Vis Comput
DOI 10.1007/s00371-017-1434-1

ORIGINAL ARTICLE

A new two-stage mesh surface segmentation method

Huayan Zhang[1] · **Chunlin Wu**[2] · **Jiansong Deng**[3] · **Zheng Liu**[4] · **Yuning Yang**[5]

Abstract Partitioning a mesh surface into several semantic components is a fundamental task in geometry processing. This paper presents a new stable and effective segmentation method, which contains two stages. The first stage is a spectral clustering procedure, while the second stage is a variational refining procedure. For spectral clustering, we construct a new Laplacian matrix which reflects more semantic information than classical Laplacian matrices. By this new Laplacian, we introduce a simple and fast spectral clustering method, which gives quite satisfying segmentation results for most surfaces and provides a good initialization for the second stage. In the second stage, we propose a variational refining procedure by a new discretization of the classical non-convex Mumford–Shah model. The variational problem is solved by efficient iterative algorithms based on alternating minimization and alternating direction method of multipliers (ADMM). The first stage provides a good initialization for the second stage, while the second stage refines the result of the first stage well. Experiments demonstrated that our method is very stable and effective compared to existing approaches. It outperforms competitive segmentation methods when evaluated on the Princeton Segmentation Benchmark.

Keywords Mesh surface segmentation · Piecewise constant function space · Spectral analysis · Mumford–Shah model · Laplacian matrix · ADMM · Augmented Lagrangian

✉ Chunlin Wu
wucl@nankai.edu.cn

Huayan Zhang
zhanghuayan@tjpu.edu.cn

Jiansong Deng
dengjs@ustc.edu.cn

1 School of Computer Science and Software Engineering, Tianjin Polytechnic University, Tianjin, China

2 School of Mathematical Sciences, Nankai University, Tianjin, China

3 School of Mathematical Sciences, University of Science and Technology of China, Hefei, China

4 Department of Information Engineering, China University of Geosciences, Beijing, China

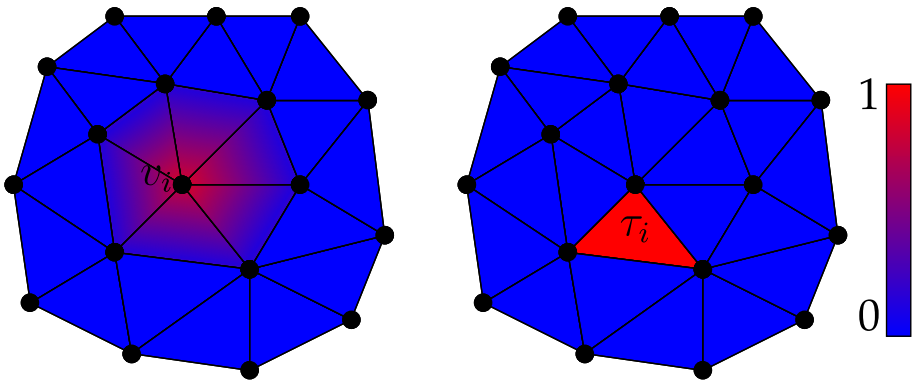
5 School of Mathematics, South China Normal University, Guangzhou, China

1 Introduction

Mesh segmentation aims at partitioning a mesh into several disjoint parts, which has become an active research topic in geometric modeling and computer graphics community. It has many applications such as parameterization, texture mapping, 3D morphing, simplification, shape retrieval, multiresolution modeling, skeleton extraction. According to demands in applications, there are two types of mesh segmentations, namely surface-type and part-type segmentations [36]. According to the type of segmentation strategies, segmentations can be classified into two types: contour-based and region-based segmentations. According to the type of clustering element, we categorize segmentations to 3 types: vertex-based, edge-based, and triangle-based segmentations. Since clustering the triangles of the mesh directly, triangle-based methods are the most popular and most methods fall into this type. Vertex-based and edge-based methods may need post-processing steps to segment the boundary triangle strips; see, e.g., Fig. 12a.

In this paper, we are interested in part-type region-based segmentation, where a mesh surface is to be partitioned into semantic components consistent with people perception. So far a wide variety of algorithms have been developed [19–21,23,28,37,50,51]. While obtaining good results, these

Published online: 19 September 2017



(a) piecewise linear basis ϕ_i on vertex v_i **(b)** piecewise constant basis ϕ_i on triangle τ_i

Fig. 1 Basis of piecewise linear function space and piecewise constant function space

methods are still either too sensitive to initialization, or not geometry-aware enough, or need post-processing steps to get real region-based segmentation.

The goal of this paper is to design a new mesh surface segmentation method. Our method consists of two stages, say, spectral clustering procedure and variational refining procedure. By constructing a new Laplacian matrix and computing its eigenvectors, we obtain a good feature space to cluster triangles on mesh surfaces. With this feature space in hand, we devise a new spectral clustering algorithm, which is simple and effective. The result in this stage is then used as an initialization of our variational refining procedure. The variational model is a new discretization of the classical Mumford–Shah model by applying piecewise constant function space (see Fig. 1 for a comparison on the basis functions) on triangle meshes, which yields a triangle clustering algorithm and avoiding post-processing steps. Besides, we use a simple alternating minimization and ADMM to solve the non-convex problem, where our spectral clustering procedure is the initialization technique. The alternating minimization is shown to have cluster points as partial optima of the energy function. Plenty of our experiments showed that our refining procedure with the spectral procedure as initialization technique works pretty well. Our two-stage method always gives satisfying segmentation results with correct human perception.

2 Related work

2.1 Mesh segmentation

Mesh segmentation is an active research topic in geometric modeling and computer graphics community. So far a wide variety of algorithms have been developed for decomposing meshes. The interested reader can refer to excellent surveys [1,36], and a benchmark for evaluating mesh segmentation algorithms [9], as well as some recent developments [3,17,19,51].

According to the objective of segmentation and demands in applications, there are two types of mesh segmentations: surface-type and part-type segmentations. In surface-type segmentation, a mesh surface is partitioned into patches with basic mathematical structures such as planes and spheres [10,44]. In part-type segmentation, a mesh surface is partitioned, based on the minima rule proposed by Hoffman et al. [16], into meaningful volumetric parts which respect human perception [3,14,21,28,42,45,51].

According to the type of segmentation strategies, segmentation methods can be classified into two types: contour-based and region-based methods. The contour-based methods aim at producing segmentation by finding the common boundaries of segments; see [3,14,45,53]. The region-based methods decompose a mesh via directly handling the disjoint parts, and can be regarded as region growing or clustering algorithms. These methods contain region growing approaches [24,32], clustering in local geometrical attributes [2,10,12,13,30,44], clustering in eigen space of Laplacian matrices [21,28,51] and two surveys [31,48], and other techniques such as Random Walks [23,50], Core Extra [20], Shape Diam [37], learning segmentation [19] and Quadric Surface Fitting [46]. Most of these methods need user interactions, especially initial inputs.

The recent progresses in part-type mesh segmentation include learning-based segmentation [15,18,19], Isoline Cut [3], and the segmentation method in [17,51]. Kalogerakis et al. [19] proposes to learn the conditional random field from a collection of labeled training meshes, and then applies it to segment a mesh. The methods in [15,18] are based on convolutional network theory to training meshes. These methods

can produce quite satisfying segmentation results. However, their computation costs are too expensive. Isoline Cut [3] proposes to construct the concave-aware segmentation field by solving a series of poisson equations and then get the isolines of the concave-aware segmentation field with higher scores as the final contour. The method is fully automatic and effective. However, it involves so many complex tricks to choose the correct isolines and is not suitable for segmenting CAD meshes. Zhang et al. [51] discretizes the Mumford–Shah image segmentation [34] in piecewise linear function space to get a vertex-based mesh segmentation method. Since clustering the vertices of a mesh, it generates unsegmented triangle strips (see the red regions in Fig. 12a), and a post-processing step is needed. In addition, as the Mumford–Shah model is a famous non-convex model, the solution severely depends on the initialization. The method in [51] completely ignores this fact and uses random initializations. Therefore, the method is quite unstable. The segmentation method in [17] first oversegments the models into approximate convex components, and then a further merging process and a boundary refining postprocess are used to achieve final segmentation. This is a greedy method. It cannot effectively merge these convex components and the computational cost is high. All these methods have been tested and compared using the benchmark [9]. While obtaining good results, most existing segmentation methods involve multiple step operations with higher computational cost. Some are still either unstable, or not geometry-aware enough, or even need user interaction.

Different from most existing segmentation methods, in the paper, we propose a new two-stage mesh segmentation method. The first stage is a simple and effective spectral clustering procedure by introducing a new Laplacian matrix, while the second stage is an effective variational refining procedure by a new discretization of the classical non-convex Mumford–Shah model.

2.2 Our contributions and paper organization

The contributions of this paper are as follows:

- In the first stage of our method:
  - We constructed a new Laplacian matrix which reflects more semantic information than classical Laplacian matrices;
  - A simple and fast spectral clustering method is introduced, which gives quite satisfying segmentation results for most surfaces and provides a good initialization for the second stage.
- In the second stage of our method:
  - A variational refining procedure is proposed by a new discretization of the classical non-convex Mumford–Shah model;
  - We proposed to solve the variational problem by an efficient iterative algorithm based on alternative minimization and augmented Lagrangian method;
  - We show that the iteration algorithm has cluster points as partial optima of the energy function.

The remainder of the paper is organized as follows. Section 3 gives some notations. In Sect. 4, we introduce our new Laplacian matrix. Section 5 presents the first stage of our segmentation method. In Sect. 6, we propose the second stage of our segmentation method, followed by iterative algorithms. In Sect. 7, we present our experiments and comparisons. Section 8 concludes the paper.

3 Notations

Assume $M \subset \mathbb{R}^3$ to be a compact triangulated surface of arbitrary topology with no degenerate triangles. M has V vertices, E edges, and T triangles, which are denoted as $\{v_i : i = 0, 1, \ldots, \mathrm{V}-1\}$, $\{e_i : i = 0, 1, \ldots, \mathrm{E}-1\}$ and $\{\tau_i : i = 0, 1, \ldots, \mathrm{T}-1\}$, respectively. If v is an endpoint of an edge e, we denote it as $v \prec e$. Similarly, that e is an edge of a triangle τ is denoted as $e \prec \tau$; that v is a vertex of a triangle τ is denoted as $v \prec \tau$. Let $N_I(\tau)$ be the first type 1-neighborhood of triangle τ, which are the triangles sharing some common edges with τ. Let $N_{II}(\tau)$ be the second type 1-neighborhood of triangle τ, which are the triangles sharing some common vertex with τ. For an edge $e \prec \tau$, if the orientation of e is consistent with the orientation of τ, then $\mathrm{sgn}(e, \tau) = 1$; otherwise $\mathrm{sgn}(e, \tau) = -1$.

We now introduce several basic spaces and operators, which will be used to formulate our mesh segmentation method. We first present the space $V_M = \mathbb{R}^{\mathrm{T}}$, which is isomorphic to the piecewise constant function space over M. An element $u \in V_M$ is $u = (u_0, u_1, \ldots, u_{\mathrm{T}-1}) \in V_M$. It means that the value of u restricted on the triangle τ is u_τ (written as $u|_\tau$ sometimes).

As in [49], for any $u^1, u^2, u \in V_M$, V_M is equipped with the following inner product and norm:

$$(u^1, u^2)_{V_M} = \sum_\tau u^1|_\tau u^2|_\tau s_\tau, \quad \|u\|_{V_M} = \sqrt{(u, u)_{V_M}}, \tag{1}$$

where s_τ is the area of triangle τ.

For $u \in V_M$, we have its gradient as follows:

$$\nabla : u \to \nabla u, \quad \nabla u|_e = [u]_e, \ \forall e,$$

where $[u]_e$ is a jump of u crossing e defined as follows:

$$[u]_e = \begin{cases} \sum\limits_{e \prec \tau} u|_\tau \mathrm{sgn}(e, \tau), & e \nsubseteq \partial M \\ 0, & e \subseteq \partial M \end{cases}. \tag{2}$$

The range of ∇ is denoted by Q_M, i.e., $Q_M = \text{Range}(\nabla)$.

