P}}$ work together in Fig. 4. In Fig. 4(a), two clusters (red and blue) are to be merged; the final choice of primitive to approximate the merged area is a cylinder as shown in Fig. 4(b). Figures 4(c)–4(f) show the approximation result if instead a plane, cylinder, sphere, or cone is used, respectively, without taking $\alpha^{\mathcal{P}}$ into account; numerically a cone gives the smallest $E^{\mathcal{P}}_{\text{fit}}$. However, when taking $\alpha^{\mathcal{P}}$ into account, since the cone has the lowest priority, a cylinder gives the lowest $\alpha^{\mathcal{P}} E^{\mathcal{P}}_{\text{fit}}$. Note that in this example, since the red cluster and the blue cluster in Fig. 4(a) have comparable areas, the term $\alpha^{\mathcal{P}}_{\text{fide}}$ has little effect.

### 4.2 Boundary smoothness

Smooth segmentation boundaries are necessary in CAD models. Existing work always processes irregular boundaries in a post-processing step. Here we take boundary smoothness into consideration when deciding merging of $i$-th and $j$-th clusters. This is done by evaluating

$$E^{i,j}_{\text{smth}} = \sum_k \theta^{i \cup j}_k - \sum_l \theta^{(i)}_l - \sum_n \theta^{(j)}_n + \text{Rate}(B/D) \tag{7}$$

where $\theta^{i \cup j}_k$ is the clockwise angle between the $k^{\text{th}}$ and $k+1^{\text{th}}$ boundary edge in the merged region formed by the $i$-th and $j$-th clusters, $\theta^{(i)}_l$ is the clockwise angle between the $l^{\text{th}}$ and $l+1^{\text{th}}$ boundary edge of the $i$-th cluster, and similarly for $\theta^{(j)}_n$. Equation (7) encourages merging of two clusters with rough

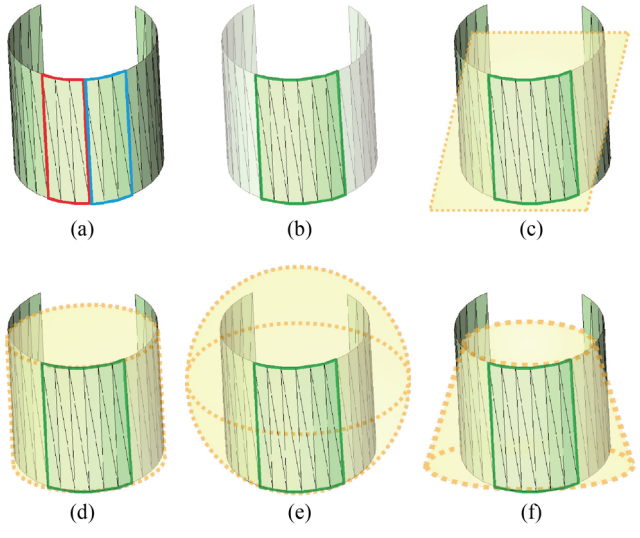



**Fig. 4** (a) Two clusters (red and blue) to be merged; (b) the cylinder used as the final choice of primitive to approximate the merged area; (c)–(f) show the resulting primitive if instead the type is fixed as a plane, a cylinder, a sphere, and a cone.

boundaries into a region with smoother boundaries. Very rough boundaries with zigzags will yield a large sum of the turning angles of the edges (see Fig. 5). Rate$(B/D)$ = (total boundary length)/(box diagonal length) is a penalty, which reflects the complexity of the clustering results relative to the original input model.

The parameter $\beta$ is set to balance the magnitude of the energy terms $E^{i,j}_{\text{smth}}$ and $E^{i,j}_{\text{pri}}$:

$$\beta = E_{\text{pri}}/(4\pi) \tag{8}$$

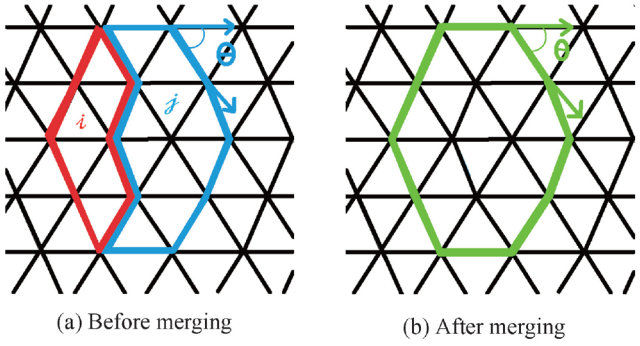



**Fig. 5** Sum of boundary turning angles. (a) Before merging, clusters $i$ (red) and $j$ (blue). (b) After merging, cluster $(i, j)$ (green). Successive boundary edges make a turning angle, and a cluster with smooth boundaries has a smaller sum of edge turning angles.

