Spectral 3D Mesh Segmentation with a Novel Single Segmentation Field

Hao Wang[a], Tong Lu[a], Oscar Kin-Chung Au[b], Chiew-Lan Tai[c]

[a] *State Key Lab for Novel Software Technology, Nanjing University, Nanjing, China*
[b] *School of Creative Media, City University of Hong Kong*
[c] *Department of Computer Science and Engineering, the Hong Kong University of Science and Technology*

Abstract

We present an automatic mesh segmentation framework, which achieves 3D segmentation in two stages, comprising *hierarchical spectral analysis* and *isoline-based boundary detection*. During hierarchical spectral analysis, a novel single segmentation field is defined to capture concavity-aware decompositions of eigenvectors from a concavity-aware Laplacian. Specifically, on the eigenvector hierarchy, a sufficient number of eigenvectors is first adaptively selected and simultaneously partitioned into sub-eigenvectors through spectral clustering. Next, on the sub-eigenvector hierarchy, we evaluate the confidence of identifying a spectral-sensitive mesh boundary for each sub-eigenvector by two joint measures, namely, *inner variations* and *part oscillations*. Selection and combination of sub-eigenvectors are thereby formulated as an optimization problem to generate a single segmentation field. During the isoline-based boundary detection, segmentation boundaries are recognized by a divide-merge algorithm and a cut score, which respectively filters and measures desirable isolines directly from the concise single segmentation field. Experimental results on the Princeton Segmentation Benchmark and a number of complex meshes demonstrate the effectiveness of the proposed method, which is comparable to recent state-of-the-art algorithms.

Keywords: single segmentation field, spectral analysis, sub-eigenvector, isoline

1. Introduction

Mesh segmentation is a fundamental problem in geometric processing and shape understanding. It aims at decomposing a 3D polygonal mesh into disjoint but meaningful parts. Mesh segmentation provides a high-level structure of a mesh, and thereby serves as an initial step of numerous tasks, such as 3D modeling, animation, surface parameterization, compression, manufacturing, and shape processing [1, 2]. Most existing methods yield good results when segmenting an individual 3D mesh by computing classical geometric features [3, 4, 5], or co-segmenting various 3D meshes simultaneously using data-driven statistical techniques [6, 7]. They have achieved promising results which are comparable to human work. However, automatic mesh segmentation without any prior knowledge or any human assistance is still an open and challenging problem due to the lack of shape semantics in mesh representation [8, 9].

Recently, spectral analysis shows its effectiveness in 3D mesh segmentation by manipulating eigenvalues, eigenvectors, eigenspace projections, or a combination of these quantities derived from an appropriately defined linear operator [10]. Eigenvectors are especially commonly used and called spectral components. The success is due to the fact that the harmonic behavior of spectral components can individually reveal the underlying shape characteristics. However, most of the existing spectral-based mesh segmentation algorithms focus on using spectral components as a kind of low-level feature for clustering [3, 8] or eigenspace projection [4] to explore potential mesh boundaries, rather than explicitly discover geometric associations hidden in spectral components to facilitate automatic 3D mesh segmentation.

Essentially, we show that spectral components derived from a concavity-aware Laplacian operator of a mesh have the capability to explicitly characterize its geometric concavity information. Fig. 1 shows an *elk* mesh with its spectral analysis results by respectively using the first four eigenvectors. We made two observations. Our first observation is that the decomposition of eigenvectors is required for accurate segmentation. Take the result in Fig. 1(b) to (e) as an

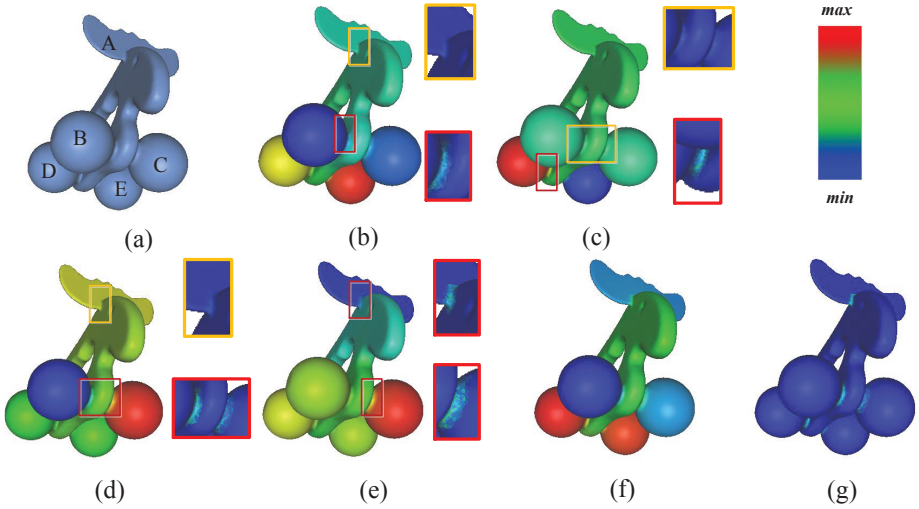


Figure 1: Automatic mesh segmentation by exploring geometric associations hidden in spectral components: (a) original *elk* mesh consisting of five parts A, B, C, D, and E, (b) B, C, D and E can be segmented using the first eigenvector, (c) segmenting D and E using the second eigenvector, (d) segmenting B and C using the third eigenvector, (e) segmenting A and C using the fourth eigenvector, (f) combining the four eigenvectors into a single segmentation field, and (g) the gradient map of (f). Gradient variation in a red rectangle indicates a potential spectral-sensitive mesh boundary, while a yellow rectangle denotes useless or even negative sub-eigenvectors for segmentation. Note that, for simplicity, not all such rectangles are drawn.

example. The sub-eigenvectors indicated by the red rectangle have large gradient variations and thereby serve as a basis for segmentation. Comparatively, the sub-eigenvectors indicated by the yellow rectangles are useless or even play a negative role in segmentation. Accordingly, our second observation is that high-quality automatic mesh segmentation can be derived from an optimized selection strategy from a large number of sub-eigenvectors. All the selected sub-eigenvectors need finally be combined together to provide a uniform and concise representation for automatically segmenting meshes.

Inspired by the observations, we propose a fully automatic mesh segmentation method that exploits hierarchical spectral analysis to systematically discover the relationship between desirable segmentation boundaries on a mesh and

algebraic properties of its eigenvectors. The core of our method is to derive a *single segmentation field* to identify spectral-sensitive mesh boundaries for high-quality mesh segmentation. We consider spectral analysis for each mesh on two hierarchies, namely, eigenvector and sub-eigenvector. On the eigenvector hierarchy, a sufficient number of eigenvectors are adaptively selected, each of which is further partitioned into sub-eigenvectors through spectral clustering. For the sub-eigenvector hierarchy, we evaluate the confidence of identifying a spectral-sensitive mesh boundary for each sub-eigenvector by two joint measures *inner variations* and *part oscillations*, which compute the gradient magnitude and spectral domain, respectively. Selection and combination of sub-eigenvectors can then be formulated as an optimization problem, which further result in a single segmentation field representation to inherit the merits of characterizing spectral-sensitive boundaries from various eigenvectors in a concise way.

Segmentation boundaries are recognized by directly searching for the isolines from the single segmentation field, based on the fact that field variations along an isoline has already been minimized following the minima rule as in [9]. Specifically, we first uniformly sample isolines from the single segmentation field, then adopt a divide-merge algorithm to filter and group the isolines by their values and positions. A cut score is defined for each isoline to measure its quality as a spectral-sensitive boundary during this stage, and the final segmentation boundaries are selected from isoline groups using the cut score.

The main contributions of the paper are 1) the introduction of a single segmentation field which successfully captures concavity-aware decompositions of eigenvectors, and 2) an automatic mesh segmentation framework that supports hierarchical spectral analysis. Theoretically, the single segmentation field in our method is the optimized combination of the useful components from the eigenvectors, which has the ability to characterize boundary cues. Accordingly, the isolines sampled from the field can well combine the contributions from multiple eigenvectors in a concise and thus more efficient way for automatic mesh segmentation. From this perspective, the proposed single segmentation field enables a novel approach for compact mesh representation, and a differ-

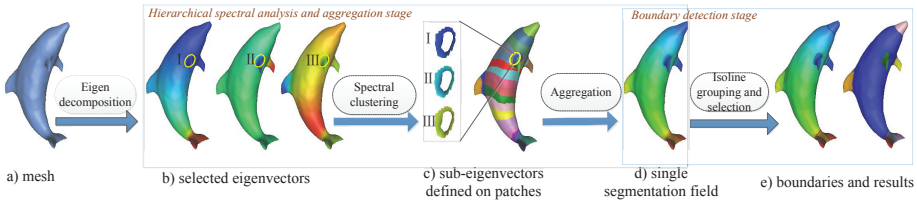


Figure 2: Our segmentation algorithm pipeline.

ent selection strategy for sub-eigenvectors will potentially lead to many other mesh-based applications. In our experiments, the proposed approach outperforms the recent Isoline Cuts algorithm of Au et al. [9] and the state-of-the-art non-learning M-S method of Zhang et al. [8] respectively by 1.1% and 0.3% averagely on the PSB benchmark [11] and a number of complex meshes. Note that neither a desirable number of mesh segments nor other user tuned parameters is necessary with our method.

The rest of the paper is organized as follows. Section 2 reviews related works. Section 3 shows our single segmentation field, then Section 4 gives the details of the proposed automatic mesh segmentation algorithm on the proposed segmentation field. Experiments and discussions are presented in Section 5. Finally, Section 6 concludes the paper.

2. Related Work

Mesh segmentation has been an active research topic in computer graphics community after Mangan and Whitaker [12] first extended the morphological watershed algorithm from images to surface meshes. Existing mesh segmentation algorithms can be roughly categorized into three classes, namely, *spectral analysis*, *region growing*, and *statistic learning*.