For $p^1, p^2, p \in Q_M$, Q_M can also be equipped with the following inner product and norm [49]:

$$(p^1, p^2)_{Q_M} = \sum_e p^1|_e p^2|_e l_e, \quad \|p\|_{Q_M} = \sqrt{(p, p)_{Q_M}} \tag{3}$$

V_M and Q_M can be extended to vectorial case. For $\mathfrak{N}$-channel data, two spaces $\mathbf{V}_M, \mathbf{Q}_M$ are defined as follows:

$$\mathbf{V}_M = \underbrace{V_M \times \cdots \times V_M}_{\mathfrak{N}}, \quad \mathbf{Q}_M = \underbrace{Q_M \times \cdots \times Q_M}_{\mathfrak{N}}.$$

For $\mathbf{u}^1, \mathbf{u}^2, \mathbf{u} \in \mathbf{V}_M$, $\mathbf{p}^1, \mathbf{p}^2, \mathbf{p} \in \mathbf{Q}_M$, [49] gives the following inner products and norms:

$$(\mathbf{u}^1, \mathbf{u}^2)_{\mathbf{V}_M} = \sum_{1 \le i \le \mathfrak{N}} (u_i^1, u_i^2)_{V_M}, \quad \|\mathbf{u}\|_{\mathbf{V}_M} = \sqrt{(\mathbf{u}, \mathbf{u})_{\mathbf{V}_M}},$$
$$(\mathbf{p}^1, \mathbf{p}^2)_{\mathbf{Q}_M} = \sum_{1 \le i \le \mathfrak{N}} (p_i^1, p_i^2)_{Q_M}, \quad \|\mathbf{p}\|_{\mathbf{Q}_M} = \sqrt{(\mathbf{p}, \mathbf{p})_{\mathbf{Q}_M}}.$$

$\nabla \mathbf{u}$ is computed channel by channel. For $\mathbf{u} \in \mathbf{V}_M$, the vectorial total variation is:

$$R_{\text{vtv}}(\nabla \mathbf{u}) = (\text{TV})(\mathbf{u}) = \sum_e \sqrt{\sum_{i=1}^{\mathfrak{N}} |[u_i]_e|^2 l_e}. \tag{4}$$

We denote a segmentation of a given mesh M as the set of $\mathbf{K}+1$ sub mesh surfaces $M_1, M_2, \ldots, M_k, \ldots, M_{\mathbf{K}+1}$ with triangles as clustering elements, and the $\mathbf{K}+1$ parts satisfy $M_i \bigcap M_j = \varnothing, i \neq j$ (no common triangles between different parts) and $\bigcup_{i=1}^{\mathbf{K}+1} M_i = M$ (no missing triangles in the union).

4 The Laplacian matrix and feature function f of the mesh

The Laplacian matrix is an important tool in image and mesh segmentation. The eigenvalues and eigenvectors of a Laplacian matrix reveal global structures of the image or mesh data and can be used to describe the feature space of the data for segmentation. The interested reader can refer to, e.g., [28,38,51], and an excellent survey [48].

In part-type mesh segmentation, the most popular Laplacian matrices similar to that in [28,51] are not semantic enough for non-CAD surfaces and sensitive to mesh quality. They fail to segment objects with thin plates and slim strips into semantic components; see Fig. 2 for our tests. We try to in this section construct a new Laplacian to overcome these problems by using more triangle neighborhoods, which is based on the nonlocal theory [6,40,52]. A comparison between our new Laplacian matrix and the classical Laplacian matrix on segmentation is also provided.

4.1 The Laplacian matrix

In part-type mesh segmentation, a Laplacian matrix $L = [L_{ij}]$ is usually defined, for a given mesh, as follows:

$$L_{ij} = \begin{cases} -w_{ij}, & i \neq j \text{ and } \tau_i,\ \tau_j \text{ share an edge} \\ \sum_k w_{i,k}, & i = j \text{ and } k \in N_I(\tau_i) \\ 0, & \text{otherwise.} \end{cases} \tag{5}$$

with a positive weight w_{ij}. It is defined as follows:

$$w_{ij} = l_e \exp\left(-\frac{d(\tau_i, \tau_j)}{\overline{d}}\right),$$

where $d(\tau_i, \tau_j)$ is a distance function to measure the difference of τ_i and τ_j, $\overline{d}$ is the average of $d(\tau_i, \tau_j)$ over all edges. We now present the distance functions $d_1(\tau_i, \tau_j)$ applied in [28,51] and our newly defined distance functions $d_2(\tau_i, \tau_j)$. The two distance functions produce two Laplacian matrices L_{Old} and L_{New}, respectively.

For each triangle τ, we first filter its normal by

$$h(\tau) = \text{normalize}\left(\sum_{\substack{i \in N_{II}(\tau) \\ n_\tau \cdot n_i > 0.93}} n_i\right). \tag{6}$$

- Old Laplacian Matrix L_{Old}:
  $d_1(\tau_i, \tau_j) = \eta_1 \|n_{\tau_i} - n_{\tau_j}\|^2, d(\tau_i, \tau_j) = d_1(\tau_i, \tau_j)$.
- Our New Laplacian Matrix L_{New}:
  $d_2(\tau_i, \tau_j) = \eta_2 \|H(\tau_i) - H(\tau_j)\|, d(\tau_i, \tau_j) = d_2(\tau_i, \tau_j)$.
  where $H(\tau_i) = \text{normalize}(\sum_{\substack{k \in N_{II}(\tau_i) \\ h(\tau_k) \cdot h(\tau_i) > 0.93}} h(\tau_k))$.

The parameter $\eta_1 = 0.5$ for a concave edge and $\eta_1 = 0.1$ for a convex edge in L_{Old}. For L_{new}, we set $\eta_2 = 1.0$ for a concave edge and $\eta_2 = 0.1$ for a convex edge. 0.93 is an empirical value.

We mention that we have tested many other forms of Laplacian such as that with distance in L_1 norm of the difference of the normal vectors in our experiments. It turns out that the above L_{new} works best.

Remark 1 Our new Laplacian Matrix is based on the nonlocal theory in image denoising [6], which has been used for surface denoising [40,52]. That is to use similar information in the neighborhood to filter data. For (6), the larger the inn product between n_i and n_j, the higher similarity. If the threshold is too large, it needs to compute more neighborhood, which makes the computational costs high. If the

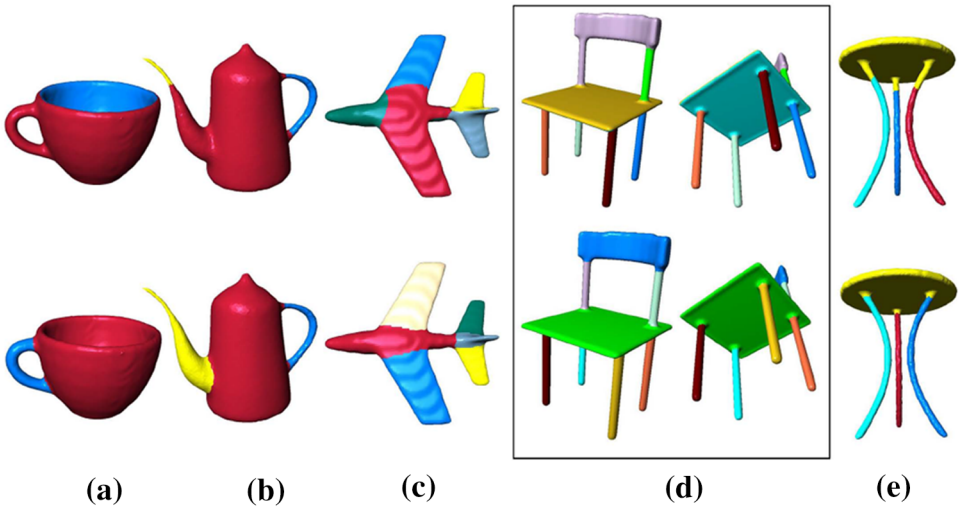


Fig. 2 Comparisons of the final segmentation results (after two stages) by using the eigenvectors of L_{Old} and L_{New}. The *first row* shows the results using L_{Old}. The *second row* shows the results using L_{New}

threshold is too small, the similarity between normals is low, which may destroy origin data normal. Many experiments show that 0.93 is suitable for most models.

4.2 Feature function f of the mesh M

A usual way to define feature functions of image and mesh data is using the eigenvectors of a Laplacian matrix [28,38,51]. For a given Laplacian matrix, we compute its $\mathbf{K}$ eigenvectors $[\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_\mathbf{K}]$ corresponding to the first $\mathbf{K}$ smallest nonzero eigenvalues. The feature function $\mathbf{f}$ is then defined as

$$\mathbf{f}_\tau = (\mathbf{v}_{1,\tau}, \mathbf{v}_{2,\tau}, \ldots, \mathbf{v}_{\mathbf{K},\tau}), \forall \tau. \tag{7}$$

The feature function $\mathbf{f}$ is used to segment the mesh by clustering mesh elements in the feature space. Different Laplacian matrices and the corresponding feature functions have quite different segmentation results.

In Fig. 2, we exhibit the comparisons of the segmentation results by using L_{Old} and L_{New} in our segmentation method. It can be seen that L_{Old} is severely sensitive to normal variation. For those meshes with thin plates and slim cylinder-like components, it fails to give geometry-aware segmentations. The thin plate and cylinder-like part are, respectively, cut into two parts; see the cup and chair meshes in the first row of Fig. 2. In contrast, our newly defined L_{New} does not have these problems.

5 The first stage: spectral clustering procedure

In this section, we introduce the first stage of our segmentation method, which is a simple and fast spectral clustering procedure based on the good properties (the distribution of maximum and minimum values of eigenvectors is an approximation of cluster centers) of the feature function $\mathbf{f}$ defined in Sect. 4. For clustering algorithms, to find good cluster centers is a key step, which affects clustering results severely. So far, in mesh segmentation, cluster centers are mainly based on user interaction, or geodesic distance, or randomly chosen; see [23,28,51] for details. According to our tests, the geodesic distance method in [23,28] is inaccurate and usually time-consuming, and the randomly chosen method [51] is unstable and difficult to get correct clustering results. Our spectral clustering procedure first proposes an automatic, simple, and effective method for computing the cluster centers $\boldsymbol{\mu}$ directly from the feature function $\mathbf{f}$. Then, the cluster labels $\mathbf{u} \in \mathbf{V}_M$ can be obtained by measuring the error between the feature function $\mathbf{f}$ and cluster centers $\boldsymbol{\mu}$. In the following, we present the details.

5.1 Computing the cluster centers μ

We now present our method to compute the cluster centers. Since to divide the mesh surface to $\mathbf{K}+1$ parts, the basic idea is to find $\mathbf{K}+1$ cluster centers as far away from each other as possible in the feature space. Our method is based on an observation: the maximum and minimum points (actually triangles) of the eigenvectors composing of feature function $\mathbf{f}$ are located far away from each other. They are approximately at the centers of semantic components; see, e.g., Fig. 3. These maximum and minimum points are good candidates of cluster centers. As we choose the first $\mathbf{K}$ eigenvectors to define the feature function, the maximum and minimum points of the $\mathbf{K}$ eigenvectors have $2\mathbf{K}$ candidate cluster centers, which correspond to $2\mathbf{K}$ candidate triangles. We need to construct $\mathbf{K}+1$ cluster centers using these $2\mathbf{K}$ candidates. After removing the duplicate triangles, we get m candidate triangles now. We observed that $\mathbf{K} \leq m \leq 2\mathbf{K}$. We now construct $\mathbf{K}+1$ triangles and corresponding cluster centers. If $m < \mathbf{K}+1$, we add new triangles until $m = \mathbf{K}+1$; if $m > \mathbf{K}+1$, we

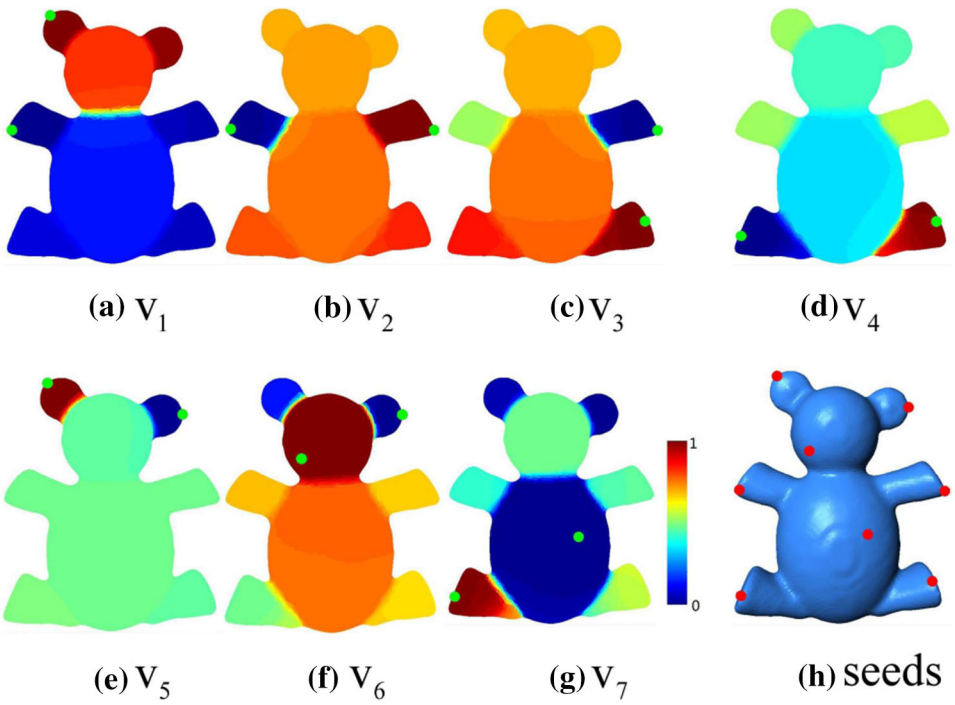


Fig. 3 Eigenvectors and their maximizers and minimizers (*green* balls) of the Laplacian L_{New} corresponding to the first seven nonzero eigenvalues. **h** The cluster centers found by our Algorithm 1, which represent the 8 geometry-aware components

remove some triangles until $m = \mathbf{K} + 1$. The details can be found in Algorithm 1.