## 5 Experimental results

### 5.1 Background

The proposed algorithm was implemented using the open-source platform *Graphite* [①]. We have validated our algorithm on a wide range of input meshes including both tessellated CAD models and scanned mechanical/organic shapes. All results shown in this section were produced automatically without user interaction. The results were produced on a machine with an Intel Core i7-7700 CPU with 16 GB RAM and Windows 10 operating system. In the following, we perform a detailed analysis of our method, as well as comparing it to several representative approaches, *hierarchical face clustering* (HFC) [18], *hierarchical mesh segmentation* (HMS) [19], *variational mesh segmentation* (VMS) [11], and *feature-aligned segmentation* (FAS) [20].

### 5.2 Analysis & comparison

First, we compare our method with the competing algorithms HMS [19], VMS [11], and FAS [20] in Fig. 6. All are able to extract complex primitives rather than planar structures. All the algorithms were tested on different types of input: the Chess and the Sword models are tessellated CAD models, the Blade is a scanned mechanical model, and the Bone model is scanned freeform shapes. The segmentation results shown have the same number of clusters for each model.

All competing algorithms work well generally for models with simple quadric primitives such as Chess. However, HMS and FAS cannot segment the base of Chess successfully because two primitives are smoothly connected. Our algorithm obtains the optimal result, as does VMS, but it is more efficient since only hierarchical clustering is performed. The Sword and Blade models have more complicated structures, for which the other methods either produce wrong clusters (HMS and VMS) or cannot segment simple primitives with smooth blending regions (FAS). Our approach outperforms the others as we consider both approximation error and regularity of the segmentation boundary simultaneously.

When processing organic models, our algorithm


**Fig. 6** Comparisons, left to right, to HMS [19], VMS [11], FAS [20], ours, and a color map. Top to bottom: tessellated CAD models Chess and Sword, scanned mechanical model Cup, and organic shape Bone. Segmentation patches are randomly coloured.

① http://alice.loria.fr/software/graphite

can detect better, more meaningful segmentation boundaries than the other methods. For example, in the simple Bone model, HMS, VMS, and our method produce correct segmentations using five clusters, while FAS gives unsatisfactory output. Our results exhibit better boundary smoothness in such examples. Our method achieves better output because we jointly perform primitive detection and boundary regularization, even though the segmented patches are not regular primitives. Further results of our approach are shown in Fig. 7.

Earlier algorithms either extract only simple primitives (plane, sphere, and cylinder) [18, 19], which is too limited, or fit general quadrics [11], which introduces unwanted primitives (such as hyperboloids of one or two sheets). In our approach, we include the cone surface as a basic primitive, which is an improvement over the HFC framework. Benefits of providing conical surfaces can be seen in Fig. 8. HMS cannot detect the cone at the top of the Screw and outputs incorrect clusters at the connection between

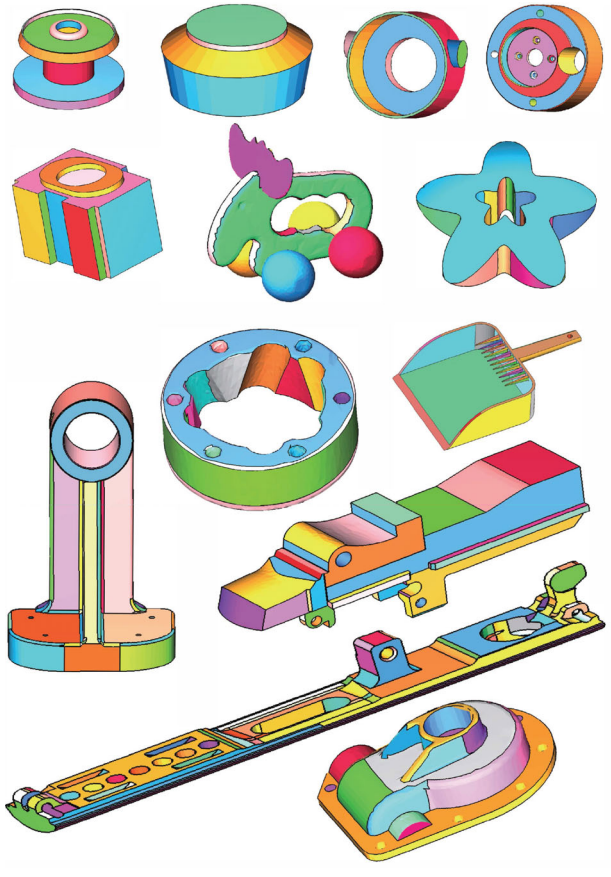


**Fig. 7** Further segmentation results produced by our algorithm. Segmentation patches are randomly coloured.

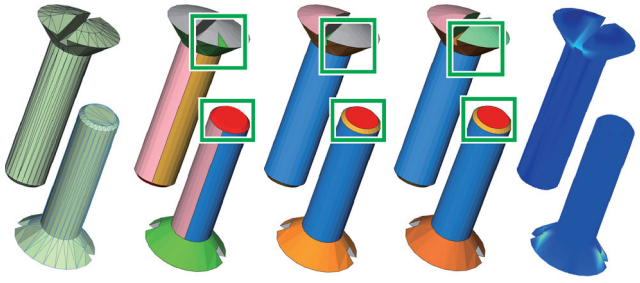


**Fig. 8** Comparisons to HMS and VMS methods, for cone fitting. Left to right: input tessellated CAD model with sparse triangulation, results of HMS, VMS, our method, and colormap.

the two primitives in red box. While VMS works as well as our method, it sometimes becomes stuck in local minima due to its random initialization, so success of the algorithm cannot be guaranteed. In such cases, user interaction is required to indicate where to insert or delete new clusters.