Spectral methods are adopted in mesh segmentation [3, 8, 4] due to its success in 3D geometry analysis of shape correspondence [13, 14] and quadrangulation [15]. Liu and Zhang [3] derive eigenvectors from adjacent matrix as features and use K-Means algorithm to cluster faces into segments. In their further work [4], the outer contour of the 2D spectral embedding of the mesh is used

to guide segmentation. Consequently, mesh vertices from continuous concave region will be close in the embedding space. Recently, Zhang et al. [8] extend the Mumford-Shah model to 3D meshes by measuring the variation within a segment through eigenvectors of a dual Laplacian matrix. The weights are computed by the dihedral angle between adjacent triangles. Another recent work on heat walk segmentation [16] can also be considered as a spectral method since the heat kernel is expressed in terms of eigenvalues and eigenvectors. Moreover, their method has the ability to utilize the implicit shape geometry information contained in spectral components. Most of the methods focus on using the raw spectral components as low-level feature descriptors, but seldom explore the relationship with geometry characteristics of a mesh. An interesting observation on this is in [17], where spectral analysis of the normalized geodesic distance matrix of vertices is found to give good results in selecting seed candidates, especially when the mesh has a large distortion.

Region growing is an intuitive approach that starts with a seed region on the mesh and then grows the seed by incrementally adding adjacent sub-meshes. Generally, the main differences among various region growing algorithms are the two criterions comprising seed selection and determination of whether a sub-mesh can be merged during growth. Local geometries such as convex [18], concave [19], hyperbolic vertices [5], curvature [20], and geodesic distance [21] are respectively considered to simplify the two criterions, but simultaneously bringing the major drawback of its dependence on the initially selected seed [22, 23]. It in turn leads to two variations of region growing. The first variation is to start from multiple seeds instead of a single source. A selection function is in general required to search for all the local minima of the function and each minimum serves as an initial seed on a mesh surface [12, 24]. In [25], a directional curvature height function is defined to start seed selection at expected vertices. Wang and Yu [26] propose a Morse function on the smoothed curvatures by bilateral filtering to extract the critical points as growing seeds. However, in practice such a selection function is not always easy to find. It inspires another optimization-based variation, that is, finding globally optimized

mesh parts under specific constraints after randomly selecting seeds on a mesh [27, 3]. Attene et al. [28] use the fits of geometric primitives as the constraint set to merge clusters at each stage of hierarchical clustering, which is able to choose the best merging from all clusters globally. Golovinskiy et al. [7] adopt the similar hierarchical clustering strategy but with a different optimization function of area-normalized cut cost.

Region growing methods work well on meshes but still may create unsatisfied results without knowing the accurate number of regions to segment. Shapira et al. [29] propose a feature shape diameter function to determine the diameter of a shape part. They then use the Gaussian Mixture Model to predict the probability assigned to each center and accordingly explore boundaries by solving an optimization function. Recently, Au et al. [9] and Zheng et al. [30] extract mesh boundaries by computing multiple segmentation fields. After solving a Laplacian system, they collect isolines from each field and select several of the isolines as the final boundaries. However, the segmentation fields cannot always guarantee the coverage of all potential segmentation information especially for the meshes without obvious protrusions.

Statistical learning methods segment 3D meshes through a data-driven approach. Golovinskiy and Funkhouser [7] use the randomized cuts technique to segment 3D surface meshes. Their strategy is to measure how often each edge lies on a segmentation boundary after generating a random set of mesh segmentations to guide boundary computation. Kalogerakis et al. [6] formulate an objective function as a Conditional Random Field model, with terms assessing the consistency of faces and terms between labels of neighboring faces. They use hundreds of geometric and contextual label features to learn different types of segmentations for different tasks. Moreover, problem-specific parameters from training examples are learned rather than manually-tuned. Benhabiles et al. [31] learn boundary edge functions to produce segmentation boundaries and achieve state-of-the-art results on the PSB dataset [11]. The potential drawback of data-driven methods may be its computational complexity and extra learning cost. Recently, Huang et al. [32] jointly segment shapes by utilizing

features from multiple shapes to improve the segmentation of a single mesh in an unsupervised way. However, the collection of a relatively large number of 3D meshes is still required.

3. Our approach

The overview of our framework is shown in Fig. 2, which consists of two stages, namely, *hierarchical spectral analysis* and *mesh boundary detection*. During the first stage, we construct a single segmentation field for each mesh in the following steps: 1) define a concavity-aware Laplacian operator according to the geometry of the input mesh, 2) obtain a sequence of eigenvectors through eigen-decomposition of the concavity-aware Laplacian operator, 3) adaptively select the derived eigenvectors that indicate useful geometry information for mesh segmentation by frequency analysis on the eigenvector hierarchy, 4) split the input mesh into patches via spectral clustering on the selected eigenvectors and obtain the sub-eigenvectors defined on them, and 5) extract the concavity information from the sub-eigenvectors to construct a single segmentation field by an optimization strategy on the sub-eigenvector hierarchy. In the second stage, we devise a divide-merge algorithm to automatically detect desired segmentation boundaries from isolines sampled from the single segmentation field.

3.1. Definition of the single segmentation field

Our single segmentation field is built on the eigenvectors and the sub-eigenvectors of a mesh Laplacian. Such a field has the following two merits. First, the field has the ability to gather sufficient boundary cues and simultaneously avoid the influence from irrelevant eigenvectors or sub-eigenvectors. Second, it allows detection of segmentation boundaries by directly sampling the isolines on the concise field representation that well inherits the harmonic behavior of the selected sub-eigenvectors. Based on these two considerations, we define our single segmentation field $\mathbf{f}$ for any mesh $\mathcal{M} = (\mathcal{G}, \mathcal{P})$, where $\mathcal{G} = (V, E)$ is a mesh graph in which V and E respectively denote its vertex set

and edge set, and $\mathcal{P}$ represent the coordinates of the vertices, by minimizing the following quadratic energy:

$$e(\mathbf{f}) = \sum_{i} \sum_{(j,k)\in E} w_{jk}^{i} |\mathbf{f}_j - \mathbf{f}_k - s_{jk}^{i}(\phi_{ij}^{L} - \phi_{ik}^{L})|^2 + c_1 |\mathbf{f}_1 - c_2|^2 \tag{1}$$

where ϕ_i^L is the ith selected eigenvector from the Laplacian operator L, w_{jk}^i and $s_{jk}^i \in \{+1, -1\}$ respectively denote the weight and the sign of edge (j, k), which are both dominated by the corresponding sub-eigenvector belonging to ϕ_i^L. Formally, $\mathbf{f}$ is a discrete vector and $\mathbf{f}_j$ denotes its value on the jth vertex. The second item in Equ. 1 is the boundary condition, c_1 and c_2 are two constants (we use $c_1 = 100$ and $c_2 = 1$).

Equivalently, we obtain the single segmentation field $\mathbf{f}$ in a least-square sense by solving the linear system $\mathbf{A}\mathbf{f} = \mathbf{b}$, with $\mathbf{A}$ and $\mathbf{b}$ respectively defined as:

$$\mathbf{A} = \begin{pmatrix} \mathbf{W}^1\Delta \\ \vdots \\ \mathbf{W}^n\Delta \\ c_1\mathbf{c}^{\mathrm{T}} \end{pmatrix} \mathbf{b} = \begin{pmatrix} \mathbf{W}^1\mathbf{S}^1\Delta\phi_1^L \\ \vdots \\ \mathbf{W}^n\mathbf{S}^n\Delta\phi_n^L \\ c_1c_2 \end{pmatrix} \tag{2}$$

where Δ is the first-order edge difference matrix of size $|E| \times |V|$, in which a nonzero value $\Delta_{ej} = 1$ if j is the larger vertex index on edge e or $\Delta_{ej} = -1$, otherwise. c is a constant vector with 1 in the first position and 0 in the others, $\mathbf{S}^i$ is an edge sign matrix, and $\mathbf{W}^i$ is a diagonal weight matrix in which both the sign and the weight of an edge is inherited from its corresponding sub-eigenvectors.

In this way, value variations of the selected eigenvectors are aggregated after evaluating the weights of the sub-eigenvectors. The details of the mentioned concavity-aware Laplacian operator, the selection of eigenvectors, the extraction of sub-eigenvectors and its weight definition as well as the construction of the sign matrix are respectively discussed in the following subsections.

3.2. Building concavity-aware Laplacian

For our segmentation utility, every derived eigenvector that indicates an intense variation around a potential segmentation boundary is desired. We employ the unnormalized form of discrete Laplacian operator, and obtain the eigen-decomposition solution using the arpack packet [33]. Our concavity-aware Laplacian $\mathbf{L}$ is defined by

$$\mathbf{L} = \begin{cases} \omega_{ij}^{L} = \frac{e_{ij}}{\bar{e}}\Theta_{ij} & (i,j) \in E \\ -\sum_i \omega_{ij}^{L} & i = j \\ 0 & otherwise \end{cases} \tag{3}$$

where e_{ij} denotes the length of an edge that connects two vertices i and j, and $\bar{e}$ is the average edge length in $\mathcal{M}$. The concavity information is encoded by Θ, which is characterized by the concave vertices lying on concave seams. As in [9], our Laplacian operator is also built on concave vertices but with an enhanced detection strategy.

The concave vertices lying on potential segmentation boundaries form the key ingredient in constructing our concavity-aware Laplacian. Preliminarily, concave vertices are the mesh vertices satisfying the following condition used in [9]:

$$\langle \mathbf{u}_{ij}, \mathbf{n}_j - \mathbf{n}_i \rangle > \varepsilon \tag{4}$$

where n_i and n_j respectively denote the outward normals of vertex i and vertex j, u_{ij} is the unit direction vector determined by i and j, and $\varepsilon = 0.01$ is an empirical truncation threshold. Equ. 4 is an edge-level condition to initially filter the vertices that do not lie on concave seams. All other vertices are considered as candidate convex vertices and further filtered by two region-level filters:

1. **Filter 1:** Preserve every candidate concave vertex on the condition that it has enough two-ring neighbors that successfully vote for the vertex. Thus, the candidate concave vertices lying on *drape*-like regions which meet Equ. 4 but essentially cannot be considered as segmentation boundaries will be filtered. The proportion of enough neighbors here is empirically set as 20% according to our experiments.