In (a–g) of Fig. 3, we show the distribution of maximum and minimum values (the green balls) of the first 7 eigenvectors of the bear mesh, and (h) is the final distribution of 8 cluster centers of the bear. As can be seen, our method is quite effective. Figure 4 further demonstrates the effectiveness of our method by applying it to the Princeton Segmentation Benchmark [9]. Also, Algorithm 1 is efficient, since $\mathbf{K} + 1$ is usually very small (e.g., < 20). With these cluster centers as initializations (see Algorithm 2), our variational method is stable and effective. See Sect. 7.4 for a comparison on results with random initialization and our Algorithm 1.

Remark 2 In the experiments, we found that m happens to be $\mathbf{K} + 1$ for most meshes we tested.

Remark 3 We consulted several geometrists about the observations in our experiments. They think it hard to prove them rigorously. We pointed out that these observations are correct in all our tests including those on the Princeton Segmentation Benchmark [9].

Algorithm 1 Computing cluster centers μ

Input: $\mathbf{K}$ eigenvectors.
Output: $\mathbf{K} + 1$ cluster centers $\boldsymbol{\mu}$;

1. Find $2\mathbf{K}$ triangles with maximum and minimum values of $\mathbf{K}$ eigenvectors;
2. Get a set T_a of m triangles by removing duplicate triangles from the above $2\mathbf{K}$ triangles;
3. **Case 1:If $m < \mathbf{K} + 1$, we add triangles to T_a**
   Repeat
   Choose the triangle $\tau \in M \setminus T_a$, which maximizes $\min_{\tau_a \in T_a} \|\mathbf{f}_\tau - \mathbf{f}_{\tau_a}\|^2$;
   Add τ to T_a;
   Until$(m == \mathbf{K} + 1)$
   Case 2:If $m > \mathbf{K} + 1$, we remove triangles from T_a
   Repeat
   Choose two closest triangles $\tau_1 \in T_a, \tau_2 \in T_a$ in the feature space, i.e.,

$$\min_{\tau_1, \tau_2 \in T_a, \tau_1 \neq \tau_2} \|\mathbf{f}_{\tau_1} - \mathbf{f}_{\tau_2}\|^2$$

   For τ_1, τ_2, compute the following distances:

$$D_{\tau_1} = \sum_{\tau \in T_a} \|\mathbf{f}_{\tau_1} - \mathbf{f}_\tau\|^2; \; D_{\tau_2} = \sum_{\tau \in T_a} \|\mathbf{f}_{\tau_2} - \mathbf{f}_\tau\|^2;$$

   Delete the triangle τ_i with low $D_{\tau_i}, i = 1$ or 2;
   Until$(m == \mathbf{K} + 1)$

5.2 Spectral clustering procedure

By the cluster centers $\boldsymbol{\mu}$ and the feature function $\mathbf{f}$, we can compute the cluster labels for triangles, which is exactly our spectral clustering procedure. Keep in mind that M is to be partitioned into $\mathbf{K} + 1$ disjoint parts with triangles as clustering elements. For each triangle τ, we obtain its label $\mathbf{u}_\tau = (u_{\tau,1}, \ldots, u_{\tau,i}, \ldots, u_{\tau,\mathbf{K}+1})$ by measuring the similarity between its feature value $\mathbf{f}_\tau$ and cluster centers μ. That is,

$$u_{\tau,i} = \delta_{ij}, \; j = \arg\min_l \{\|\mathbf{f}_\tau - \mu_l\|^2, l = 1, \ldots, \mathbf{K} + 1\}. \quad (8)$$

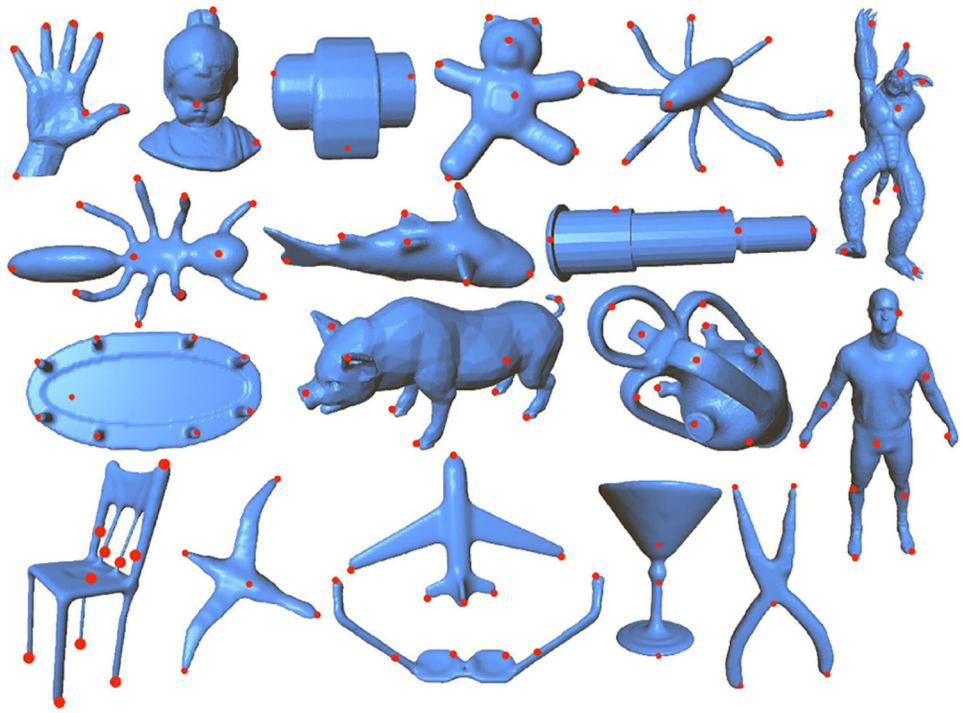


Fig. 4 Cluster centers produced by Algorithm 1 applied to the representatives of 19 categories in Princeton Segmentation Benchmark [9]

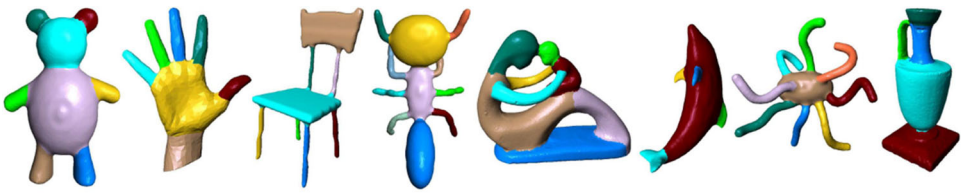


Fig. 5 Segmentation results of the first stage of our method

Our spectral clustering procedure is simple, fast and can produce satisfying segmentation results for most mesh surfaces in our experiments. Figure 5 shows several segmentation results produced by formula (8). As can be seen, the segmentation results in Fig. 5 respect human perception and the segmentation boundaries are promising. Nevertheless, there are still some surfaces for which our spectral clustering procedure does not work well; see the first row of Fig. 6. In order to improve the segmentation results, we introduce our second stage in Sect. 6, which is a variational refining procedure. The second row of Fig. 6 shows the segmentation results after the variational refining procedure.

6 The second stage: variational refining procedure

With the good initializations provided by the first stage, in this section, we devise an effective variational refining segmentation method by a new discretization of the classical non-convex Mumford–Shah model. Besides, we introduce an efficient iterative algorithm for solving the variational problem. Some convergent analysis will also be given.

6.1 The Mumford–Shah (MS) segmentation on manifold

The Mumford–Shah model [34] is a classical image approximation model, which has been successfully applied for image segmentation [7,8,26,27,39] and piecewise linear space-based surface segmentation [11,22,43,51]. In this subsection, we give a brief introduction to it.

Assume f is a feature function defined on the manifold surface $\mathcal{M}$. We wish to find a piecewise smooth approximation of f on $\mathcal{M}$ by piecewise smooth function $\mathbf{g}$, i.e., to partition $\mathcal{M}$ into $\mathbf{K}+1$ disjoint parts $\mathcal{M}_1, \mathcal{M}_2, \ldots, \mathcal{M}_k, \ldots, \mathcal{M}_{\mathbf{K}+1}$, on each of which g is smooth. Γ is the union of their boundaries. The MS model tries to minimize the following functional:

$$\min_{\mathbf{g},\Gamma} |\Gamma| + \alpha \int_{\mathcal{M}} (f-g)^2 \mathrm{d}x + \beta \int_{\mathcal{M}\setminus\Gamma} |\nabla \mathbf{g}|^2 \mathrm{d}x, \qquad (9)$$

where α, μ are positive parameters and $|\Gamma|$ is the length of Γ.

When $\mathbf{g}$ is a piecewise constant function: i.e., $\mathbf{g}|_{\mathcal{M}_k} = \mu_k$, the above formula (9) degenerates into the following piecewise constant MS model:

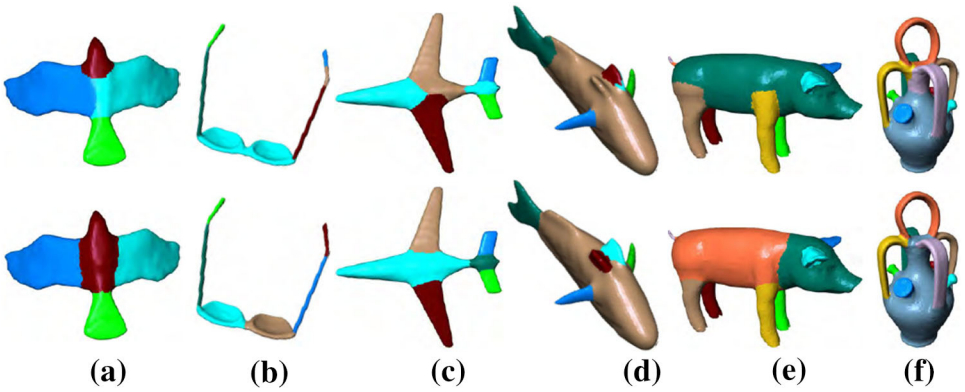


Fig. 6 Comparisons between the segmentation results before (the *first row*) and after (the *second row*) the variational refining procedure. For these mesh surfaces, the variational refining procedure improves greatly the segmentation results of the first stage

$$\min_{\mu,\Gamma} |\Gamma| + \alpha \sum_k \int_{\mathcal{M}_k} (f - \mu_k)^2 \mathrm{d}x. \tag{10}$$

However, it is difficult to minimize the above problem because of the unknown set Γ. For better solving the above problem, using the techniques in [25,26], problem (10) could be converted into the following MS model by introducing multiclass label $\mathbf{u}$:

$$\inf_{\mu_k,\mathbf{u}} \int_{\mathcal{M}} |\nabla \mathbf{u}| \mathrm{d}\mathcal{M} + \alpha \sum_k \int_{\mathcal{M}_k} (f - \mu_k)^2 u_k \mathrm{d}\mathcal{M}, \tag{11}$$

where $\mathbf{u} = (u_1, u_2, \ldots, u_k, \ldots, u_{\mathbf{K}+1})$ is a $(\mathbf{K}+1)$-dimensional label function satisfying some simplex constraints, and $|\Gamma| = \int_{\mathcal{M}} |\nabla \mathbf{u}|$ indicates that the length of the boundary Γ equals the total variation of $\nabla \mathbf{u}$ (see [54] for more explanations about the total variation). When $\mu_k, k = 1, \ldots, \mathbf{K}+1$ are known, (11) is convex; otherwise, it is still non-convex.

In this paper we will apply (11) to devise our variational refining procedure. We provide a good initialization to the non-convex problem by spectral analysis. Since it is hard to find global minimum of non-convex problems, a good initialization (near by the desired solution) is a good way. Experiments showed that our strategy works pretty well.

As the above Mumford–Shah model and its relaxation (11) are minimization models in continuous setting, in applications, we need to appropriately discretize it. Existing discretization methods on triangular meshes are based on piecewise linear finite element, which cluster vertices of the triangle mesh. They produce undetermined regions on the mesh (see the red regions in Fig. 12a) and a post-processing step is needed to get a triangle-based segmentation. Unlike existing methods, we propose in this paper to discretize the Mumford–Shah model in piecewise constant function space to segment the mesh surface, which clusters triangles directly. It will not produce undetermined regions and is more appropriate for region-based mesh segmentation.