### 5.3 Effectiveness of boundary smoothing

To demonstrate the effectiveness of our newly introduced boundary regularization term, we carried out additional comparisons with HFC [18] and HMS [19] algorithms, which are also based on hierarchical clustering. HFC only detects planar primitives and forces each cluster to be as nearly circular as possible. To make a fair comparison with HFC [18], we only enabled planar primitive fitting and disabled the other higher order primitives. We tested both HFC and our method on the Venus model, which has a very irregular low resolution triangulation.

As shown in Fig. 9, although HFC tends to avoid sharp boundary changing in angle, the clusters (top row) are poor-grouped and hard to be satisfactory. In

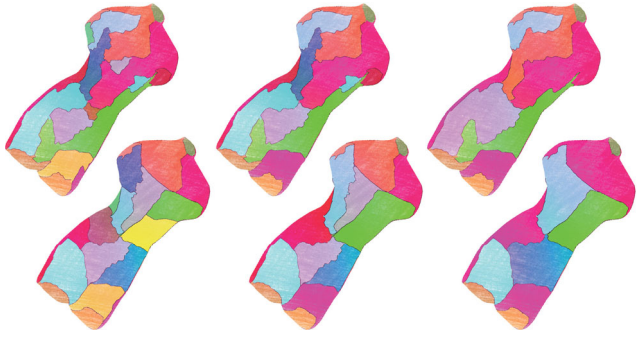


**Fig. 9** Boundary regularization comparison to the HFC method [18] using the Venus_Body model with 5672 faces. Top: HFC results, bottom: our results, each method using its own boundary optimization. Left to right: with 18, 13, and 8 clusters.

contrast, the result of our algorithm (bottom row) is more consistent with the distribution of the structure of human body. We can also detect better meaningful segmentation boundaries. At the same time, our results exhibit better boundary smoothness.

Next, we compared to HMS by only enabling plane, cylinder, and sphere primitives. Figure 10 shows several intermediate results of both methods with the same number of clusters. We found that during the whole iteration, our algorithm always produces better results, avoiding incorrect clusters and performing boundary regularization. In the process of cylinder forming and merging, HMS produces many irregular results while ours are much better. We give a quantitative comparison in terms of boundary smoothness in Table 1, by evaluating the total length of boundary edges of the corresponding segmentation.

### 5.4 Effectiveness of primitive priority

To demonstrate the effectiveness of our new priority term, we first compare our approach to HMS [19] using a simple CAD model, the Joint, as shown in Fig. 11. Because of the priority parameter, the order of the clustering changes significantly during iteration. Our algorithm tends to extract larger clusters of simple primitives as early as possible.

To explore the effectiveness of our method in terms of fidelity, we tested both HMS and our method on the Anchor model and the Fandisk model, as shown in Fig. 12. For the Anchor, HMS cannot segment the structure formed by a plane and cylinder, which is a common transitional design in industrial CAD models. When the final number of primitives to be fitted is specified, it splits a well-grouped cylinder to add a new cluster, rather than the grooved structure forming by a pair of orthogonal planes. As for the Fandisk model, the same situation occurs for a more complicated structure comprising a plane smoothly blended with two cylinders. Because these two models are composed of exact simple quadric primitives, our method can recover the original ground truth structure exactly. To summarize, our method avoids producing undesired clustering in transition regions, and so provides more reasonable results.

**Table 1** Boundary statistics. $|\mathcal{R}|$=number of clusters. $B/D$ rate= (total boundary length)/(box diagonal length), which reflects complexity of clustering results relative to the input model

| Model | Methods | $B/D$ rate | | |
|---|---|---|---|---|
| Venus | $\|\mathcal{R}\|$ | 18 | 13 | 8 |
| | HFC | 7.424 | 6.796 | 5.316 |
| | Ours | 4.630 | 3.197 | 2.061 |
| Sample | $\|\mathcal{R}\|$ | 50 | 28 | 17 |
| | HMS | 135.550 | 95.677 | 78.566 |
| | Ours | 115.245 | 81.911 | 72.652 |
| Rockerarm | $\|\mathcal{R}\|$ | 20 | 16 | 10 |
| | HMS | 11.973 | 9.390 | 6.286 |
| | Ours | 7.488 | 6.666 | 5.473 |

### 5.5 Comparison with deep learning methods

Deep learning has been widely used in various graphics and geometry processing applications. To demonstrate the effectiveness and understandability of our results, we compare our approach with work on component segmentation (LMVCNN) [50] and human body segmentation STC [51] using two simple models, the Ant and the Bracket, which have clear branching structures for semantic and patch-based segmentation.