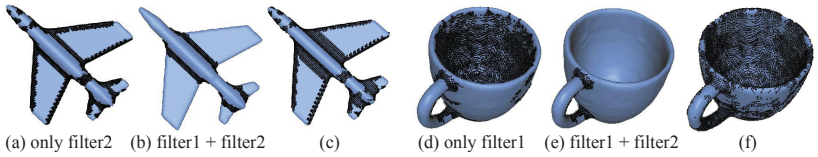


Figure 3: Detecting concave vertices with only Filter 1 or 2 (a and d) and with both Filters 1 and 2 (b and e). For comparison, (c) and (f) are derived from [9]. It can be found that the undesirable concave vertices, which are useless for segmentation, can be effectively reduced by using the proposed filters.

2. **Filter 2:** Similarly filter the concave points that lie on a locally flat region with low concavity via applying PCA (Principal Component Analysis) on the vertex position of the one-ring neighbors. Considering there are altogether three principals that are in a descending order of eigenvalue, we preserve the concave points only when its third principal is not collapsed by

$$\frac{\mu_{i,3}^{P}}{\sum_k \mu_{i,k}^{P}} > \eta \tag{5}$$

where $\mu_{i,k}^{P}$ is the kth eigenvalue in PCA of vertex i, and $\eta = 0.01$ is an empirical threshold.

Finally, ω_{ij} can be initialized by setting Θ_{ij} as a small constant 0.01 if either vertex i or vertex j successfully passes both the concaveness test in Equ. 4 and the two filters, or initialized to constant 1 otherwise. The concavity-aware Laplacian is accordingly built. Fig. 3 shows two examples of our concave vertex detection. For comparison, we also show the detected vertices derived from [9], where a number of undesired noises are determined as concave vertices. It can be observed that most of such noises can be filtered using the proposed two filters.

Finally, we obtain a sequence of eigenvectors $[\phi_0^L, \phi_1^L, \ldots]$ with an increasing order of eigenvalues $[\lambda_0^L, \lambda_1^L, \ldots]$ after eigen-decomposition of our concavity-aware Laplacian matrix.

3.3. Hierarchical spectral analysis

After building the concavity-aware Laplacian, $\mathcal{M}$ will be respectively analyzed on the two hierarchies, namely, eigenvector and sub-eigenvector, aiming at constructing the single segmentation field representation to combine the useful eigenvectors in an optimized and concise way.

3.3.1. Holistic analysis and selection of eigenvectors

For the eigenvector hierarchy, the main task is to adaptively select the eigenvectors that can characterize the concavity information on $\mathcal{M}$. The underlying idea of characterizing concavity information through eigenvectors is inspired by the fact that their harmonic behaviors can reveal particular shape geometries individually. From the signal processing point of view, the Fourier spectrum analysis theory inspires us to associate the eigenvectors derived from our concavity-aware Laplacian to particular shape geometries on a mesh. Theoretically, any real symmetric matrix $\mathbf{A}$ of dimension n can be spectrally factorized by $\mathbf{A} = \sum_i^n \lambda_i \phi_i \phi_i^T$, where eigenvalue λ_i evaluates the contribution of eigenvector ϕ_i to reconstruct $\mathbf{A}$. Moreover, according to the Courant-Fischer-Weyl Theorem, eigenvalues $\lambda_1 < \lambda_2 < ... < \lambda_n$ satisfy the condition $\lambda_i = \min_{dim(\mathcal{V})=i} \max_{\mathbf{v}\in\mathcal{V}} \frac{\mathbf{v}^T\mathbf{A}\mathbf{v}}{\mathbf{v}^T\mathbf{v}}$, where $\mathcal{V}$ is a subspace of $\mathbb{R}^n$ and the optimized v equals to its corresponding ϕ_i. That is, the eigenvectors constituting $\mathbf{A}$ are derived from the low dimension subspaces to the high dimension subspaces sequentially. In the case of discrete Laplacian operator, each eigenvector thus has the ability to characterize a particular global shape geometry which in general corresponds to a low dimensional subspace, or some local geometry which corresponds to high dimensional subspaces.

We explain this phenomena by an *Armadillo* mesh as an example. There are theoretically altogether 60,000 increasingly ordered eigenvectors of the concavity-aware Laplacian. We visualize the first 18 eigenvectors in Fig. 4. Take the first eigenvector as an example, we find that the transition of its values from the maximum (e.g., the two hands colored *red*) to the minimum (e.g., the right leg colored *blue*) essentially covers multiple regions distributed on the *Armadillo*.

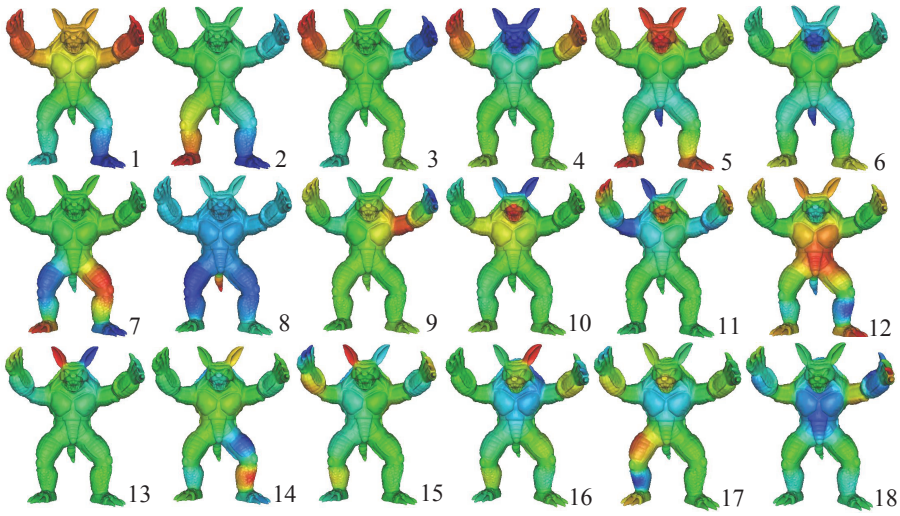


Figure 4: Visualization of the first 18 non-constant eigenvectors of our concavity-aware Laplacian for the *Armadillo* mesh. Values of the eigenvectors are mapped into colors from *red* (maximum) to *blue* (minimum).

Another example for comparison is the 14th eigenvector, where the main propagation from the maximum value to the minimum value only covers a local region on the right leg. Essentially, all the eigenvectors derived from our concavity-aware Laplacian can be similarly associated to characterizing either the global or the local geometries on a mesh.

Adaptively selecting a subset of eigenvectors for shape analysis is necessary because directly using all the $|V|$ eigenvectors (this number is theoretically equivalent to that of the vertices on $\mathcal{M}$) potentially requires a large computational cost. Moreover, unlike the Fourier frequency bases, the eigenvectors derived from any Laplacian operator are essentially mesh-relevant. That is, the selection of eigenvectors should be adaptively decided by the geometry on $\mathcal{M}$, rather than assigning a fixed number of eigenvectors for all kinds of meshes as in [3, 8].

We further hypothesize that the order and the eigenvalue magnitude of each eigenvector reflect the particular frequency information and the contribution of the frequency for reconstructing the original operator, respectively. The

eigenvectors that simultaneously have similar orders and eigenvalue magnitudes thereby can be categorized into the same frequency level, namely, an eigenvector group. Thus, the eigenvectors inside the same group can be essentially considered as the components that have similar contributions in the Laplacian operator for characterizing shape geometries on the same scale. Specifically, we group all the eigenvectors by their eigenvalues and then select the eigenvector groups which are on low frequency level for further sub-eigenvector decomposition in an adaptive way:

1. Model the frequencies in $\mathcal{M}$. After calculating the eigenvectors $[\phi_1^L, \phi_2^L, \ldots]$ and their corresponding eigenvalues $[\lambda_1^L, \lambda_2^L, \ldots]$ that are increasingly ordered, we search for all the local maximums $[max_1, max_2, ...]$ of the second-order difference of the eigenvalues, defined as:

$$SOD = |\delta\lambda_{i+1}^L - \delta\lambda_i^L| \tag{6}$$

   where $\delta\lambda_{i+1}^L = \lambda_{i+1}^L - \lambda_i^L$.

2. Group and select eigenvectors. We adaptively categorize all the eigenvectors between every two adjacent maximums into the same group from the local maximums on the second-order difference curve by

$$\Lambda_j = \{\phi_i^L | \lambda_i^L \in [max_j, max_{j+1})\} \tag{7}$$

   We found that a small number of eigenvector groups (about 10 eigenvectors) is sufficient for characterizing geometry variations in meshes, even for complex meshes. We thereby empirically preserve the first two groups that always correspond to a low frequency and directly discard all the rest. As a result, eigenvectors will be adaptively selected for $\mathcal{M}$. Note that the number of selected eigenvectors derived from different meshes generally differ from each other.

3. Use the selected eigenvector groups to generate patches on $\mathcal{M}$. We normalize all the selected eigenvectors to $[-1, 1]$, and apply the K-Means clustering method on the mesh vertices with the multichannel features

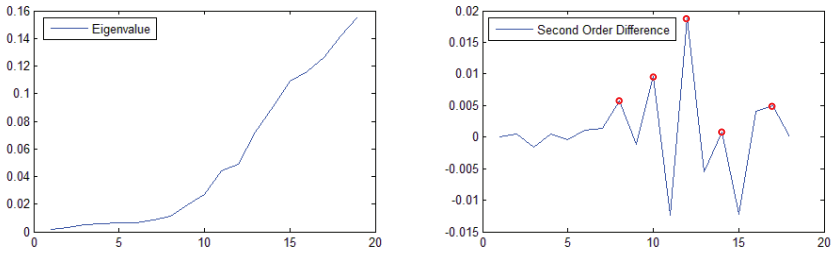


Figure 5: Adaptive selection of eigenvectors derived from our concavity-aware Laplacian on the eigenvector hierarchy. Left: the eigenvalue curve of the *ant* mesh in Fig. 6; Right: its second-order difference curve.