6.2 A triangle-based variational mesh segmentation model

Assume a mesh M to be partitioned into $\mathbf{K}+1$ disjoint parts with triangles as clustering elements. By discretizing (11) in piecewise constant function space, our variational refining mesh segmentation model reads:

$$\min_{\{\mathbf{u}\in C_{\mathbf{u}},\mu\}} E(\mathbf{u}, \boldsymbol{\mu}) = R_{\mathrm{vtv}}(\nabla \mathbf{u}) + \alpha \langle \mathbf{u}, \mathbf{s}(\mathbf{f}, \boldsymbol{\mu}) \rangle_{\mathbf{V}_M}, \tag{12}$$

where $C_{\mathbf{u}} = \{\mathbf{u}_\tau = (u_{1,\tau}, \ldots, u_{k,\tau}, \ldots, u_{\mathbf{K}+1,\tau}) : u_{k,\tau} \geq 0, \sum_{k=1}^{\mathbf{K}+1} u_{k,\tau} = 1, \quad \forall \tau\} \subset \mathbf{V}_M$; $\boldsymbol{\mu} = (\mu_1, \mu_2, \ldots, \mu_k, \ldots, \mu_{\mathbf{K}+1})$ is the mean feature values corresponding to the segmentation parts $(M_1, M_2, \ldots, M_{\mathbf{K}+1})$ with each component μ_k as a $\mathbf{K}$-dimensional vector; $\mathbf{s}_\tau(\mathbf{f}, \boldsymbol{\mu}) = \mathbf{s}(\mathbf{f}_\tau, \boldsymbol{\mu}) = (s(\mathbf{f}_\tau, \mu_1), \ldots, s(\mathbf{f}_\tau, \mu_2), \ldots, s(\mathbf{f}_\tau, \mu_{\mathbf{K}+1}))$ with $s(\mathbf{f}_\tau, \mu_k) = (\mathbf{f}_\tau - \boldsymbol{\mu}_k, \mathbf{f}_\tau - \boldsymbol{\mu}_k)$ indicating the error between the feature $\mathbf{f}_\tau$ of triangle τ and the mean feature $\boldsymbol{\mu}_k$ of part M_k; $\mathbf{f}_\tau$ is the feature of triangle τ; and α is a positive parameter to balance the two terms.

The $\mathbf{u}$ in the above model is a classification function for the segmentation, which will by converted as part label by the method in [26]. We mention that previous segmentations models in [11,43,51] use piecewise linear function spaces to extend the Mumford–Shah model for clustering vertices. Here our discretization is for clustering triangles. Since triangles on a mesh are usually nonuniformly distributed, triangle clustering is more ready for region-based segmentation.

6.3 An alternating minimization method (AMM) for solving (12)

By introducing the following function:

$$\chi(\mathbf{u}) = \begin{cases} 0, & \mathbf{u} \in C_{\mathbf{u}} \\ +\infty, & \mathbf{u} \notin C_{\mathbf{u}}, \end{cases} \tag{13}$$

our minimization problem (12) is reformulated as

$$\min_{\{\mathbf{u},\mu\}} E_C(\mathbf{u},\boldsymbol{\mu}) = R_{\text{vtv}}(\nabla\mathbf{u}) + \alpha(\mathbf{u},\mathbf{s}(\mathbf{f},\boldsymbol{\mu}))_{\mathbf{V}_M} + \chi(\mathbf{u}). \tag{14}$$

Since $\mathbf{u}$ and $\boldsymbol{\mu}$ in (14) are coupled together, they are hard to solve simultaneously. We thus apply the following alternating minimization to solve them:

– For fixed $\boldsymbol{\mu}$: $\mathbf{u}$ can be obtained by

$$\min_{\mathbf{u}} E_C(\mathbf{u},\boldsymbol{\mu}) = R_{\text{vtv}}(\nabla\mathbf{u}) + \alpha(\mathbf{u},\mathbf{s}(\mathbf{f},\boldsymbol{\mu}))_{\mathbf{V}_M} + \chi(\mathbf{u}), \tag{15}$$

which is non-differentiable, and can be efficiently solved by augmented Lagrangian and ADMM [5,35,41,47]; see Sect. 6.4 for details.

– For fixed $\mathbf{u}$: $\boldsymbol{\mu}$ can be obtained by

$$\min_{\mu} E_C(\mathbf{u},\boldsymbol{\mu}) = R_{\text{vtv}}(\nabla\mathbf{u}) + \alpha(\mathbf{u},\mathbf{s}(\mathbf{f},\boldsymbol{\mu}))_{\mathbf{V}_M} + \chi(\mathbf{u}),$$

and is exactly

$$\boldsymbol{\mu}_k = \frac{\sum_\tau u_{k,\tau}\mathbf{f}_\tau s_\tau}{\sum_\tau u_{k,\tau} s_\tau}, \quad k = 1, 2, \ldots, \mathbf{K}+1. \tag{16}$$

For this alternating minimization, we have

Theorem 1 *The sequence* $\{(\mathbf{u}^n,\boldsymbol{\mu}^n)\}$ *generated by the alternating minimization has cluster points. Any cluster point is a partial optimum of* $E_C(\mathbf{u},\mu) = R_{\text{vtv}}(\nabla\mathbf{u}) + \alpha(\mathbf{u},\mathbf{s}(\mathbf{f},\boldsymbol{\mu}))_{\mathbf{V}_M} + \chi(\mathbf{u})$.

Proof The alternating minimization is

$$\begin{aligned}\mathbf{u}^{n+1} &= \arg\min_{\mathbf{u}} E_C(\mathbf{u},\mu^n),\\ \boldsymbol{\mu}^{n+1} &= \arg\min_{\boldsymbol{\mu}} E_C(\mathbf{u}^{n+1},\mu),\end{aligned} \tag{17}$$

for $n = 0, 1, \ldots$. This gives

$$\begin{aligned}E_C(\mathbf{u}^{n+1},\mu^n) &\le E_C(\mathbf{u}^n,\mu^n),\\ E_C(\mathbf{u}^{n+1},\mu^{n+1}) &\le E_C(\mathbf{u}^{n+1},\mu^n),\end{aligned} \tag{18}$$

which indicates

$$E_C(\mathbf{u}^{n+1},\mu^{n+1}) \le E_C(\mathbf{u}^n,\mu^n).$$

Since $\{E_C(\mathbf{u}^n,\mu^n)\}$ is bounded from below, the sequence $\{E_C(\mathbf{u}^n,\mu^n)\}$ is convergent, and we denote the limit as $E^\star$. From (18), we also have $\lim_{n\to+\infty} E_C(\mathbf{u}^{n+1},\mu^n) = E^\star$.

As $\mathbf{u}^n \in C_{\mathbf{u}}$, $\forall n$, $\{\mathbf{u}^n\}_{n=0}^{+\infty}$ and $\{\mu^n\}_{n=0}^{+\infty}$ are bounded. Therefore, $\{(\mathbf{u}^n,\mu^n)\}_{n=0}^{+\infty}$ has convergent subsequences, and hence has cluster points.

Assume $\{(\mathbf{u}^{n_j},\mu^{n_j})\}$ is a convergent subsequence of $\{(\mathbf{u}^n,\mu^n)\}$ and $\lim_{j\to+\infty}(\mathbf{u}^{n_j},\mu^{n_j}) = (\overline{\mathbf{u}},\overline{\mu})$. It is clear that $E(\overline{\mathbf{u}},\overline{\mu}) = E^\star$. In the following, we show that $(\overline{\mathbf{u}},\overline{\mu})$ is a partial optimum of $E(\mathbf{u},\mu)$.

From the optimality of μ^{n_j}, we have

$$E_C(\mathbf{u}^{n_j},\mu^{n_j}) \le E_C(\mathbf{u}^{n_j},\mu),\ \forall\mu.$$

Letting $j\to+\infty$, it follows that

$$E_C(\overline{\mathbf{u}},\overline{\mu}) \le E_C(\overline{\mathbf{u}},\mu),\ \forall\mu.$$

From the optimality of $\mathbf{u}^{n_j+1}$, we have

$$E_C(\mathbf{u}^{n_j+1},\mu^{n_j}) \le E_C(\mathbf{u},\mu^{n_j}),\ \forall\mathbf{u}.$$

It follows that

$$\begin{aligned}E^\star &= \lim_{j\to+\infty} E_C(\mathbf{u}^{n_j+1},\mu^{n_j}) \le \lim_{j\to+\infty} E_C(\mathbf{u},\mu^{n_j})\\ &= E_C(\mathbf{u},\overline{\mu}),\ \forall\mathbf{u},\end{aligned}$$

which is exactly

$$E_C(\overline{\mathbf{u}},\overline{\mu}) \le E_C(\mathbf{u},\overline{\mu}),\ \forall\mathbf{u}.$$

Thus $(\overline{\mathbf{u}},\overline{\mu})$ is a partial optimum of $E_C(\mathbf{u},\mu)$. □

In our implementation, the alternating minimization is listed in Algorithm 2.

Algorithm 2 AMM for mesh segmentation

1. **Initialization**:
   - 1.1 μ^0 : computed by Algorithm 1;
   - 1.2 $\mathbf{u}^0$: computed by (8); $\mathbf{p}^0 = 0$; $\lambda_{\mathbf{p}}^0 = 0$; $\lambda_{\mathbf{z}}^0 = 0$;
2. **Repeat**
   - 2.1 For fixed $\boldsymbol{\mu}^n$, computing $\mathbf{u}^{n+1}$ by minimizing (15) through Algorithm 3, which gives simutaneously $(\mathbf{u}^{n+1},\mathbf{p}^{n+1},\mathbf{z}^{n+1})$.
   - 2.2 For fixed $\mathbf{u}^{n+1}$, computing $\boldsymbol{\mu}^{n+1}$ by (16);

   Until$(\|\mathbf{u}^{n+1}-\mathbf{u}^n\|_{\mathbf{V}_M}^2 < 10^{-5}$ or $40 \le n \le 100)$.
3. **Classify u** by method in [26].

6.4 ADMM for solving (15)

The problem (15) can be further written as

$$\begin{aligned}\min_{\{\mathbf{u}\in\mathbf{V}_M,\mathbf{p}\in\mathbf{Q}_M,\mathbf{z}\in\mathbf{V}_M\}} & R_{\text{vtv}}(\mathbf{p}) + \alpha(\mathbf{z},\mathbf{s}(\mathbf{f},\boldsymbol{\mu}))_{\mathbf{V}_M} + \chi(\mathbf{z}),\\ \text{s.t. } & \mathbf{p} = \nabla\mathbf{u}, \quad \mathbf{z} = \mathbf{u},\end{aligned} \tag{19}$$

by introducing two auxiliary variables $\mathbf{p} \in \mathbf{Q}_M$ and $\mathbf{z} \in \mathbf{V}_M$.

Equation (19) is an equality constrained problem of (15) and can be solved by augmented Lagrangian or ADMM, which is based on the following augmented Lagrangian functional:

$$\begin{aligned}\mathcal{L}_{\text{vtv}}(\mathbf{u}, \mathbf{p}, \mathbf{z}; \lambda_{\mathbf{p}}, \lambda_{\mathbf{z}}, \boldsymbol{\mu}) = {} & R_{\text{vtv}}(\mathbf{p}) + \alpha(\mathbf{z}, \mathbf{s}(\mathbf{f}, \boldsymbol{\mu}))_{\mathbf{V}_M} + \chi(\mathbf{z}) \\ & + (\lambda_{\mathbf{p}}, \mathbf{p} - \nabla\mathbf{u})_{\mathbf{Q}_M} \\ & + (\lambda_{\mathbf{z}}, \mathbf{z} - \mathbf{u})_{\mathbf{V}_M} \\ & + \frac{r_{\mathbf{p}}}{2}\|\mathbf{p} - \nabla\mathbf{u}\|^2_{\mathbf{Q}_M} \\ & + \frac{r_{\mathbf{z}}}{2}\|\mathbf{z} - \mathbf{u}\|^2_{\mathbf{V}_M}, \end{aligned} \tag{20}$$

where $r_{\mathbf{p}}$ and $r_{\mathbf{p}}$ are positive constants.