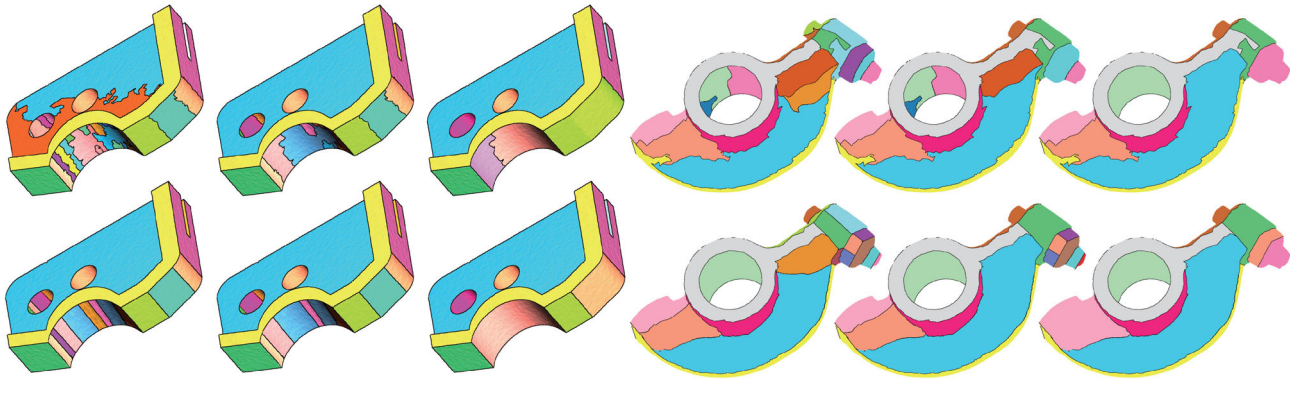


**Fig. 10** Comparison to the HMS method [19]. Left: sample model with 50, 28, 17 clusters. Right: Rockerarm model with 20, 16, 10 clusters. Top: HMS results without boundary regularization. Bottom: our results.

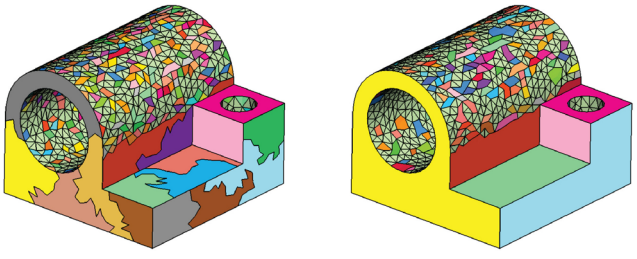


**Fig. 11** Comparison to HMS in terms of priority, using the Joint model with 6060 faces. Left: Attene's result without primitive priority. Right: ours with primitive priority.

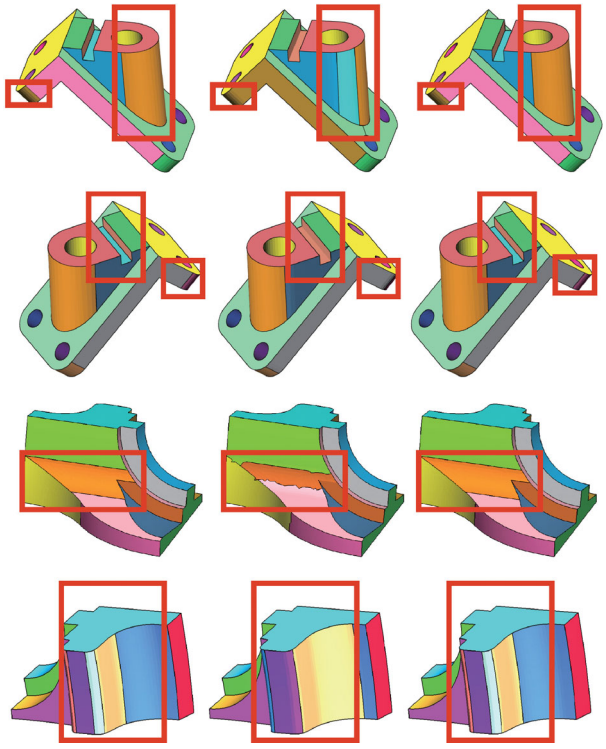


**Fig. 12** Comparison to HMS in terms of primitive priority using the Anchor (top) and the Fandisk (bottom) models. Left to right: ground truth, HMS result, and our result. The ground truth was provided by experts using manually segmentation.

As is shown in Fig. 13, deep learning relies heavily on functional relationships within the model. These methods are good at handling functional parts, such as the body and aircraft wings in their paper, but cannot recognize the many legs and antennae of the Ant model. For standard CAD models, they also fail to produce meaningful results. On the contrary, our algorithm based on primitive surface fitting can segment semantic structure well and obtains very reasonable results. It also shows that our algorithm can be applied to a wider range of models without learning.