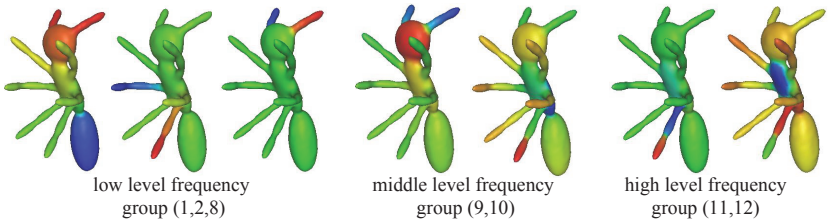


Figure 6: The eigenvector groups on three different frequency levels of the *ant* mesh, from which we can find that the groups on the low frequency level contain rich segmentation cues. Note that only 3 out of 8 eigenvectors in the first group are shown here.

that are extracted from the selected eigenvectors to generate a fixed number of patches on $\mathcal{M}$ (we use 50 patches). Each eigenvector is further decomposed into sub-eigenvectors according to the extracted patches such that each sub-eigenvector defines on the vertices of the associated patch.

As an example, Fig. 5 shows the eigenvalue curve of an *ant* mesh and its second-order difference curve. To show the influence of different frequencies on our adaptive eigenvector selection, Fig. 6 further visualizes several groups sampled from the first 20 eigenvectors on low-frequency, middle-level and high-frequency levels, respectively. Visually, the groups on the low frequency level contain our desired segmentation cues that are sufficient for mesh segmentation. Therefore, we only select the first two groups of eigenvectors and discard the rest.

3.3.2. Sub-eigenvector analysis

After selecting eigenvectors, all the sub-eigenvectors from the selected eigenvectors are analyzed on the sub-eigenvector hierarchy. How to discriminatively evaluate the potential contributions of the sub-eigenvectors in mesh segmentation, and thereby smartly combining the sub-eigenvectors according to their contributions is crucial in the optimization (Equ. 1 & 2).

The usefulness of each sub-eigenvector is observed from two aspects of spectral analysis: gradient magnitude and spectral domain. We encourage all the sub-eigenvectors that have large value gradient (see gradient maps in Fig. 1(g)), and simultaneously suppress all the sub-eigenvectors with their value range overlapping with some others, which are essentially considered as noise sub-eigenvectors for mesh segmentation. In this way, the selected sub-eigenvectors will be assigned with different weights in describing concavity-aware shape geometry and then combined together in a concise way for the final optimization.

We thereby define two joint measures comprising *inner variation* and *part oscillation* for every sub-eigenvector on the sub-eigenvector hierarchy. The former evaluates the sub-eigenvectors from different eigenvectors by computing their gradients, while the latter evaluates the sub-eigenvectors inside the same eigenvector by spectral domain analysis. Specifically, the inner variation measure encourages the sub-eigenvectors that simultaneously have a large face gradient magnitude and a consistent gradient direction, defined as follows:

- ***Inner Variation (IV)*** is a gradient-based weight to evaluate the shape concavity of each sub-eigenvector by

$$IV_{se} = \frac{|\sum_{i \in F_{se}} a_i \mathbf{g}_i|}{\sum_{i \in F_{se}} a_i} \tag{8}$$

  where F_{se} denotes the corresponding mesh faces on which the sub-eigenvector se is defined, and a_i is the area of the ith face. g_i is a gradient vector inside the ith face, which can be calculated by the linear variation assumption inside the face. IV_{se} helps distinguish the sub-eigenvectors that are from different eigenvectors on the same patch.

The part oscillation measure distinguishes the contributions of the sub-eigenvectors that are from the same eigenvector by analyzing their spectral value domains. From the intrinsic oscillatory nature, each eigenvector consists of two parts: several principle sub-eigenvectors that essentially indicate the useful and global stretch property (see the red rectangles for each eigenvector in Fig. 1), and the rest sub-eigenvectors with overlapped ranges, which is the oscillation part from the signal point of view, indicating the particular local detail information which contributes less to mesh segmentation (see the yellow rectangles in Fig. 1). We thus prefer to assign a larger weight to the principle sub-eigenvectors and simultaneously suppress the vibration noise ones using the part oscillation measure as follows:

- ***Part Oscillation (PO)*** is a spectral-based weight to evaluate the effectiveness of each sub-eigenvector inside the same eigenvector, defined as:

$$PO_{se} = \min_{se', se \in \phi_i^L, se' \neq se} irr(se, se')^{-1} \tag{9}$$

where $irr(r_{se}, r_{se'})$ denotes the intersected range ratio of the two sub-eigenvectors in the same eigenvector, like Jaccard similarity measurement, with the definition of

$$irr(se, se') = \frac{|range(se) \bigcap range(se')|}{|range(se) \bigcup range(se')|} \tag{10}$$

where $range(se)$ is a subinterval of $[-1, 1]$, denoting the value range of sub-eigenvector se.

Finally, we define the sub-eigenvector weight by combining the two complementary scores by

$$w_{se} \propto IV_{se} \cdot \log(PO_{se}) \tag{11}$$

where the logarithm function is set to leverage the two scores. This weight scheme is used to guide the combination of sub-eigenvectors, which acts as the

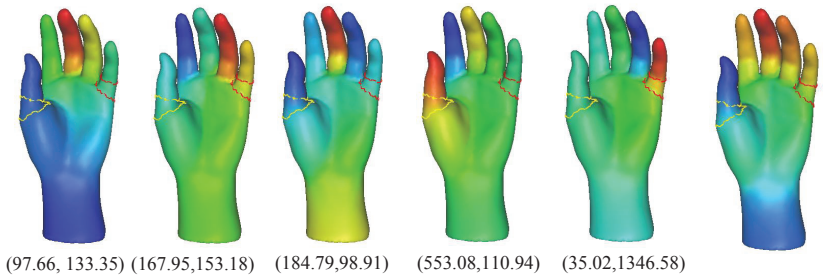


Figure 7: Quantitative evaluations of the joint unnormalized weight on the sub-eigenvectors of a *hand* model. Two patches are sampled with their boundaries colored *yellow* and *red*, respectively. Below each eigenvector, the unnormalized weights w_{se} of the two patches are shown. The left five are non-constant eigenvectors, and the rightmost is the derived single segmentation field representation by inheriting all the sub-eigenvectors using the weight scheme.

core of optimizing the single segmentation field. Specifically, all the weights of the sub-eigenvectors are organized into edge weight matrixes and assigned to $\mathbf{A}$ and $\mathbf{b}$ in Equ. 2.

Fig. 7 shows the quantitative results to illustrate our weight scheme on evaluating and combining sub-eigenvectors. The first five non-constant eigenvectors of a *hand* mesh are given in Fig. 7. We sample two example patches that are enclosed by the boundaries colored *yellow* on its thumb and *red* on its little finger, respectively. Below every eigenvector, the unnormalized weights $w_{se_{yellow}}$ and $w_{se_{red}}$ of the sub-eigenvectors on these two patches are listed. It can be found that our weighting scheme successfully assigns the largest weight in the fourth eigenvector valued 553.8 for the yellow patch, while the red patch is assigned with the largest weight in the fifth eigenvector valued 1346.58. Sub-eigenvectors defined on the patches are thus measured, aggregating the concavity-aware ones into our single segmentation field as shown in the rightmost of Fig. 7.

The hierarchical spectral analysis facilitates mesh segmentation in a more concise and accurate way comparing with the methods that directly use patch borderline as cut boundaries. Fig. 8(a) shows some patches examples by directly clustering on multidimensional eigenvectors. From Fig. 8(b), we find that several

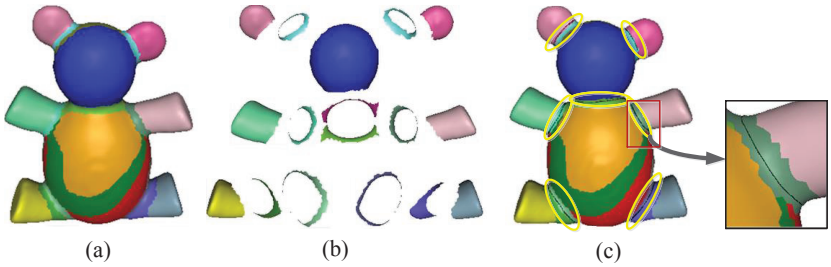


Figure 8: Mesh segmentation by directly using patch borderlines as cut boundaries. (a) the patches generated by clustering all the eigenvectors, (b) the sampled representative patches from (a), (c) the desired boundaries that meet human cognitions.

patches generated by the K-Means clustering algorithm tend to be a ring-like or a half-ring-like region. A ring-like region here generally connects two mesh segments, containing one desirable segmentation boundary or at least half of the desirable boundary (called half-ring-like region). The borderlines of such patches can be directly considered as potential segmentation boundaries. However, we can see from Fig. 8(c) that the desired cutting boundaries satisfying human cognitions always lie inside these patches. In other words, directly clustering the eigenvectors without the analysis on the two hierarchies comprising eigenvector and sub-eigenvector will potentially lead to inaccurate mesh segmentations. To refine the boundaries, extra computations will be necessary. For example, the M-S model presented in the recent work [8] is actually a K-Means formulation by adding an boundary smoothing term, an extra pre-computed segment number K and an optimization strategy, which thereby needs additional GPU accelerations to assure time efficiency.

3.4. Building edge sign matrix

According to the definition of our single segmentation field in Section 3.1, we finally introduce how to build the sign matrixes for each eigenvector. Since opposite propagation directions among different eigenvectors potentially counteract the effective value variations of eigenvectors in the additive-style formulation of Equ. 1, it is necessary to unify the propagation direction of the

selected eigenvectors before building the single segmentation field. By treating every sub-eigenvector as a whole on a patch, the edges on which it is defined share the same weight and sign. Then edge signs for eigenvectors to unify the direction can be organized into a sign matrix sequence. Formally, we set $\mathbf{S}^i$ as an $|E| \times |E|$ diagonal sign matrix with 1 or -1 on its diagonals.