The ADMM is an iterative method. In each iteration, we need to solve a minimization problem for fixed multipliers $(\lambda_{\mathbf{p}}, \lambda_{\mathbf{z}})$, which can be separated into three subproblems:

– The $\mathbf{u}$-subproblem: for given $\mathbf{p}$, $\mathbf{z}$

$$\begin{aligned}\min_{\mathbf{u}\in\mathbf{V}_M} & \left(\lambda_{\mathbf{p}}, -\nabla\mathbf{u}\right)_{\mathbf{Q}_M} + (\lambda_{\mathbf{z}}, -\mathbf{u})_{\mathbf{V}_M} \\ & + \frac{r_{\mathbf{p}}}{2}\|\mathbf{p} - \nabla\mathbf{u}\|^2_{\mathbf{Q}_M} + \frac{r_{\mathbf{z}}}{2}\|\mathbf{z} - \mathbf{u}\|^2_{\mathbf{V}_M}. \end{aligned} \tag{21}$$

– The $\mathbf{p}$-subproblem: for given $\mathbf{u}$, $\mathbf{z}$,

$$\sum_e |(\mathbf{p})_e| l_e + (\lambda_{\mathbf{p}}, \mathbf{p})_{\mathbf{Q}_M} + \frac{r_{\mathbf{p}}}{2}\|\mathbf{p} - \nabla\mathbf{u}\|^2_{\mathbf{Q}_M}. \tag{22}$$

– The $\mathbf{z}$-subproblem: for given $\mathbf{u}$, $\mathbf{p}$,

$$\alpha(\mathbf{z}, \mathbf{s}(\mathbf{f}, \boldsymbol{\mu}))_{\mathbf{V}_M} + \chi(\mathbf{z}) + (\lambda_{\mathbf{z}}, \mathbf{z}) + \frac{r_{\mathbf{z}}}{2}\|\mathbf{z} - \mathbf{u}\|^2_{\mathbf{V}_M}. \tag{23}$$

The $\mathbf{u}$-subproblem (21) is a quadratic programming. Its solution can be achieved by a sparse linear system.

The $\mathbf{p}$-subproblem (22) has the following closed form solution

$$\forall e, \mathbf{p}_e = \begin{cases} \left(1 - \frac{1}{r_{\mathbf{p}}|\mathbf{w}_e|}\right)\mathbf{w}_e, & |\mathbf{w}_e| > \frac{1}{r_{\mathbf{p}}}, \\ 0, & |\mathbf{w}_e| \le \frac{1}{r_{\mathbf{p}}}, \end{cases} \tag{24}$$

where $\mathbf{w}_e = (\nabla\mathbf{u})_e - \frac{\lambda_{\mathbf{p}}|_e}{r_{\mathbf{p}}}$.

The $\mathbf{z}$-subproblem (23) can be solved through

$$\mathbf{z} = \text{Proj}_{C_{\mathbf{u}}}\left(\mathbf{u} - \frac{\alpha\mathbf{s} + \lambda_{\mathbf{z}}}{r_{\mathbf{z}}}\right), \tag{25}$$

which can be calculated via Michelot's algorithm [33].

This algorithm for (15) is listed in Algorithm 3. Since the objective function in (15) is convex, proper, coercive, and lower semi-continuous, standard convergence results as in [41] hold for Algorithm 3.

Algorithm 3 Solving the minimization problem (15)

1. **Initialization**:
   1.1 $\mathbf{u}^{n+1,0} = \mathbf{u}^n$, $\mathbf{z}^{n+1,0} = \mathbf{z}^n$, $\mathbf{p}^{n+1,0} = \mathbf{p}^n$;
   1.2 $\lambda_{\mathbf{p}}^{n+1,0} = \lambda_{\mathbf{p}}^n$, $\lambda_{\mathbf{z}}^{n+1,0} = \lambda_{\mathbf{z}}^n$.
2. **Repeat**
   2.1 $\mathbf{z}$-subproblem:
   For fixed $(\lambda_{\mathbf{z}}^{n+1,l}, \mathbf{u}^{n+1,l})$, compute $\mathbf{z}^{n+1,l+1}$ by (25);
   2.2 $\mathbf{u}$-subproblem:
   For fixed $(\lambda_{\mathbf{p}}^{n+1,l}, \lambda_{\mathbf{z}}^{n+1,l}, \mathbf{p}^{n+1,l}, \mathbf{z}^{n+1,l+1})$, compute $\mathbf{u}^{n+1,l+1}$ by (21);
   2.3 $\mathbf{p}$-subproblem:
   For fixed $(\lambda_{\mathbf{p}}^{n+1,l}, \mathbf{u}^{n+1,l+1})$, compute $\mathbf{p}^{n+1,l+1}$ by (24);
   2.4 Update Lagrange multipliers;
   $\lambda_{\mathbf{p}}^{n+1,l+1} = \lambda_{\mathbf{p}}^{n+1,l} + r_{\mathbf{p}}(\mathbf{p}^{n+1,l+1} - \nabla\mathbf{u}^{n+1,l+1})$;
   $\lambda_{\mathbf{z}}^{n+1,l+1} = \lambda_{\mathbf{z}}^{n+1,l} + r_{\mathbf{z}}(\mathbf{u}^{n+1,l+1} - \mathbf{z}^{n+1,l+1})$;
   Until$(\|\mathbf{u}^{n+1,l+1} - \mathbf{u}^{n+1,l}\|^2_{\mathbf{V}_M} < 10^{-3})$.
3. $\mathbf{u}^{n+1} = \mathbf{u}^{n+1,l}$, $\mathbf{z}^{n+1} = \mathbf{z}^{n+1,l}$, $\mathbf{p}^{n+1} = \mathbf{p}^{n+1,l}$.

7 Experimental results and discussions

In this section, we present experiments of our segmentation method on a wide variety of mesh surfaces, including the whole set of surfaces in Princeton Segmentation Benchmark and other surfaces. We implemented our segmentation method and the method in [51] by Microsoft Visual Studio 2010. All the examples were tested on a laptop with Intel Corei3 and 4GB RAM, and all mesh surfaces were rendered using flat shading. We will discuss our method from various aspects, such as choices of parameters, and comparisons to previous techniques by visual effects and quantitative errors.

7.1 Parameters and the number of segments

There are three parameters in our algorithm: α, $r_{\mathbf{p}}$ and $r_{\mathbf{z}}$. α is a variational model parameter; $r_{\mathbf{p}}$, $r_{\mathbf{z}}$ are two optimization algorithmic parameters. In the algorithm, the parameters $r_{\mathbf{p}}$ and $r_{\mathbf{z}}$ are fixed by $r_{\mathbf{p}} = 0.1$; $r_{\mathbf{z}} = 1$. The parameter α affects the segmentation boundary. The larger α is, the poorer the segmentation boundary is. As the number of segments, the size of meshes, the similarity term and the regularization term all affect α, it is not reasonable to fix α as a constant. According to our experiments, we provide an empirical formula, which reads:

$$\alpha = 0.06 \times \text{V} \times (\mathbf{K} + 1) \times \frac{(\text{TV})(\mathbf{f})}{(\mathbf{r}, \mathbf{s}(\mathbf{f}, \boldsymbol{\mu}))_{\mathbf{V}_M}}, \tag{26}$$

where $\mathbf{f}$ is the feature function; V is vertex number and $\mathbf{r}$ is a random element in $C_{\mathbf{u}}$. This empirical formula works well for many situations.

The number of segments $\mathbf{K} + 1$ depends on the geometry and semantic components of the mesh surface. It is quite

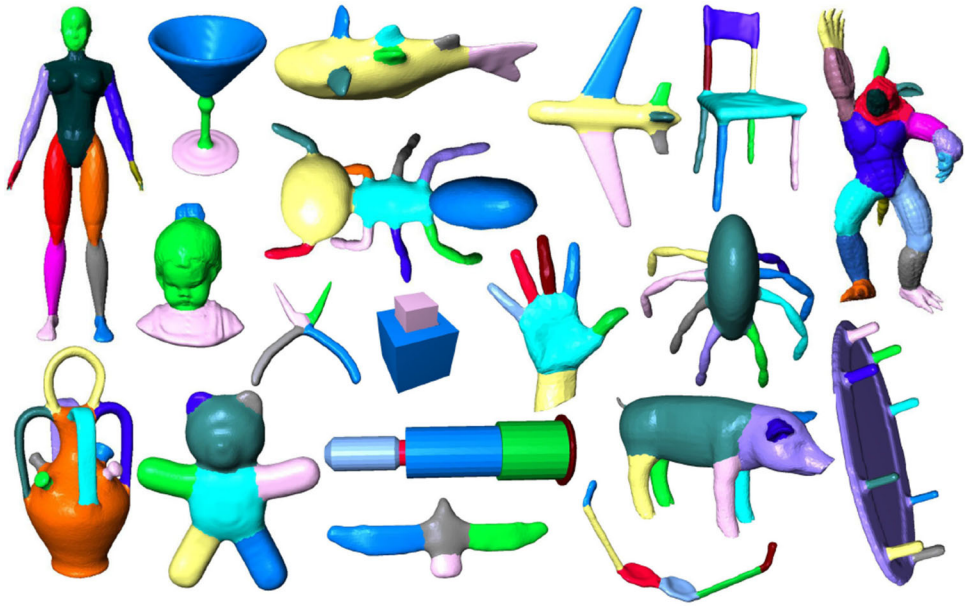


Fig. 7 Our segmentation results of representatives of 19 categories in Princeton Segmentation Benchmark [9]

difficult to find an empirical formula suitable for all mesh surfaces. Possible choices are the eigengap heuristic proposed in [31] and the ratio cut heuristic in [51]. However, these two methods can get correct segment number for only less than 40% mesh surfaces in the Benchmark. In the paper, for a given mesh, we input the number of segments manually, with the number of segments by the eigengap heuristic [31] as a reference.

Remark 4 As stated before, it is quite difficult to find an empirical formula to compute **K** suitable for all mesh surfaces. **K** is set manually in the whole paper. For setting **K**, the eigengap heuristic proposed in [31], the ratio cut heuristic in [51] and the segmentation numbers in [9] are referred.

7.2 Overall performance of our method and comparisons to existing methods

We tested our segmentation results on the Princeton Segmentation Benchmark [9], which contains 19 categories of 380 meshes, human segmentation results, and code for evaluating segmentation methods. Since our method does not involve user interaction, we denote our segmentation results by "Our Auto". Figure 7 shows the results by our method for representative meshes of the 19 categories in the Benchmark. As can be seen, our results follow the semantics of human perception.

With this benchmark, we also compare our approach both quantitatively and visually to several typical existing methods, including Rand Cuts [14], Shape Diam [37], Norm Cuts [14], Core Extra [20], Random Walks [23], Fit Prim [2], Isoline Cut [3], WCSeg [17] and M-S [51]. Their segmentation results are provided by Princeton Segmentation Benchmark [9] or the authors (For the method in [51], the authors cannot provide the segmentation results. We can only find their RandIndex data metrics). For evaluating segmentation results, many evaluation metrics have been proposed [4, 9, 29]. Chen et al. [9] proposed four evaluation metrics: Cut Discrepancy ("CD"), Hamming Distance-missing rate/false alarm rate ("Hamming-Rm/Rf"), Rand Index("RI") and Global/Local Consistency Error ("G/LCE"). Liu et al. [29] introduces two new evaluation metrics: Similarity Hamming Distance ("SHD") and Adaptive Entropy Increment ("AEI") to further improve the accuracy. The interested readers can also refer survey paper [4]. In this paper, we compare segmentation results according to the four evaluation metrics in [9].

Figure 9 shows the quantitative comparisons between our method and other methods according to the four error metrics. We can see that our results have lower errors than other methods. More details on the Rand Index metric are shown in Table 1, where the Rand Index scores of all the compared methods for the 19 categories of meshes in Princeton Segmentation Benchmark are listed. We can see that overall our results are better than the other methods. Figure 8 presents the comparison on segmentation results between our method and other six methods, which further demonstrates the priority of our segmentation method.

In addition to the benchmark data set, in Fig. 10, we tested our algorithm on more meshes and CAD meshes (the last row of Fig. 10). We can see that our method can produce satisfying results.