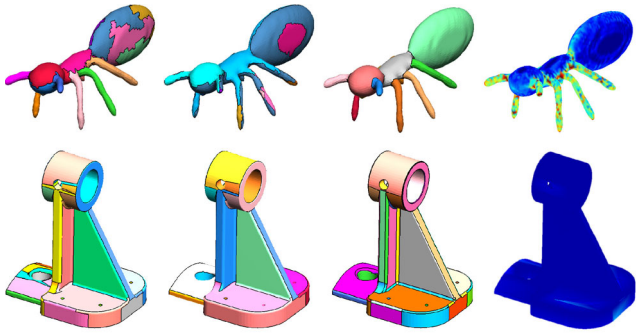



**Fig. 13** Comparison to deep learning approaches using the Ant model. Left to right: (a) STC, (b) LMVCNN, (c) our result, and (d) color map.

### 5.6 Global regularization

Although our algorithm can ensure that each cluster uses the closest quadric for fitting and parametric form for storage, in practical industrial production, global relationships between disconnected parts are often considered to be an important issue, such as multiple parts forming subcomponents, and different parts aligned with each other via symmetry (e.g., parallel or orthogonal axes). We use the symmetry detection method in Ref. [52] to identify all components with the same transformation (translation, rotation), and then determine whether a subcomponent is formed according to the connection relationship. We also consider that objects with geometric properties such as discrete cylindrical and coaxial circular surfaces should have global symmetry.

As shown in Fig. 14, each part is obtained by fitting a corresponding quadric primitive. However, the result appears too fragmentary and cumbersome to be used easily and conveniently. In Fig. 14(c), we detect that the cylinder and plane on the edge exhibit the same rotational transformation. By their spatial relationships, we judge that two adjacent cylinders

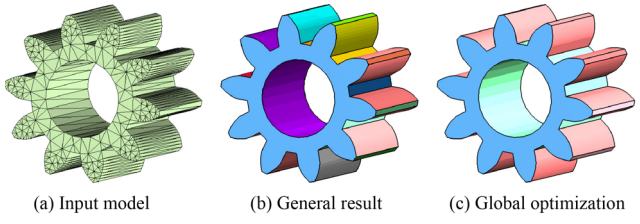



**Fig. 14** Global optimization results for the Pinion model. Left to right: (a) input model, (b) initial result, and (c) result with global alignment.

and a plane can form a tooth subcomponent and in this model, all parts on the rim just repeat this tooth subcomponent. The gap between any two teeth, which we consider to be a discrete partition of a complete cylinder, form a hollow cylinder together with the internal entity. The regularized result is more compact and concise, so is more convenient for subsequent engineering processing and applications.

### 5.7 Performance

The running time required for all demonstrated models is presented in Table 2. The time taken is affected by both the size of the input model and the number of the final clusters. Our algorithm is slightly slower than Attene's HMS [19] due to the additional computation, but our results are improved significantly. Compared to optimization-based approaches [11, 20], our algorithm is much faster. However, our implementation is not fully optimized, and we hope to further improve the performance in future work.

**Table 2** Timings (s)

| Model | $n_f$ | $\|\mathcal{R}\|$ | HMS | VMS | FAS | Ours |
|---|---|---|---|---|---|---|
| Blade | 78k | 35 | 71.35 | 100.54 | 122.12 | 87.61 |
| Chess | 24k | 8 | 14.84 | 25.45 | 27.48 | 16.43 |
| Cup | 5.7k | 5 | 2.51 | 5.82 | 6.64 | 3.46 |
| Ant | 11.7k | 12 | 12.28 | 20.62 | 22.40 | 12.58 |
| Dragknob | 0.3k | 6 | <1 | 2.63 | 2.96 | <1 |
| Couplingdown | 3.7k | 36 | 7.76 | 11.81 | 13.55 | 8.98 |
| Elk | 10k | 13 | 15.79 | 24.45 | 28.15 | 18.35 |
| Bracket | 37k | 38 | 22.19 | 47.81 | 45.73 | 31.48 |
| Dustpan | 4v | 70 | 1.57 | 9.14 | 9.82 | 2.75 |
| Boat | 4k | 57 | 7.66 | 15.31 | 17.88 | 7.84 |
| Screw | 0.3k | 9 | <1 | 4.62 | 4.68 | <1 |
| Rockerarm | 7k | 20 | 9.24 | 21.04 | 29.87 | 5.92 |
| Joint | 6k | 12 | 4.62 | 14.88 | 19.87 | 7.18 |
| Fandisk | 13k | 22 | 11.19 | 21.88 | 23.95 | 16.99 |
| Joint | 0.5k | 12 | <1 | 1.94 | 2.37 | <1 |
| Sword | 80k | 8 | 100.37 | 190.17 | 192.75 | 132.47 |
| Bone | 30k | 5 | 22.63 | 34.14 | 34.34 | 25.47 |
| Spool | 1.3k | 13 | <1 | 2.22 | 2.74 | 1.34 |
| Rotor | 1.2k | 10 | 1.74 | 5.92 | 6.39 | 1.77 |
| oblong | 0.8k | 24 | <1 | 1.56 | 1.47 | <1 |
| Star | 10k | 27 | 12.63 | 26.96 | 28.93 | 15.58 |
| Rollingstage | 100k | 30 | 72.77 | 124.82 | 139.47 | 90.76 |
| M145 | 27k | 60 | 16.12 | 28.82 | 26.66 | 16.55 |
| Casting | 37k | 16 | 18.50 | 24.23 | 24.45 | 19.38 |
| Sample | 26.7k | 17 | 16.12 | 24.99 | 28.15 | 22.04 |
| Venus | 5.7k | 15 | 9.41 | 14.50 | 15.14 | 6.03 |
| Anchor | 1k | 28 | <1 | 1.56 | 1.78 | <1 |
| Moai | 20k | 28 | 25.22 | 37.6 | 34.00 | 26.76 |