We use the greedy algorithm to calculate the sign matrix. Specifically, we gradually compute the sign matrix according to an increasing order of the eigenvalues, thus preferentially matching low frequency eigenvectors. We initially set the first sign matrix $\mathbf{S}^1$ to be an identity matrix. Then in each iteration, a new sign matrix is determined to preserve the propagation direction with the weighted previous ones by

$$\mathbf{S}^{k+1} = argmax_{\mathbf{S}} \sum_{i=1}^{k} \langle \mathbf{S}\Delta\phi_{k+1}^{L}, \mathbf{S}^i \Delta\phi_i^L \rangle_{\mathbf{W}^i} \tag{12}$$

where $\mathbf{W}_i$ is the diagonal edge weight matrix and $\mathbf{W}$-inner-product means $\langle \mathbf{u}, \mathbf{v} \rangle_{\mathbf{W}} = \mathbf{u}^{\mathrm{T}}\mathbf{W}\mathbf{v}$. The iteration is repeated until all the sign matrixes are computed.

As a result of introducing the concavity-aware Laplacian operator, the selection of eigenvectors, the extraction of sub-eigenvectors and its weight definition as well as the construction of the sign matrix, our single segmentation field defined in Equ. 1 will be accordingly generated.

4. Sampling isolines and boundary cuts selection

The derived single segmentation field aggregates the sub-eigenvectors that possibly indicate desirable segmentation boundaries. Considering that the uniformly sampled isolines from our single segmentation field will be densely concentrated on concave regions, we further propose an isoline-based algorithm to explore the final segmentation boundaries directly from our single segmentation field.

We detect segmentation boundaries by isoline sampling and selection. To cover all the potential boundaries and generate high-quality candidate isolines,

we uniformly sample a large number of isolines from our single segmentation field. Each isoline here is represented by a sequence of connected line segments, which cross mesh faces with their end vertices possessing the same field value. Directly selecting isolines from multiple fields, like in [9], can be a choice; however, it faces the following difficulties 1) the noisy isolines that are not on the correct boundaries will mislead mesh segmentation, and 2) directly selecting the boundaries from all the isolines globally is inaccurate since the candidate isolines of different local boundaries are not comparable.

To solve these two difficulties, we propose a divide-merge algorithm to simultaneously filter non-boundary isolines and divide the rest into segment boundaries. A double threshold idea inspired by Canny edge detector [34] is adopted in the algorithm. Specifically, we first divide all the isolines into groups. The group of a small size is considered as noises and directly discarded. Then, we re-merge the groups that are close to each other from the rest into a larger one by evaluating their isovalues o and attributes $a = \{$isoline length l, isoline normals n, isoline centers $c\}$:

$$\begin{aligned} &dif_{o,a=\{r,n,c\}}(l_i, l_j) = \\ &\{|o_i - o_j|, \max\{\overline{l_i}/\overline{l_j}, \overline{l_j}/\overline{l_i}\}, acos\langle \mathbf{n}_i, \mathbf{n}_j \rangle, \|\mathbf{c}_i - \mathbf{c}_j\|\} \end{aligned} \tag{13}$$

where the absolute difference of o, ratio of l, inclined angle between n and the distance between c are calculated. Specifically, for each isoline, n is computed as the normal of its fitted plane and c is calculated as the average of the end vertices of its line segments. The details of the *divide* step and *merge* step are given in Algorithm 1 and Algorithm 2, respectively.

Next, in each of the groups, a boundary line is precisely compared and selected. We select the isoline that has the highest score from each group as our final segmentation boundaries by defining a *cut criterion* as follows:

$$sc_i = sc_{g,i} \cdot sc_{v,i} \cdot sc_{c,i} \cdot sc_{l,i}^{-1} \cdot sc_{m,i}^{-1} \tag{14}$$

where $sc_{g,i}$ and $sc_{v,i}$ respectively denote the gradient score and the shape variation score as in [9]. Since we need to select isoline candidates for the same

Algorithm 1 Divide Isolines into Groups

Input: an isoline set IS, a similarity threshold $sth = \{T_l, T_n, T_c\}$ and a density threshold $dth = \{T_d\}$

Output: Isoline groups $\{G\}$

$i \leftarrow 0, j \leftarrow 0$
Start with an isoline l_i and initial a group $G_j = \{l_i\}$, $IS = IS - \{l_i\}$
repeat
  Select l_k from IS satisfying $l_k = \text{argmin}_l dif_o(l, l_i)$
  $IS = IS - \{l_k\}$
  if $dif_a(l_k, l_i) \leq sth$ **then**
    $G_j = G_j \bigcup \{l_k\}$
  else
    if $|G_j| \leq dth$ **then**
      delete G_j
    end if
    $j \leftarrow j + 1$
    Initial another group $G_j = \{l_k\}$
  end if
  $i \leftarrow k$
until $IS = \varnothing$

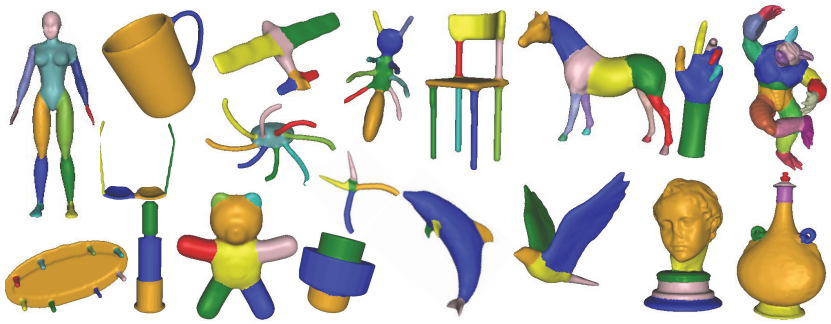


Figure 9: Mesh segmentation results by using our single segmentation field on the Princeton Segmentation Benchmark.

Algorithm 2 Merge Similar Isoline Groups

Input: isoline groups $\{G\}$ and a similarity threshold $sth = \{T_l, T_n, T_c\}$

Output: Merged isoline groups $\{G\}$

$sth \leftarrow sth \times 2$
for all G_i and G_j **do**
  Compute distance $d_{ij} = \min_{l \in G_i, l' \in G_j} dif_a(l, l')$
end for
repeat
  Select G_i and G_j by satisfying $(i, j) = \text{argmin} d_{ij}$
  if $d_{ij} \leq sth$ **then**
    $G_i = G_i \bigcup G_j$, delete G_j and recompute d_{i*}
  end if
until all $d_{ij} > sth$

boundary, we design effective scores to evaluate the quality of each isoline. Particularly, we introduce an extra length score $sc_{l,i}$, an extra smooth score $sc_{m,i}$ and an extra concavity score $sc_{c,i}$ to better characterize the quality of an isoline being a desirable segmentation boundary. Since human generally prefers to select short and tight lines as boundaries, we use length score $sc_{l,i}$, which is the length of isoline i to evaluate isolines . $sc_{c,i}$ is defined by modeling the concavity of a candidate isoline as follows

$$sc_{c,i} = \sum_{f \in F_i} \sum_{(p,q) \in f} \langle \mathbf{u}_{pq}, \mathbf{n}_q - \mathbf{n}_p \rangle \tag{15}$$

where $f \in F_i$ denotes the faces crossed by the isoline i, u_{pq} is defined in Equ. 4 and (p, q) is the edge of face f. Similarly, $sc_{m,i}$ is defined by the direction variation of each adjacent line segment to evaluate the smoothness of each isoline by

$$sc_{m,i} = \sum_{(p,q) \in l_i, (q,o) \in l_i} acos\langle \mathbf{u}_{pq}, \mathbf{u}_{qo} \rangle \tag{16}$$

where (p, q) and (q, o) denote two adjacent line segments of isoline l_i.

Finally, we partition a mesh by gradually adding each selected isoline that

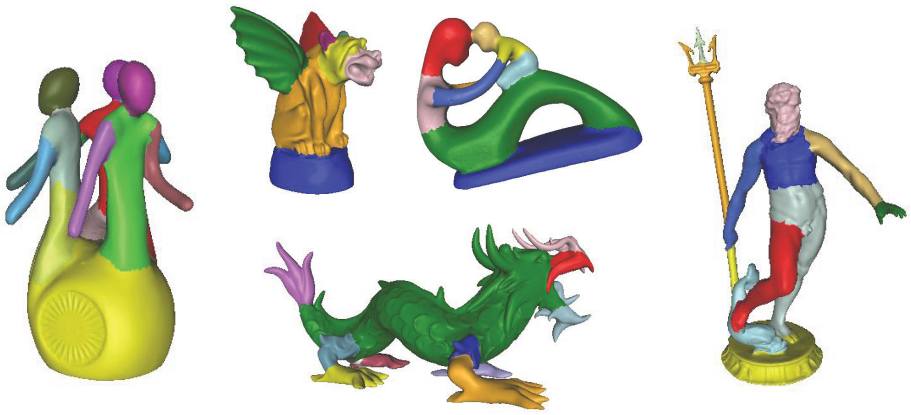


Figure 10: More results of complex meshes collected from the Internet.

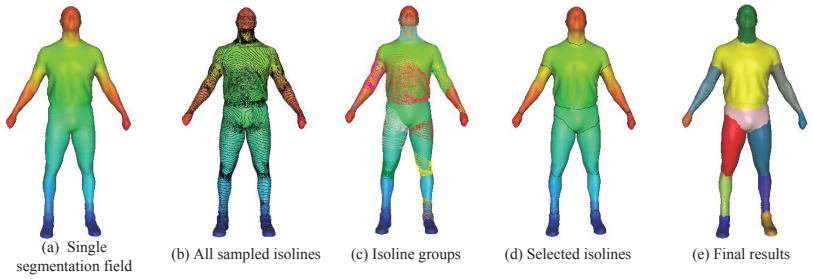


Figure 11: Isoline grouping and selection to obtain the final segments: (a) the generated single segmentation field representation, (b) all the isolines directly sampled from (a), (c) the isoline groups selected from (b) by Algorithm 1 and Algorithm 2, (d) the selected isolines using the cut criterion, and (e) the final segments.

is ordered by its length. Fig. 11 shows the results of this step, where Fig. 11(c) gives the isoline groups selected from all the sampled isolines in Fig. 11(b). It can be seen that most isolines are concentrated on potential cutting regions after filtering the noisy isolines (see Fig. 11(d)). It is also revealed that the isolines from the same group are actually candidates for a partition boundary (see Fig. 11(c)). As a result, high-quality partition boundaries will be selected by comparing the isolines in the same group as shown in Fig. 11(d).