Table 1 Rand Index scores (the smaller the value, the better the segmentation results) across all 19 categories using our method and previous methods based on the Princeton Segmentation Benchmark of [9]

Categories	Human	Rand Cuts	Shape Diam	Norm Cuts	Rand Walks	Core Extra	Fit Prim	Isoline Cut	WCSeg	M-S	Our Auto
Human	13.5	13.1	17.9	15.2	21.9	22.5	15.3	12.3	12.8	**11.1**	12.9
Cup	13.6	21.9	35.8	24.4	35.8	30.7	41.3	21.	17.1	20.4	**10.6**
Glasses	10.1	10.11	20.4	14.1	31.1	30.1	23.5	9.8	17.3	**9.4**	9.5
Airplane	9.2	12.18	9.2	18.5	24.8	25.6	16.5	12.7	**8.9**	11.1	10.0
Ant	3.0	2.5	2.2	4.7	6.8	6.5	8.6	3.9	**2.1**	2.2	**2.1**
Chair	8.9	18.3	11.1	8.8	15.6	18.7	21.1	12.1	10.3	10.9	**6.8**
Octopus	**2.4**	6.3	4.5	6.1	6.7	5.1	10.1	4.1	2.9	2.5	**2.4**
Table	9.3	38.2	18.4	9.3	13.1	24.4	18.1	6.5	9.1	10.3	**6.6**
Teddy	4.9	4.5	5.7	12.1	12.7	11.4	13.2	5.3	5.6	**3.2**	3.6
Hand	9.1	8.9	20.2	15.5	18.9	15.5	20.2	11.5	11.6	7.9	**7.8**
Plier	**7.1**	10.9	37.5	18.3	23.0	9.3	16.9	7.3	8.7	8.9	8.7
Fish	**15.5**	29.6	24.8	39.4	38.8	27.3	42.4	24.3	20.3	29.6	21.8
Bird	**6.2**	10.7	11.5	18.4	24.9	12.4	19.6	9.7	10.1	9.4	7.3
Armadillo	8.3	9.2	9.0	11.5	11.5	14.1	9.1	10.6	**8.1**	8.7	9.5
Bust	**22.0**	23.2	29.8	31.6	29.8	31.5	29.9	24.4	26.5	25.1	24.8
Mech	13.1	27.7	23.8	15.9	21.1	38.7	30.5	12.2	18.2	13.1	**9.2**
Bearing	10.4	12.3	11.9	18.3	24.6	39.8	18.8	17.7	11.9	16.6	**9.2**
Vase	14.4	13.3	23.9	25.6	24.5	22.6	25.6	16.8	16.1	12.5	**11.9**
FourLeg	14.9	17.4	16.1	20.7	21.8	19.1	18.5	18.1	15.2	**14.4**	15.6
Average	**10.3**	15.3	17.6	17.2	21.5	21.1	21.0	12.7	12.3	12.0	11.0

Best results are highlighted in bold

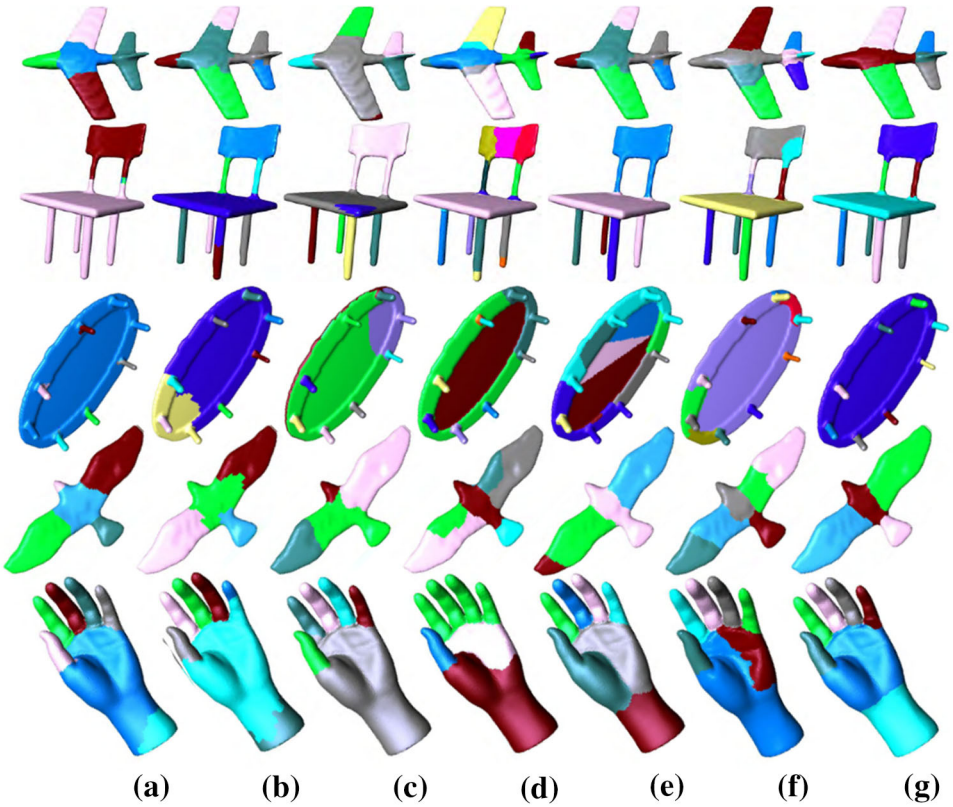


Fig. 8 Comparison results between our segmentation method and other segmentation methods. **a** Core Extra [20]; **b** Norm Cuts[14]; **c** Random Walks [23]; **d** Shape Diam [37]; **e** Rand Cuts [14]; **f** WCSeg [17]; **g** Our Auto

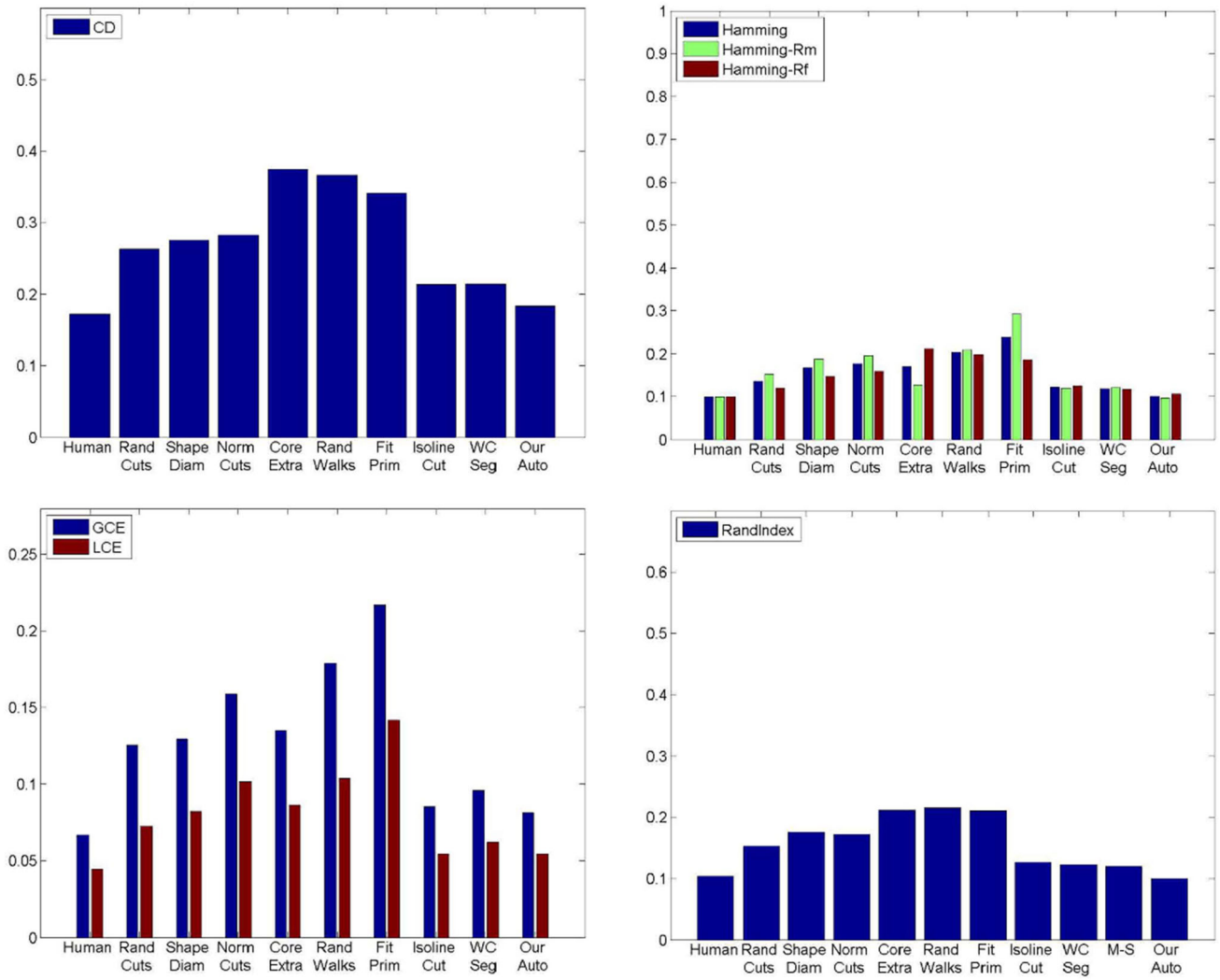


Fig. 9 Evaluation of segmentation in terms of four evaluation metrics. The evaluations are performed according to the protocols and human segmentations in the Princeton Segmentation Benchmark of [9]

7.3 Our method versus learning-based segmentation method in [18,19]

As most above methods belong to variational segmentation method, this section, we further compare our method with learning-based segmentation methods in [18,19] (The segmentation results are provided by authors. As triangle sequences in [18] are different with that in [9], for fairness, we re-adjust their results by simple algorithms. The segmentation results in [19] with 6 shapes, 12 shapes, 19 shapes as training are denoted by "SB6 2010","SB12 2010","SB19 2010," respectively. The segmentation results in [18] with 12 shapes as training are denoted by "PCN12 2017". In addition, the results of "SB6 2010" and "SB12 2010" are from [19]).

Figure 11 presents the quantitative comparisons between our method and methods in [18,19] according to the four error metrics and the Princeton Segmentation Benchmark in [9]. In fact, the learning-based segmentation results, especially with more shapes as training, are better than our segmentation results. However, when training with less shapes, our segmentation results are better than learning-based segmentation results; see the results of "SB6 2010" in (d) in Fig. 11.

7.4 Our method versus variational mesh decomposition method in [51]

Since both our method and variational mesh decomposition method in [51] use the Mumford–Shah segmentation model, we would like to compare the two methods in more details.

– Firstly, the discretization schemes are different. The approach in [51] uses piecewise linear function space to discretize the Mumford–Shah model for clustering the

Fig. 10 Our segmentation results of other models and CAD models

vertices of a mesh. Therefore, it is a vertex-based segmentation method. Our method uses piecewise constant function space to discretize the Mumford–Shah model for clustering the triangles of a mesh. It is a triangle-based method. Consequently, the algorithm in [51] generates some undetermined triangle strips; see the red regions in Fig. 12a and the first row of Fig. 14. The triangles in the red regions are not determined and a post-processing step is needed to segment these red regions. This problem does not exist in our method; see Fig. 12b and the second row in Fig. 14. Considering the non-uniform distributions of triangles on meshes, our method is more ready for region-based segmentation and does not need any post-processing step.

– Secondly, the initialization methods are different. As the Mumford–Shah model is non-convex, the solution depends on initialization severely. Initialization plays a key role for the algorithm to succeed. The method in [51] initializes the segmentation function **u** randomly. This causes their method quite unstable. In the first row of Fig. 14, we show their segmentation results by 8 different random initializations, which are 8 different results. Similarly, our method is also unstable with random initialization. Figure 13 presents segmentation results by our method with random initializations (with 5 different random initializations, our method only can produce two correct segmentation results). In order to overcome this problem, based on an interesting observation on the eigenvectors of Laplacian, we propose a simple and

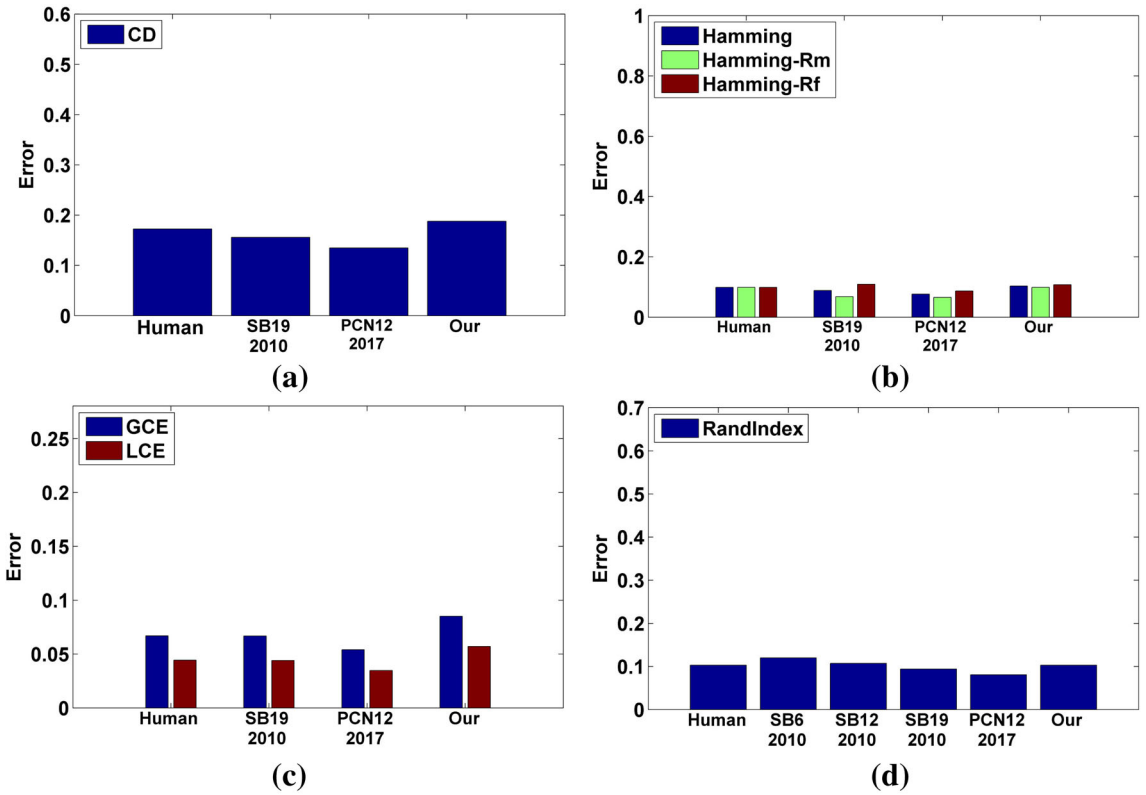


Fig. 11 Evaluation of segmentation in term of four evaluation metrics in [9] between our method and learning-based methods in [18,19]