### 5.8 Limitations

Although our method can produce acceptable results for various inputs, it depends on the fact that the detected primitives themselves must have certain geometric properties. This means that a segmented cluster, to some extent, should be approximated by a simple quadric shape properly. Figure 15 shows an unsatisfactory example, which exhibits no clear structure. Therefore, our algorithm cannot work well and produces unsatisfactory output. Other competing algorithms cannot deal with such input either.

**Fig. 15** An unsatisfactory example. Left to right: results of HMS [19], VMS [11], FAS [20], our method, and colormap, for 28 clusters.

## 6 Conclusions & future work

We have proposed a simple primitive detection algorithm, which is based on hierarchical clustering [18, 19]. We improve the original algorithm in three ways. Firstly, we add cones as a new primitive, which improves the approximation flexibility of the framework. Next, we introduce primitive priority to encourage simpler primitives. Finally, we propose a new boundary regularization term which improves the results significantly. Our algorithm works well on a wide range of inputs. However, we cannot produce satisfactory output for shapes without clear structure, as shown in Fig. 15. This is a common drawback of many mesh segmentation algorithms.

There are several promising ways in which this work could be extended. Firstly, the current algorithm performs only local processing with little consideration of global relationships between non-adjacent clusters. We plan to take more global constraints (angular relationships, symmetry, and so on) into account to further improve the regularity of the detected primitives. Secondly, although we can extract correct primitives for many CAD meshes, we do not convert them into a solid model that can be processed in modeling systems such as Solidworks. Such conversion is very important for many reverse

engineering applications. In the future, we hope to put more effort into completing the whole pipeline for reverse engineering.

## Acknowledgements

This work was supported by the National Natural Science of Foundation for Outstanding Young Scholars (12022117), the National Natural Science Foundation of China (61872354), the Beijing Natural Science Foundation (Z190004), and the Intelligent Science and Technology Advanced subject project of University of Chinese Academy of Sciences (115200S001).

## References

[1] Várady, T.; Martin, R. R.; Cox, J. Reverse engineering of geometric models: An introduction. *Computer-Aided Design* Vol. 29, No. 4, 255–268, 1997.

[2] Petitjean, S. A survey of methods for recovering quadrics in triangle meshes. *ACM Computing Surveys* Vol. 34, No. 2, 211–262, 2002.

[3] Luo, L.; Baran, I.; Rusinkiewicz, S.; Matusik, W. Chopper: Partitioning models into 3D-printable parts. *ACM Transactions on Graphics* Vol. 31, No. 6, Article No. 129, 2012.

[4] Hu, R.; Li, H.; Zhang, H.; Cohen-Or, D. Approximate pyramidal shape decomposition. *ACM Transactions on Graphics* Vol. 33, No. 6, Article No. 213, 2014.

[5] Lien, J. M.; Amato, N. M. Approximate convex decomposition of polyhedra and its applications. *Computer Aided Geometric Design* Vol. 25, No. 7, 503–522, 2008.

[6] David, C.-S.; Pierre, A.; Mathieu, D. Variational shape approximation. *ACM Transactions on Graphics* Vol. 23, No. 3, https://doi.org/10.1145/1015706.1015817, 2004.

[7] Wu, J.; Kobbelt, L. Structure recovery via hybrid variational surface approximation. *Computer Graphics Forum* Vol. 24, No. 3, 277–284, 2005.

[8] Julius, D.; Kraevoy, V.; Sheffer, A. D-charts: Quasi-developable mesh segmentation. *Computer Graphics Forum* Vol. 24, No. 3, 581–590, 2005.

[9] Simari, P. D.; Singh, K. Extraction and remeshing of ellipsoidal representations from mesh data. In: Proceedings of Graphics Interface, 161–168, 2005.

[10] Lu, L.; Choi, Y. K., Wang, W. P.; Kim, M. S. Variational 3D shape segmentation for bounding volume computation. *Computer Graphics Forum* Vol. 26, No. 3, 329–338, 2007.

[11] Yan, D. M.; Wang, W. P.; Liu, Y.; Yang, Z. W. Variational mesh segmentation via quadric surface fitting. *Computer-Aided Design* Vol. 44, No. 11, 1072–1082, 2012.