5. Experimental results and discussion

Results and comparisons. We evaluate our method on a variety of meshes, covering all the models from the Princeton Segmentation Benchmark [11] and a number of other complex models collected from the Internet. Snapshots of some visual results are shown in Fig. 9 and Fig. 10. All the meshes are automatically segmented using the same fixed parameter setting. In general, our method obtains satisfactory segmentation results which are comparable to human perceptions. Fig. 9 demonstrates that our method can be applied to a variety of mesh categories including man-made objects and other nature objects. Besides, out method achieves acceptable results on complex models that have abundant details as shown in Fig. 10.

To evaluate the effectiveness of our method, we compare our experimental results with the classic algorithms of Shape Diameter [29], Random Cuts [7], Norm Cuts [7], Core Extra [35], Rand Walks [36] and Fit Prim [28]. Fig. 12 shows the quantitive histograms of the comparison results by adopting four different benchmark evaluation metrics of *cut discrepancy*, *hamming distance*, *rand index*, and *consistency error* that are proposed in [11]. It can be found that the proposed method performs better on all these four metrics on average.

Table 1 gives more comparison details over all the categories from the Princeton Segmentation Benchmark by adopting the Per-category Rand Index Error measure. Particularly, our method outperforms two recent state-of-the-art algorithms, namely, Isoline Cuts [9] and M-S method [8] in most categories. Comparing with the Isoline Cuts algorithm (see column ISO-Cuts in Table 1), our method performs better in most categories especially for meshes without obvious protrusions. The reason is potentially that the proposed single segmentation field well covers more effective concavity information through the optimized selections on the two hierarchies. Our method is also comparable to the recent M-S algorithm (see column M-S in Table 1). It can be found that our method performs better in most man-made object categories. This is probably because these kinds of meshes can be well described by fewer eigenvectors in our frame-

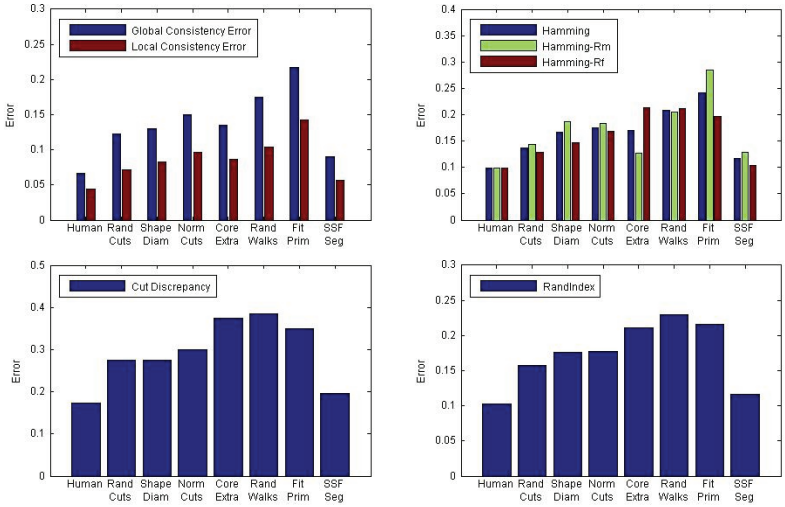


Figure 12: Comparison results with Manual Segmentation [11], and six automatic mesh segmentation algorithms of Random Cuts [7], Shape Diameter [29], Norm Cuts [7], Core Extra [35], Rand Walks [36], and Fit Prim [28]. We adopt four different benchmark evaluation metrics of *cut discrepancy*, *hamming distance*, *rand index*, and *consistency error* [11] for comparisons. The proposed single segmentation field (SSF) method is shown in the last column.

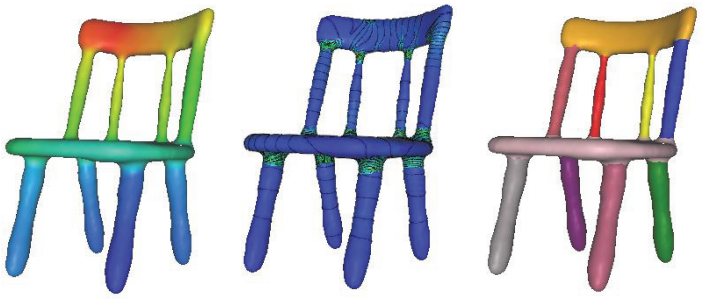
work, while clustering on a fixed number of eigenvectors may bring undesirable segments. Moreover, the proposed method is much easier to implement.

Table 1: More comparison details with Manual Segmentation [11], Random Cuts [7], Shape Diameter [29], SB19 ([6] with the training set over 90% in), SB6 ([6] with the training set dropping down to 30%), Isoline Cuts [9] and M-S method [8] over all the categories in the Princeton Segmentation Benchmark by adopting the Per-category Rand Index Error measure [11]. The proposed single segmentation field method is shown in the last column.

	Bench Mark	Rand cuts	Shape Diam	SB19	SB6	M-S	Iso Cuts	SSF Seg
Human	13.5	15.8	17.9	11.9	14.3	11.1	12.3	12.8
Cup	13.6	22.4	35.8	9.9	10.0	20.4	21.1	14.6
Glasses	10.1	9.7	20.4	13.6	14.1	9.4	9.8	11.3
Airplane	9.2	11.5	9.2	7.9	8.0	11.1	12.7	13.2
Ant	3.0	2.5	2.2	1.9	2.3	2.2	3.9	2.8
Chair	8.9	18.9	11.1	5.4	6.1	10.9	12.1	8.4
Octopus	2.4	6.7	4.5	1.8	2.2	2.5	4.1	2.6
Table	9.3	37.4	18.4	6.2	6.4	10.3	6.5	6.1
Teddy	4.9	4.5	5.7	3.1	5.3	3.2	5.3	3.6
Hand	9.1	9.7	20.2	10.4	13.9	7.9	11.5	11.0
Plier	7.1	10.9	37.5	5.4	10.0	8.9	7.3	8.5
Fish	15.5	29.7	24.8	12.9	14.2	29.6	24.3	21.5
Bird	6.2	11.4	11.5	10.4	14.8	9.4	9.7	7.8
Armadillo	8.3	8.1	9.0	9.0	8.4	8.7	10.6	9.1
Bust	22.0	25.1	29.8	21.4	33.4	25.1	24.4	28.6
Mech	13.1	28.3	23.8	10.0	12.7	13.1	12.2	12.6
Bearing	10.4	12.9	11.9	9.7	21.7	16.6	17.7	14.8
Vase	14.4	16.0	23.9	16.0	19.9	12.5	16.8	15.4
FourLeg	14.9	17.7	16.1	13.3	14.7	14.4	18.1	16.5
Average	10.3	15.7	17.6	9.4	12.2	12.0	12.7	11.6

Finally, we compared our method with the learning-based mesh segmentation algorithm in [6]. We notice that when the training set for each category is over 90%, their method achieves the best accuracy over most categories (see column SB19 in Table 1). However, when the size of the training set is reduced to 30%, our method outperforms it (see column SB6). Our method is also comparable with theirs when their training set is less than 60% (see more results

in [6]). In fact, method [31] can achieve even lower average rand index error (valued 8.8) than manual results via categorical learning with over 90% meshes. Since there exists difficulties in manually collecting enough mesh instances for each category in real-life applications, our method can be considered more suitable for the dataset that contains only a small number of meshes in each category, or when no training dataset is available.



(a) results from concavity-aware Laplaican operator

(b) results from cotangent weighting Laplacian operator

Figure 13: Comparison between our concavity-aware Laplacian operator and the classic cotangent weighting Laplacian operator. (Left) the segmentation fields, (Middle) the gradient maps with 50 uniformly sampled isolines, and (Right) segmentation results.

Our method is completely automatic, without requiring any predefined number of mesh segments or an extra mesh dataset for training. Moreover, our experiments showed that our method is time efficient. For a typical mesh with 100k triangles, the computation of our single segmentation field generally takes less than 10 seconds when running on an Intel 2.0GHZ laptop with 2GB memo-

ry. For the same mesh, boundary detection from the concise single segmentation field averagely takes less than 1 second.

Evaluations of the concavity-aware Laplacian operator. We evaluate our concavity-aware Laplacian operator by comparing it with the classic Laplacian operator using cotangent weighting scheme under our segmentation framework. Some results of the two operators are shown in Fig. 13. It can be found that our concavity-aware Laplacian operator leads to larger gradient magnitude and the sampled isolines mainly concentrate on the segmentation boundaries. Classic Laplacian operator results in irregularly distributed isolines that may lead to undesirable segmentation results.

Evaluations on the eigenvector hierarchy. For the eigenvector hierarchy, the main task is how to select a proper number of eigenvectors effectively and efficiently. Generally, a too small number of selected eigenvectors will potentially lead to the lost of sufficient details that are required for accurate segmentation, while a too large number of eigenvectors always increases the complexity of cuts selection and bring extra computation cost in optimization.

We use the derived eigenvector groups to select eigenvectors as discussed in Section 3.3. We find an interesting phenomena in our experiments that most of the meshes can be well segmented by utilizing the first two eigenvector groups, namely, altogether around 10 eigenvectors are sufficient to build the proposed single segmentation field for most meshes. Fig. 14 shows four examples to demonstrate this conclusion. For each mesh, we respectively select the first one eigenvector group and the first two eigenvector groups for comparison. The numbers of the adaptively selected eigenvectors are labeled below the meshes. The results show that all the four mesh examples can be well segmented by selecting a small number of eigenvectors since the desired boundaries on these meshes have been covered.