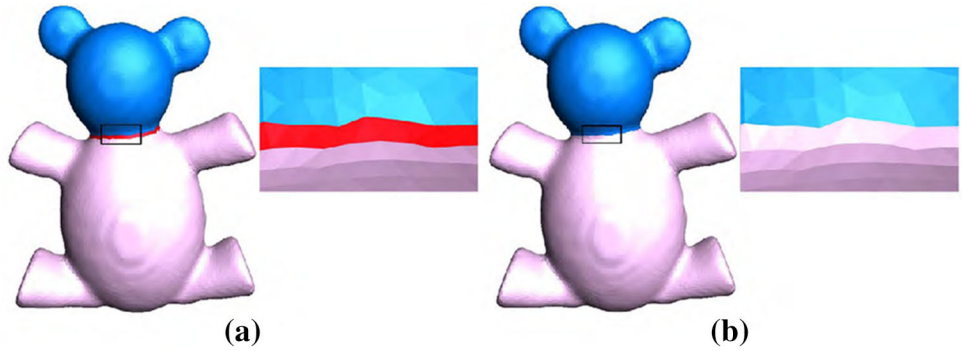


Fig. 12 Segmentation results by the vertex-based method in [51] (**a**) and our triangle-based method (**b**)

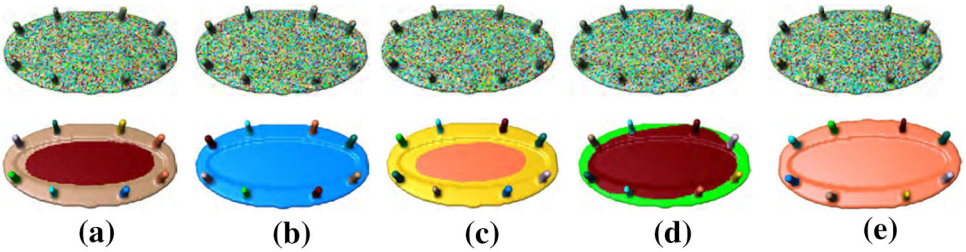


Fig. 13 Segmentation results produced by our method with random initialization

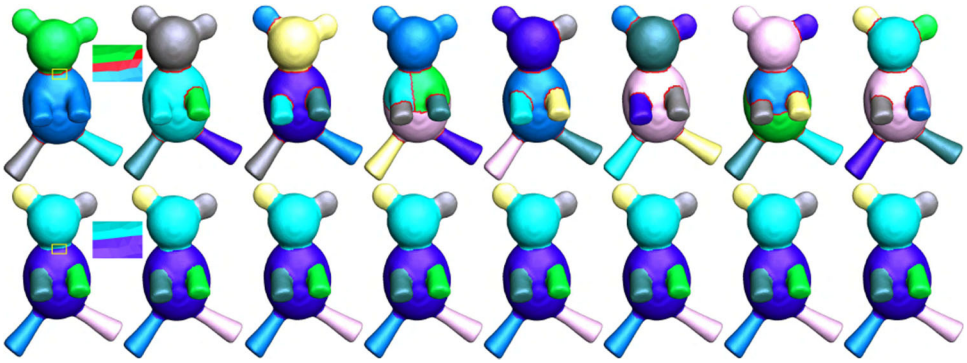


Fig. 14 Stability comparison between the segmentation method in [51] (*first row*) and our segmentation method (*second row*)

effective spectral analysis stage, which provides good initializations for the non-convex Mumford–Shah model. Our algorithm is thus very stable. See the second row in Fig. 14, where our segmentation method produces almost the same segmentation result. To further illustrate this difference, we tested more mesh surfaces and summarized in Table 2. As can be seen, the success rate of the method in [51] is much lower, and it even failed to segment correctly the Desk surface shown in Fig. 8 with 100 trials. However, our method always gave correct results (100 correct results in 100 trials for every tested surface).

– Thirdly, our initialization method cannot be used in the method [51]. As stated before, our variational method and method in [51] are both based on the classical Mumford–Shah model. Whereas, the discrete schemes are different. Our method is based on piecewise constant function space (see Fig. 1b), which clusters triangles. Then, our initialization $\mathbf{u^0} \in R^{\mathrm{T}\times K}$ is defined on triangles. In contrast, the method in [51] is based on piecewise linear function space (see Fig. 1a), which clusters vertices. The initialization $\mathbf{u}^0 \in R^{\mathrm{V}\times K}$ in [51] is defined on vertices. In all, the dimensions between our input and the input in [51] are totally different. Therefore, our initialization method cannot be used by the method in [51].
– Fourthly, for the same stopping condition, our algorithm needs fewer iteration steps than that in [51]; see the column "# Iterations averaged" of Table 2. The "–" in the table means the M-S method fails to segment that surface. In addition, the absolute error $\|\mathbf{u}^{n+1} - \mathbf{u}^n\|^2_{\mathbf{V}_M}$ with respect to the iteration number of our algorithm decreases much faster than that in [51], see Fig. 15.
– Finally, our segmentation method can generate more geometry-aware segmentations than the method in [51]. The two methods use different Laplacian matrices. Although effective in segmentation, the Laplacian matrix in [51] is defined by the difference of triangle normals and is severely sensitive to smooth normal variation. In contrast, the Laplacian in our method incorporates more neighborhood information of the mesh surface. It

Table 2 Comparisons on "# Iterations averaged" and "# correct segmentation results" between our method and that in [51]

Mesh	# Iterations averaged		# Correct segmentation results	
	M-S [51]	Our	M-S [51]	Our
Figure 8 Row 1	40	7	10	100
Figure 8 Row 2	25	5	15	100
Figure 8 Row 3	–	4	0	100
Figure 8 Row 4	35	4	45	100
Figure 8 Row 5	38	4	36	100

"# Iterations averaged" means the average number of iterative steps for the algorithm to reach the stopping condition 10^{-5} with 100 initializations; "# Correct segmentation results" represents the number of correct segmentation results by the algorithm with 100 initializations. The "–" in the table means the M-S [51] fails to segment that surface. It is clear that the success rate of our method is much higher than [51]

can reveal more semantic information and produce more robust segmentation results. See Fig. 2.

7.5 Computational cost

We now discuss the computational cost. Our method contains the computation of eigenvectors, spectral clustering, and variational refining procedure. The main part of computational cost is the iterative algorithm for the variational refining stage. In this step, each subproblem can be solved efficiently; see Sect. 6. Therefore, our algorithm is simple and efficient. The main ingredients affecting the computational cost are mesh size, segmentation number and the parameter α. The running time is more costly for large-size meshes and large-number segmentations. We also find that for small α, our algorithm is slow. For a model with about 30K triangles to be partitioned into 8 parts, our method takes about 8s in total. Table 3 presents the CPU costs for examples in Fig. 8.

Table 3 CPU costs of our segmentation method for surfaces in Fig. 8

Mesh	V	T	# Segmentation parts	CPU (s)
Figure 8 Row 1	8679	17,354	6	3
Figure 8 Row 2	13,463	26,926	8	7
Figure 8 Row 3	9802	19,600	9	5
Figure 8 Row 4	4501	8998	4	0.6
Figure 8 Row 5	8674	17,290	7	4

7.6 Limitations

Our method has been demonstrated very stable, effective and geometry-aware, but it still has some limitations. Firstly, we cannot give a formula to precisely compute the segmentation number. Secondly, for meshes with quite poor quality, our algorithm cannot give satisfying semantic segmentation; see Fig. 16 for an example. The mesh in Fig. 16a is quite irregular, in this case, we cannot get the correct seed triangles (see Fig. 16b) and the final segmentation result is also unsatisfying (see Fig. 16c).

8 Conclusions

In this paper, we presented a new mesh segmentation method, which contains two stages. The first stage is a spectral clustering procedure, and the second stage is a variational refining procedure. For the spectral clustering procedure, we constructed a new Laplacian matrix by incorporating more neighborhood information of the mesh surface. The new Laplacian matrix can reflect more geometry-aware information than classical Laplacian matrices. Based on the feature function defined by the eigenvectors of our new Laplacian matrix, we introduced a simple spectral clustering

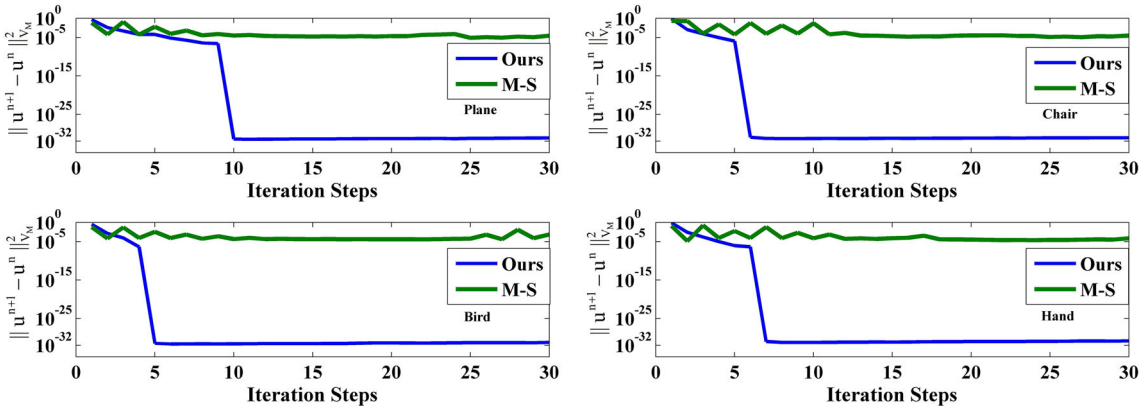


Fig. 15 Comparison on the absolute error $\|\mathbf{u}^{n+1} - \mathbf{u}^n\|^2_{\mathbf{V}_M}$ with respect to the iteration number: the segmentation method in [51] (*green line*) and our segmentation method (*blue line*)

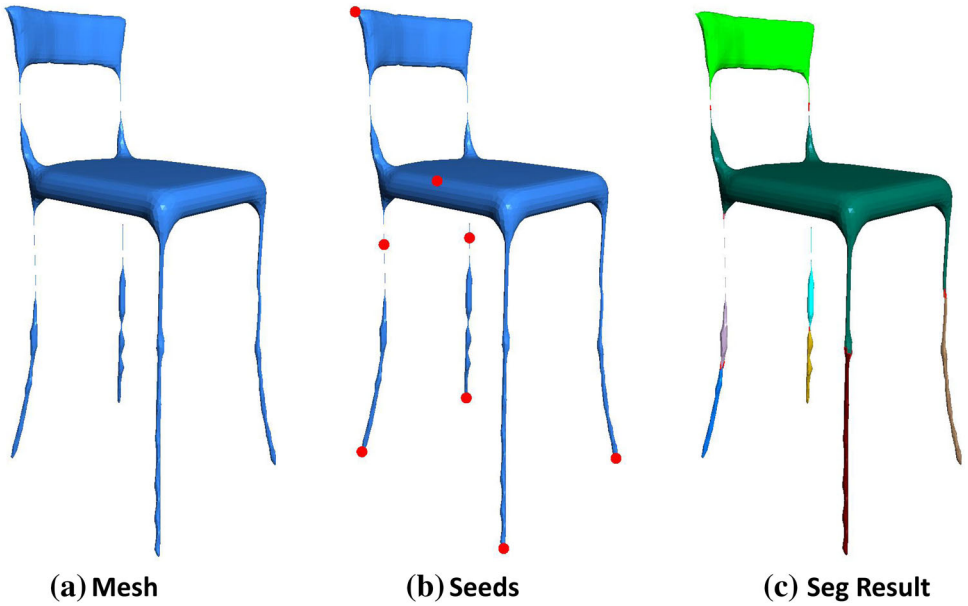


Fig. 16 Failed results produced by our cluster center algorithm and our segmentation method

method. Although, according to our consultation to several geometrists, the underlying principle is hard to rigorously prove, our spectral clustering procedure can produce satisfying segmentation results for most surfaces and provide a good initialization for the second stage, as demonstrated by our experiments. For the second stage, we devised a variational refining procedure by a new discretization and an efficient optimization algorithm for the classical non-convex Mumford–Shah model. The refining procedure directly clusters triangles instead of vertices, and thus generates real region-based segmentation results, avoiding post-processing steps for segmenting undetermined triangles. The optimization algorithm is an iterative algorithm based on a simple alternating minimization and ADMM, each step of which has closed form solution. Some convergence analysis has also been provided. With our spectral analysis as the initialization technique, our variational refining procedure works pretty well. There is no need for further complicated relaxations. It is also quite stable and effective, in contrast to other existing approaches with random initializations. Plenty of experiments (including tests on the Princeton Segmentation Benchmark) demonstrated the stability and effectiveness of our method. It outperforms competitive segmentation methods when evaluated on the Princeton Segmentation Benchmark.