[12] Kim, Y. M.; Mitra, N. J.; Yan, D. M.; Guibas, L. Acquiring 3D indoor environments with variability and repetition. *ACM Transactions on Graphics* Vol. 31, No. 6, Article No. 138, 2012.

[13] Katz, S.; Tal, A. Hierarchical mesh decomposition using fuzzy clustering and cuts. *ACM Transactions on Graphics* Vol. 22, No. 3, 954–961, 2003.

[14] Ji, Z.; Liu, L.; Chen, Z.; Wang, G. Easy mesh cutting. *Computer Graphics Forum* Vol. 25, No. 3, 283–291, 2006.

[15] Lavoué, G.; Dupont, F.; Baskurt, A. A new CAD mesh segmentation method, based on curvature tensor analysis. *Computer-Aided Design* Vol. 37, No. 10, 975–987, 2005.

[16] Yan, D. M.; Liu, Y.; Wang, W. P. Quadric surface extraction by variational shape approximation. In: *Geometric Modeling and Processing - GMP 2006. Lecture Notes in Computer Science, Vol. 4077.* Kim, M. S.; Shimada, K. Eds. Springer Berlin Heidelberg, 73–86, 2006.

[17] Le, T.; Bui, G.; Duan, Y. A multi-view recurrent neural network for 3D mesh segmentation. *Computers & Graphics* Vol. 66, 103–112, 2017.

[18] Garland, M.; Willmott, A.; Heckbert, P. S. Hierarchical face clustering on polygonal surfaces. In: Proceedings of the Symposium on Interactive 3D Graphics, 49–58, 2001.

[19] Attene, M.; Falcidieno, B.; Spagnuolo, M. Hierarchical mesh segmentation based on fitting primitives. *The Visual Computer* Vol. 22, No. 3, 181–193, 2006.

[20] Zhuang, Y. X.; Dou, H.; Carr, N., Ju, T. Feature-aligned segmentation using correlation clustering. *Computational Visual Media* Vol. 3, No. 2, 147–160, 2017.

[21] Shamir, A. A survey on mesh segmentation techniques. *Computer Graphics Forum* Vol. 27, No. 6, 1539–1556, 2008.

[22] Kaiser, A.; Ybanez Zepeda, J. A.; Boubekeur, T. A survey of simple geometric primitives detection methods for captured 3D data. *Computer Graphics Forum* Vol. 38, No. 1, 167–196, 2019.

[23] Fitzgibbon, A. W.; Eggert, D. W.; Fisher, R. B. High-level cad model acquisition from range images. *Computer-Aided Design* Vol. 29, No. 4, 321–330, 1997.

[24] Vieira, M.; Shimada, K. Surface mesh segmentation and smooth surface extraction through region growing. *Computer Aided Geometric Design* Vol. 22, No. 8, 771–792, 2005.

[25] Lloyd, S. Least squares quantization in PCM. *IEEE Transactions on Information Theory* Vol. 28, No. 2, 129–137, 1982.

[26] Schnabel, R.; Wahl, R.; Klein, R. Efficient RANSAC for point-cloud shape detection. *Computer Graphics Forum* Vol. 26, No. 2, 214–226, 2007.

[27] Li, H.; Wan, G. W.; Li, H. H.; Sharf, A.; Xu, K.; Chen, B. Q. Mobility fitting using 4D ransac. *Computer Graphics Forum* Vol. 35, No. 5, 79–88, 2016.

[28] Lee, Y.; Lee, S.; Shamir, A.; Cohen-Or, D.; Seidel, H. P. Mesh scissoring with minima rule and part salience. *Computer Aided Geometric Design* Vol. 22, No. 5, 444–465, 2005.

[29] Rodrigues, R. S. V.; Morgado, J. F. M.; Gomes, A. J. P. Part-based mesh segmentation: A survey. *Computer Graphics Forum* Vol. 37, No. 6, 235–274, 2018.

[30] Lai, Y. K.; Hu, S. M.; Martin, R. R., Rosin, P. L. Rapid and effective segmentation of 3D models using random walks. *Computer Aided Geometric Design* Vol. 26, No. 6, 665–679, 2009.

[31] Lafarge, F.; Keriven, R.; Brédif, M. Insertion of 3-D-primitives in mesh-based representations: Towards compact models preserving the details. *IEEE Transactions on Image Processing* Vol. 19, No. 7, 1683–1694, 2010.

[32] Chen, X. B.; Golovinskiy, A., Funkhouser, T. A benchmark for 3D mesh segmentation. In: Proceedings of the ACM SIGGRAPH 2009 papers, Article No. 73, 2009.

[33] Zheng, Y. Y.; Tai, C. L., Au, O. K. C. Dot scissor: A single-click interface for mesh segmentation. *IEEE Transactions on Visualization and Computer Graphics* Vol. 18, No. 8, 1304–1312, 2012.

[34] Au, O. K.-C.; Zheng, Y.; Chen, M.; Xu, P.; Tai, C.-L. Mesh segmentation with concavity-aware fields. *IEEE Transactions on Visualization and Computer Graphics* Vol. 18, No. 7, 1125–1134, 2012.