We also select eigenvectors using a group of fixed numbers as 5, 10 and 20 for comparisons. The results together with the adaptive selection method are shown in Fig. 15. The numbers of eigenvectors shown in the last column are dynamically selected by the first two eigenvector groups. We find that a

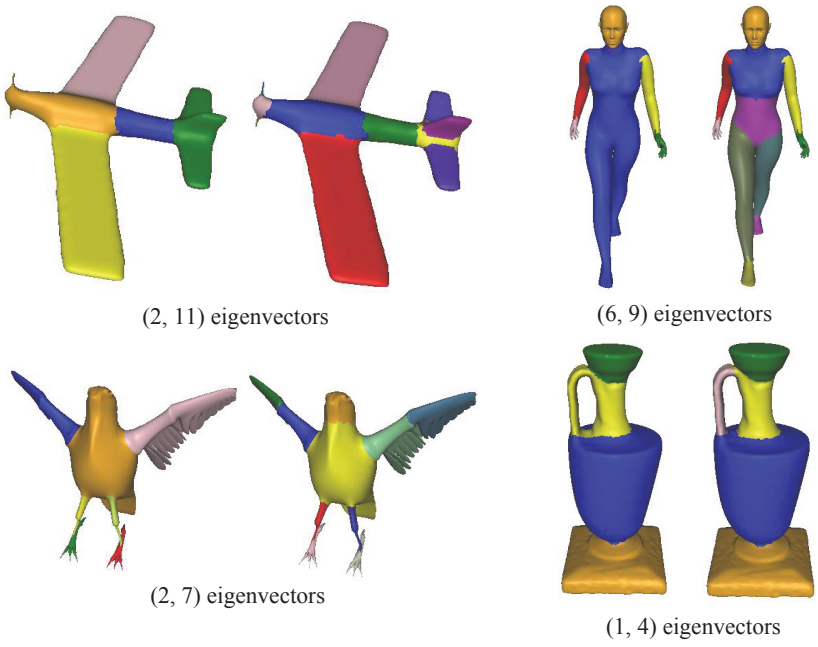


Figure 14: Evaluations of selecting different numbers of eigenvectors for automatic mesh segmentation. For the four pairs of models: mesh segmentation by using the eigenvectors in the first group (left), and in the first and second groups (right). The numbers of the selected eigenvectors are accordingly labeled below the meshes. It can be seen that all these mesh examples can be well segmented by selecting only a small number of eigenvectors since all the desired boundaries on these meshes have been recovered.

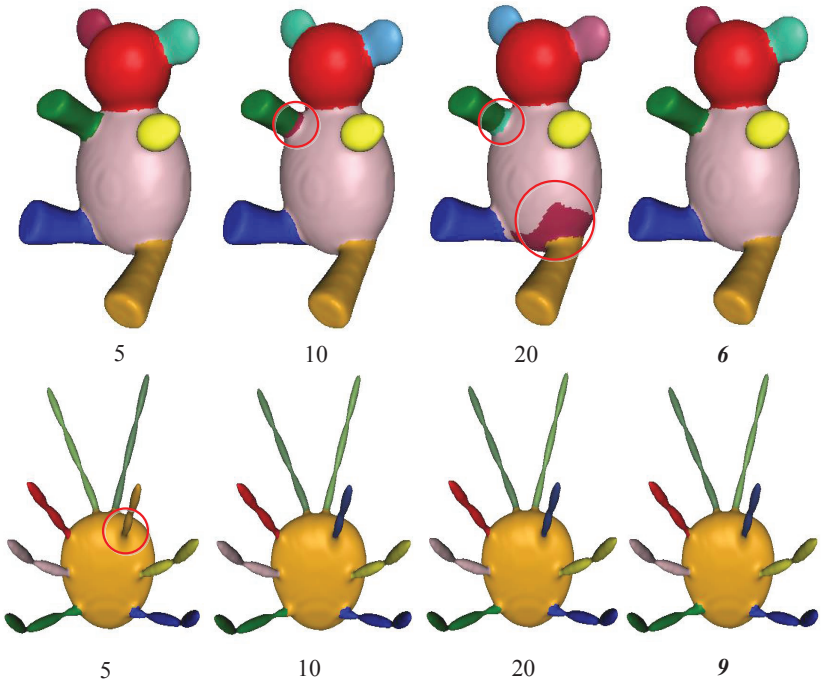


Figure 15: Comparisons by selecting a fixed number versus an adaptive number of eigenvectors for mesh segmentation. For each row: the first three meshes are the segmented results by selecting 5, 10 and 20 eigenvectors; while the last one is by selecting an adaptive number of eigenvectors. It can be seen that a predefined fixed number will potentially lead to either over-segmentation (see the red circles for either number 10 or number 20 in the first row), or insufficient segmentation (see the red circle for number 5 in the second row).

predefined fixed number will potentially lead to either over-segmentation or insufficient segmentation. That is, segmenting any mesh using a predefined number of eigenvectors is essentially unreasonable. We found that the adaptively selected eigenvectors do help in generating satisfactory results in most cases.

Evaluations on the sub-eigenvector hierarchy. We further evaluate the proposed weight scheme of IV and PO on the sub-eigenvector hierarchy. Fig. 16 gives the intermediate results for four example meshes after adopting the weight scheme of both IV and PO, only IV and only PO, respectively. We found that over-segmentation and inconsistency between the parts that are semantically

similar were be partially avoided (see the left column). As comparisons, the automatic segmentation results on different mesh types were not satisfactory by independently adopting IV (see the middle column) or PO (see the right column). These results verify our hypothesis that the two joint measures comprising IV and PO assist in discovering useful sub-eigenvectors for automatic segmentation in a complementary way.

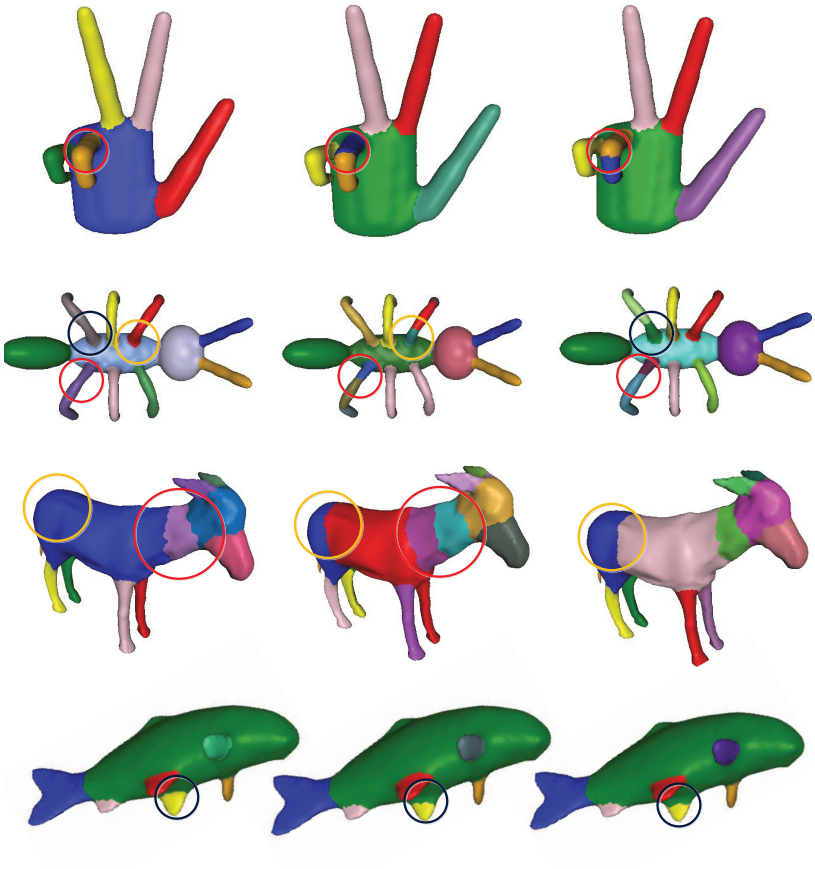


Figure 16: Evaluations of the inner variation and part oscillation measures on the sub-eigenvector hierarchy by four example meshes. For each mesh: (Left) adopting the joint weight scheme of both IV and PO, (Middle) only IV, and (Right) only PO.

Sensitivity to mesh tessellation. To evaluate the sensitivity of our method to the internal representations of the same mesh, we compared seg-

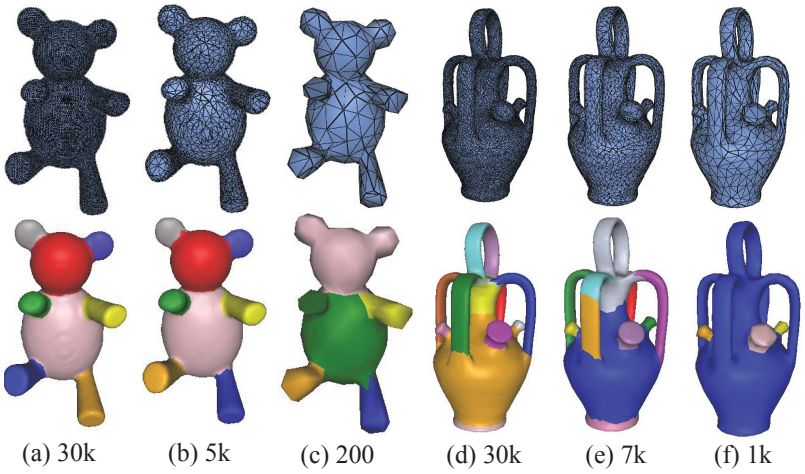


Figure 17: Segmentation results on the meshes with different tessellations. In each top-bottom pair: (top) the input mesh, and (bottom) the corresponding segmentation result.

mentation results on a group of meshes with different tessellations generated by the Meshlab software. The experimental results verify that the eigenvector variations on concave regions are largely insensitive to mesh tessellation. However, note that extreme tessellation variations may bring unpredictable results. For example, in Fig. 17, the *bear* mesh may be incorrectly segmented after simplifying the original mesh from 30k (Fig. 17(a)) to 500 faces. Additionally, the separation of too close segmentation boundaries of complex meshes, e.g., the *vase* mesh in Fig. 17, needs more precisely concavity modeling with sufficient faces. A simplified mesh may affect the field quality around these close boundaries, leading to the missing of certain handles (Fig. 17(e)(f)). This is because the detection of concavity information in such a simplified mesh would be inaccurate, which further weakens the effectiveness of eigenvectors by using our concavity-aware Laplacian operator.