There are a few problems for further investigation. For instance, the surface-type mesh segmentation and mesh simplification based on our piecewise constant function space will be reported in details subsequently.

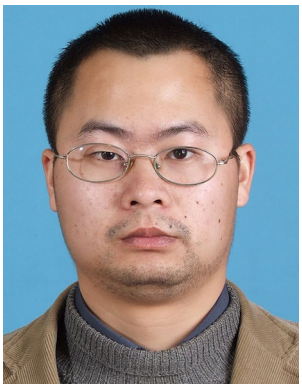
Acknowledgements We would like to thank Pengfei Xu, Noa Fish, Evangelos Kalogerakis for providing their segmentation data of [3,17] and [18,19], and the Princeton Segmentation Benchmark [9]. This work was supported by the NSF of China (Nos. 11626169, 11301289, 11371341, 61602341 and 11531013).

References

1. Agathos, A., Pratikakis, I., Perantonis, S., Azariadis, P.: 3d mesh segmentation methodologies for cad applications. Comput. Aided Design Appl. **4**(6), 827–841 (2007)
2. Attene, M., Falcidieno, B., Spagnuolo, M.: Hierarchical mesh segmentation based on primitives. Vis. Comput. **22**(3), 181–193 (2006)
3. Au, O., Zheng, Y., Chen, M., Xu, P., Tai, C.: Mesh segmentation with concavity aware fields. IEEE Trans. Vis. Comput. Graph. **18**(7), 1125–1134 (2012)
4. Benhabiles, H., Vandeborre, J., Lavoue, G., Daoudi, M.: A comparative study of existing metrics for 3D-mesh segmentation evaluation. Vis. Comput. **26**(12), 1451–1466 (2010)
5. Boyd, S., Parikh, N., Chu, E., Peleato, B., Eckstein, J.: Distributed optimization and statistical learning via the alternating direction method of multipliers. Found. Trend® Mach. Learn. **3**(1), 1–122 (2011)
6. Buades, A., Coll, B., Morel, J.: A non-local algorithm for image denoising. CVPR **2**, 60–65 (2005)
7. Chambolle, A.: Image segmentation by variational methods: Mumford and Shah functional and the discrete approximations. SIAM J. Appl. Math. **55**(3), 827–863 (1995)
8. Chan, T., Esedoglu, S., Nikolova, M.: Algorithms for finding global minimizers of image segmentation and denoising models. SIAM J. Appl. Math. **66**(5), 1632–1648 (2006)
9. Chen, X., Golovinskiy, A., Funkhouser, T.: A benchmark for 3D mesh segmentation. ACM Trans. Graph. **28**(3) (2009)
10. Cohen-Steiner, D., Alliez, P., Desbrun, M.: Variational shape approximation. ACM Trans. Graph. **23**(3), 905–914 (2004)
11. Delaunoy, A., Fundana, K., Prados, E., Heyden, A.: Convex multi-region segmentation on manifolds. In :IEEE 12th International Conference on Computer Vision, pp. 662–669. IEEE, (2009)
12. Garland, M., Willmott, A., Heckbert, P.: Hierarchical face clustering on polygonal surfaces. In Proceeding of ACM Symposium on Interactive 3D graphics, pp. 49–58. (2001)

13. Gelfand, N., Guibas, L.: Shape segmentation using local slippage analysis. In: Proceeding of SGP, pp. 214–223. (2004)
14. Golovinskiy, A., Funkhouser, T.: Randomized cuts for 3D mesh analysis. ACM Trans. Graph. **27**(5) (2008)
15. Guo, K., Zou, D., Chen, X.: 3d mesh labeling via deep convolutional neural networks. ACM Trans. Graph. **35**(1), 1–12 (2015)
16. Hoffman, D., Richards, W.: Parts of recognition. Cognition **18**(1–3), 65–96 (1984)
17. Kaick, O., Fish, N., Kleiman, Y., Asafi, S., Cohen-Or, D.: Shape segmentation by approximate convexity analysis. ACM Trans. Graph. **34**(1) (2014)
18. Kalogerakis, E., Averkiou, M., Maji, S., Chaudhuri, S.: 3D shape segmentation with projective convolutional networks. In: CVPR, pp. 1–11. (2017)
19. Kalogerakis, E., Hertzmann, A., Singh, K.: Learning 3D mesh segmentation and labeling. ACM Trans. Graph. **29**(3) (2010)
20. Katz, S., Leifman, G., Tal, A.: Mesh segmentation using feature point and core extraction. Vis. Comput. **21**(8–10), 649–658 (2005)
21. Katz, S., Tal, A.: Hierarchical mesh decomposition using fuzzy clustering and cuts. ACM Trans. Graph. **22**(3), 954–961 (2003)
22. Lai, R., Chan, T.: A framework for intrinsic image processing on surfaces. Comput. Vis. Image Underst. **115**(12), 1647–1661 (2011)
23. Lai, Y., Hu, S., Martin, R., Rosin, P.: Fast mesh segmentation using random walks. In: Proceedings of SPM, pp. 183–191. (2008)
24. Lavoue, G., Dupont, F., Baskurt, A.: A new cad mesh segmentation method, based on curvature tensor anaylsis. Comput. Aided Design **37**(10), 975–987 (2005)
25. Lellmann, J., Kappes, J., Yuan, J., Becker, F., Schnoerr, C.: Convex multi-class image labeling by simplex-constrained total variation. Lect. Note Comput. Sci. **5567**, 150–162 (2009)
26. Lellmann, J., Schnoerr, C.: Continuous multiclass labeling approaches and algorithms. SIAM J. Imaging Sci. **4**(4), 1049–1096 (2011)
27. Lie, J., Lysaker, M., Tai, X.-C.: A binary level set model and some applications for Mumford–Shah image segmentation. IEEE Trans. Image Proc. **15**(5), 1171–1181 (2006)
28. Liu, R., Zhang, H.: Segmentation of 3d meshes through spectral clustering. In: Proceedings of the Pacific Conference on Computer Graphics and Applications, pp. 298–305. (2004)
29. Liu, Z., Tang, S., Bu, S., Zhang, H.: New evaluation metrics for mesh segmentation. Comput. Graph. **36**(6), 553–564 (2013)
30. Lloyd, S.: Least square quantization in pcm. IEEE Trans. Inf. Theor. **28**, 129–137 (1982)
31. Luxburg, U.V.: A tutorial on spectral clustering. Stat. Comput. **17**(4), 395–416 (2007)
32. Mangan, A., Whitaker, R.: Partitioning 3D surface meshes using watershed segmentation. IEEE Trans. Vis. Comput. Graph. **5**(4), 308–321 (1999)
33. Michelot, C.: A finite algorithm for finding the projection of a point onto the canonical simplex of r^n. J. Optim. Theor. Appl. **50**(1), 195–200 (1986)
34. Mumford, D., Shah, J.: Optimal approximations by piecewise smooth functions and associated variational problems. Commun. Pur. Appl. Math. **42**(5), 577–685 (1989)
35. Ng, M., Weiss, P., Yuan, X.: Solving constrained total-variation image restoration and reconstruction problems via alternating direction methods. SIAM J. Sci. Comput. **32**(5), 2710–2736 (2010)
36. Shamir, A.: A survey on mesh segmentation techniques. Comput. Graph. Forum **27**(6), 1539–1556 (2008)
37. Shapira, L., Shamir, A., Cohen-Or, D.: Consistent mesh partitioning and skeletonisation using the shape diameter function. Vis. Comput. **24**(4), 249–259 (2008)
38. Shi, J., Malik, J.: Normalized cuts and image segmentation. IEEE Trans. Pattern Anal. Mach. Intell. **22**(8), 888–905 (2000)
39. Vese, L., Chan, T.: A multiphase level set framework for image segmentation using the Mumford and Shah model. Int. J. Comput. Vis. **50**(3), 271–293 (2002)
40. Wang, P., Fu, X., Liu, Y., Liu, S., Guo, B.: Rolling guidance normal filter for geometric processing. ACM Trans. Graph. **34**(6), 173 (2015)
41. Wu, C., Tai, X.: Augmented Lagrangian method, dual methods, and split Bregman iteration for rof, vectorial tv, and high order models. SIAM J. Imaging Sci. **3**(3), 300–339 (2010)
42. Wu, C., Tai, X.: A level set formulation of geodesic curvature flow on simplicial surfaces. IEEE Trans. Vis. Comput. Graph. **16**(4), 647–662 (2010)
43. Wu, C., Zhang, J., Duan, Y., Tai, X.: Augmented lagrangian method for total variation based image restoration and segmentation over triangulated surfaces. J. Sci. Comput. **50**(1), 145–166 (2012)
44. Wu, J., Kobbelt, L.: Structure recovery via hybrid variational surface approximation. Comput. Graph. Forum. **24**(3), 277–284 (2005)
45. Xin, S., He, Y., Fu, C.: Efficiently computing exact geodesic loops within finite steps. IEEE Trans. Vis. Comput. Graph. **18**(6), 879–889 (2012)
46. Yan, D., Wang, W., Liu, Y., Yang, Z.: Variational mesh segmentation via quadric surface fitting. Comput. Aided Design **44**(11), 1072–1082 (2012)
47. Yang, J., Zhang, Y., Yin, W.: A fast alternating direction method for tvl1–l2 signal reconstruction from partial fourier data. IEEE J. Select. Topic Signal Process. **4**(2), 288–297 (2010)
48. Zhang, H., van Kaick, O., Dyer, R.: Spectral mesh processing. Comput. Graph. Forum. **29**(6), 1865–1894 (2010)
49. Zhang, H., Wu, C., Zhang, J., Deng, J.: Variational mesh denoising using total variation and piecewise constant function space. IEEE Trans. Vis. Comput. Graph. **21**(7), 873–886 (2015)
50. Zhang, J., Zheng, J., Cai, J.: Interactive mesh cutting using constrained random walks. IEEE Trans. Vis. Comput. Graph. **17**(3), 357–367 (2011)
51. Zhang, J., Zheng, J., Wu, C., Cai, J.: Variational mesh decomposition. ACM Trans. Graph. **31**(3), 21 (2012)
52. Zhang, W., Deng, B., Zhang, J., Bouaziz, S., Liu, L.: Guided mesh normal filtering. Comput. Graph. Forum **34**(7), 23–34 (2015)
53. Zheng, Y., Tai, C., Wu, O.K.-C.: Dot scissor: A single-click interface for mesh segmentation. IEEE Trans. Vis. Comput. Graph. **18**(8), 1304–1312 (2012)
54. Ziemer, W.: Weakly Differentiable Functions, vol. 120. Springer, New York (1989)

Huayan Zhang received the Ph.D. degree from the University of Science and Technology of China, Hefei, People's Republic of China, in 2015. Currently, she is a assistant professor in the School of Computer Sciences, Tianjin Polytechnic University. Her research interests are in computer graphics and geometric computation.



Chunlin Wu was born in Jiangxi, People's Republic of China, in 1982. He received the PhD degree from the University of Science and Technology of China, Hefei, People's Republic of China, in 2006. Currently, he is an associate professor in the School of Mathematics, Nankai University, China. His research interests are in computer graphics and image processing.

Jiansong Deng was born in Shandong, People's Republic of China, in 1971. He received the PhD degree from the University of Science and Technology of China, Hefei, People's Republic of China, in 1998. Currently, he is a professor in the School of Mathematical Sciences, University of Science and Technology of China. His research interests include computer-aided geometric design and computer graphics.

Zheng Liu received the B.Sc. and M.Sc. degrees in computer science and technology from China University of Geosciences (Wuhan), in 2006 and 2009, respectively, and the Ph.D. degree in instructional technology from Central China Normal University, in 2012. He is currently a assistant professor with National Engineering Research Center of Geographic Information System, China University of Geosciences (Wuhan). From 2013 to 2014, he held a postdoctoral position with School of Mathematical Sciences, University of Science and Technology of China. His main interests include digital geometry processing, image and video processing.

Yuning Yang received the Ph.D. degrees in computational mathematics from Nankai University, Tianjin, China, in 2013. He has been a Post-Doctoral Researcher with the Department of Electrical Engineering, Stadius Centre for Dynamical Systems, Signal Processing and Data Analytics, Katholieke Universiteit Leuven, Leuven, Belgium, with Prof. J. A. K. Suykens, since 2013. His current research interests include optimization, machine learning, and multilinear algebra.