[35] Liu, R.; Zhang, H. Mesh segmentation via spectral embedding and contour analysis. *Computer Graphics Forum* Vol. 26, No. 3, 385–394, 2007.

[36] Theologou, P.; Pratikakis, I.; Theoharis, T. Unsupervised spectral mesh segmentation driven by heterogeneous graphs. *IEEE Transactions on Pattern Analysis and Machine Intelligence* Vol. 39, No. 2, 397–410, 2017.

[37] Ohtake, Y., Belyaev, A., Seidel, H. P. Ridge-valley lines on meshes via implicit surface fitting. *ACM Transactions on Graphics* Vol. 23, No. 3, 609–612, 2004.

[38] Cao, Y. H.; Yan, D. M.; Wonka, P. Patch layout generation by detecting feature networks. *Computers & Graphics* Vol. 46, 275–282, 2015.

[39] Dai, C.; Wang, C. C. L.; Wu, C.; Lefebvre, S.; Fang, G.; Liu, Y.-J. Support-free volume printing by multi-axis motion. *ACM Transactions on Graphics* Vol. 37, No. 4, Article No. 134, 2018.

[40] Chen, X.; Li, H.; Fu, C.-W.; Zhang, H.; Daniel, C.-O.; Chen, B. 3D fabrication with universal building blocks and pyramidal shells. *ACM Transactions on Graphics* Vol. 37, No. 6, Article No. 189, 2018.

[41] Kalogerakis, E.; Averkiou, M.; Maji, S.; Chaudhuri, S. 3D shape segmentation with projective convolutional networks. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 6630–6639, 2017.

[42] Shu, Z. Y.; Qi, C. W.; Xin, S. Q.; Hu, C.; Wang, L.; Zhang, Y.; Liu, L. G. Unsupervised 3D shape segmentation and co-segmentation via deep learning. *Computer Aided Geometric Design* Vol. 43, 39–52, 2016.

[43] Guo, K.; Zou, D. Q.; Chen, X. W. 3D mesh labeling via deep convolutional neural networks. *ACM Transactions on Graphics* Vol. 35, No. 1, Article No. 3, 2015.

[44] Charles, R. Q.; Su, H.; Kaichun, M.; Guibas, L. J. Pointnet: Deep learning on point sets for 3D classification and segmentation. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 77–85, 2017.

[45] Xu, H.; Dong, M.; Zhong, Z. Directionally convolutional networks for 3D shape segmentation. In: Proceedings of the IEEE International Conference on Computer Vision, 2698–2707, 2017.

[46] Nan, L. L.; Xie, K.; Sharf, A. A search-classify approach for cluttered indoor scene understanding. *ACM Transactions on Graphics* Vol. 31, No. 6, Article No. 137, 2012.

[47] Dai, A.; Chang, A. X.; Savva, M.; Halber, M.; Funkhouser, T.; Niefiner, M. ScanNet: Richly-annotated 3D reconstructions of indoor scenes. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2432–2443, 2017.

[48] Nguyen, D. T.; Hua, B. S.; Yu, L. F.; Yeung, S. K. A robust 3D-2D interactive tool for scene segmentation and annotation. *IEEE Transactions on Visualization and Computer Graphics* Vol. 24, No. 12, 3005–3018, 2018.

[49] Taubin, G. Estimation of planar curves, surfaces, and nonplanar space curves defined by implicit equations with applications to edge and range image segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence* Vol. 13, No. 11, 1115–1138, 1991.

[50] Huang, H. B.; Kalogerakis, E.; Chaudhuri, S.; Ceylan, D.; Kim, V. G.; Yumer, E. Learning local shape descriptors from part correspondences with multiview convolutional networks. *ACM Transactions on Graphics* Vol. 37, No. 1, Article No. 6, 2018.

[51] Maron, H.; Galun, M.; Aigerman, N.; Trope, M.; Dym, N.; Yumer, E.; Kim, V.G.; Lipman. Convolutional neural networks on surfaces via seamless toric covers. *ACM Transactions on Graphics* Vol. 36, No. 4, Article No. 71, 2017.

[52] Mitra, N. J.; Guibas, L. J.; Pauly, M. Partial and approximate symmetry detection for 3D geometry. *ACM Transactions on Graphics* Vol. 25, No. 3, 560–568, 2006.

**Xiaolong Yang** received his B.S. degree in information and computing science from Northwestern Polytechnical University in 2017 and is currently pursuing his M.S. and Ph.D. degrees at the Key Laboratory of Mathematics Mechanization, Academy of Mathematics and Systems Sciences, Chinese Academy of Sciences. His research interests are in computer graphics and computer vision.

**Xiaohong Jia** is an associate professor at the Key Laboratory of Mathematics Mechanization, Academy of Mathematics and Systems Science, Chinese Academy of Sciences. She received her Ph.D. and bachelor degrees from the University of Science and Technology of China in 2009 and 2004, respectively. Her research interests include computer graphics, computer aided geometric design, and computational algebraic geometry.