Thresholds. Thresholds are required by any automatic mesh segmentation algorithm. In our method, we use $sth = \{T_c, T_l, T_n\}$ to measure the similarity between isolines, and use $dth = \{T_d\}$ to measure the density of isolines. We give empirical values we used to set these thresholds automatically. We set the

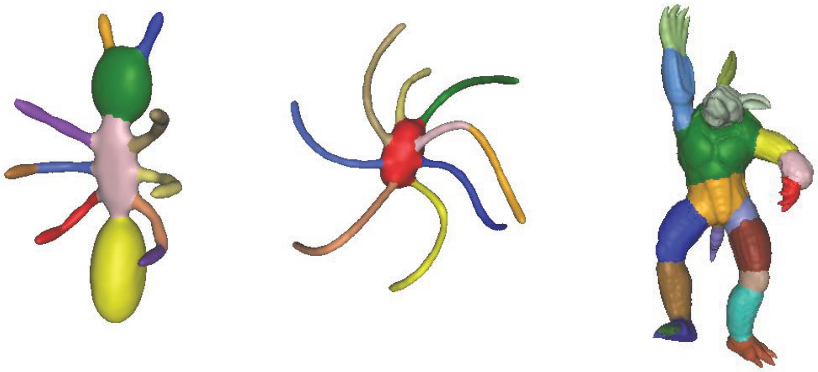


Figure 18: Some failure cases of our algorithm.

center distance threshold T_c to $1.5\bar{e}$, length ratio threshold T_l to 1.5, and normal angel threshold T_n to $acos(0.9)$. We sample 250 isolines per-model for boundary detection and thereby T_d is set to 2% according to this sampling number since models generally have less than $50 = 1/0.02$ desirable parts.

Limitations. The experimental results show that our method can automatically detect the expected boundaries even for regions without sufficient concavity characteristics. However, our method is easily misled and thereby generates inconsistent segmentation results for the regions that have too large concavities. For instance, the severely bent leg of the *ant* mesh and the *octopus* in Fig. 18 are further segmented into undesirable parts. This is essentially caused by the lack of mesh semantics since only mesh geometries are considered in our method. Another inconsistency failure case is the *Armadillo* mesh in Fig. 18, where its left ear is missed. It can be explained by the fact that our eigenvector analysis can not well distinguish whether a high frequency component describes a perceptually significant part or a local noise on a mesh.

6. Conclusion

We propose a fully automatic mesh segmentation method in this paper. After building concavity-aware Laplacian, our method exploits hierarchical analysis to discover the relationship between desirable segmentation boundaries on a

mesh and the algebraic properties of its spectral components. A novel single segmentation field is defined by aggregating the selected sub-eigenvectors from a Laplacian operator. Isolines are accordingly detected and grouped directly from the concise single segmentation field to generate segmentation boundaries automatically. The proposed framework is comparable to some recent state-of-the-art algorithms on the PSB benchmark and a number of complex meshes. Our future work includes improvements of the eigenvector selection strategy, and the use of our single segmentation field in other 3D applications.

Acknowledgments

The work described in this paper was supported by the Natural Science Foundation of China under Grant No. 61272218 and 61321491, the 973 Program of China under Grant No. 2010CB327903, Hong Kong Research Grant Council (Project Nos. GRF619611 and GRF619012), and the Program for New Century Excellent Talents under NCET-11-0232.

References

[1] T. A. Funkhouser, M. M. Kazhdan, P. Shilane, P. Min, W. Kiefer, A. Tal, S. Rusinkiewicz, D. P. Dobkin, Modeling by example, ACM Trans. Graph. 23 (3) (2004) 652–663.

[2] M. Zöckler, D. Stalling, H.-C. Hege, Fast and intuitive generation of geometric shape transitions, The Visual Computer 16 (5) (2000) 241–253.

[3] R. Liu, H. Zhang, Segmentation of 3d meshes through spectral clustering, in: Pacific Conference on Computer Graphics and Applications, 2004, pp. 298–305.

[4] R. Liu, H. Zhang, Mesh segmentation via spectral embedding and contour analysis, Comput. Graph. Forum 26 (3) (2007) 385–394.

[5] C. Chuon, S. Guha, Surface mesh segmentation using local geometry, Sixth International Conference on Computer Graphics, Imaging and Visulization (2009) 250–254.

[6] E. Kalogerakis, A. Hertzmann, K. Singh, Learning 3d mesh segmentation and labeling, ACM Trans. Graph. 29 (4).

[7] A. Golovinskiy, T. A. Funkhouser, Randomized cuts for 3d mesh analysis, ACM Trans. Graph. 27 (5) (2008) 145.

[8] J. Zhang, J. Zheng, C. Wu, J. Cai, Variational mesh decomposition, ACM Trans. Graph. 31 (3) (2012) 21.

[9] O. K.-C. Au, Y. Zheng, M. Chen, P. Xu, C.-L. Tai, Mesh segmentation with concavity-aware fields, IEEE Trans. Vis. Comput. Graph. 18 (7) (2012) 1125–1134.

[10] H. Zhang, O. van Kaick, R. Dyer, Spectral mesh processing, Comput. Graph. Forum 29 (6) (2010) 1865–1894.

[11] X. Chen, A. Golovinskiy, T. A. Funkhouser, A benchmark for 3d mesh segmentation, ACM Trans. Graph. 28 (3).

[12] A. Mangan, R. Whitaker, Partitionning 3d surface meshes using watershed segmentation, IEEE Trans. Vis. Comput. Graph. 5 (4) (1999) 308–321.

[13] V. Jain, H. Zhang, O. van Kaick, Non-rigid spectral correspondence of triangle meshes, International Journal of Shape Modeling 13 (1) (2007) 101–124.

[14] M. Leordeanu, M. Hebert, A spectral technique for correspondence problems using pairwise constraints, in: ICCV, 2005, pp. 1482–1489.

[15] S. Dong, P.-T. Bremer, M. Garland, V. Pascucci, J. C. Hart, Spectral surface quadrangulation, ACM Trans. Graph. 25 (3) (2006) 1057–1066.

[16] W. Benjamin, A. W. Polk, S. V. N. Vishwanathan, K. Ramani, Heat walk: Robust salient segmentation of non-rigid shapes, Comput. Graph. Forum 30 (7) (2011) 2097–2106.

[17] K. Zhou, J. Synder, B. Guo, H.-Y. Shum, Isocharts: streth-driven mesh parameterization using spectral analysis, SGP'04 (2004) 45–54.

[18] V. Kraevoy, D. Julius, A. Sheffer, Shuffler: modeling with interchangeable parts, Tech. Rep. TR2006-09, Dept. of Computer Science, University of British Columbia.

[19] B. Chazelle, D. Dobkin, N. Shourhura, A. Tal, Strategies for polyhedral surface decomposition: an experiment study, Computational Geometry: Theory and Applications 7 (4-5) (1997) 372–342.

[20] M. Vieira, K. Shimada, Surface mesh segmentation and smooth surface extraction through region growing, Computer Aided Geometric Design 22 (8) (2005) 771–792.

[21] T. Srinark, C. Kambhamettu, A novel method for 3d surface mesh segmentation, Sixth International Conference on Computers, Graphics and Imaging (2003) 212–224.

[22] A. Shamir, A survey on mesh segmentation techniques, Computer Graphics Forum 27 (6) (2008) 1539–1556.

[23] E. Zuckerberger, A. Tal, S. Shlafman, Polyhedral surface decomposition with applications, Computers & Graphics 26 (5) (2002) 733–743.

[24] Y. Zhou, Z. Huang, Decomposing polygon meshes by means of critical points, Multimedia Modeling (2004) 187.

[25] D. Page, A. Koschan, M. Abidi, Perception-based 3d triangle mesh segmentation using fast marching watersheds, CVPR (2003) 27–32.

[26] J. Wang, Z. Yu, Geometric decomposition of 3d surface meshes using morse theory and region growing, International Journal of Advanced Manufacturing Technology 56 (9-12) (2011) 1091–1103.

[27] S. Katz, A. Tal, Hierarchical mesh decomposition using fuzzy clustering and cuts, ACM Trans. Graph. 22 (3) (2003) 954–961.

[28] M. Attene, B. Falcidieno, M. Spagnuolo, Hierarchical mesh segmentation based on fitting primitives, The Visual Computer 22 (3) (2006) 181–193.

[29] L. Shapira, A. Shamir, D. Cohen-Or, Consistent mesh partitioning and skeletonisation using the shape diameter function, The Visual Computer 24 (4) (2008) 249–259.

[30] Y. Zheng, C.-L. Tai, O. K.-C. Au, Dot scissor: A single-click interface for mesh segmentation, IEEE Trans. Vis. Comput. Graph. 18 (8) (2012) 1304–1312.

[31] H. Benhabiles, G. Lavoué, J.-P. Vandeborre, M. Daoudi, Learning boundary edges for 3d-mesh segmentation, Comput. Graph. Forum 30 (8) (2011) 2170–2182.

[32] Q. Huang, V. Koltun, L. Guibas, Joint shape segmentation with linear programming, ACM Transactions on Graphics 30 (6) (2011) 1–11.

[33] R. B. Lehoucq, D. C. Sorensen, C. Yang, ARPACK users' guide - solution of large-scale eigenvalue problems with implicitly restarted Arnoldi methods, Software, environments, tools, SIAM, 1998.

[34] J. Canny, A computational approach to edge detection, IEEE Trans. Pattern Anal. Mach. Intell. 8 (6) (1986) 679–698.

[35] S. Katz, G. Leifman, A. Tal, Mesh segmentation using feature point and core extraction, The Visual Computer 21 (8-10) (2005) 649–658.

[36] Y.-K. Lai, S.-M. Hu, R. R. Martin, P. L. Rosin, Fast mesh segmentation using random walks, in: Symposium on Solid and Physical Modeling, 2008, pp. 183–191.