STATISTICS, OPTIMIZATION AND INFORMATION COMPUTING
*Stat., Optim. Inf. Comput.*, Vol. **14**, August 2025, pp 789–807.
Published online in International Academic Press (www.IAPress.org)

# Improving Spectral Segmentation of 3D Meshes Using Face Patches

Fatma Khairy [1], Mohamed H. Mousa [2], Hamed Nassar[1,*]

[1] *Computer Science Department, Suez Canal university, Ismailia 41522, Egypt*
[2] *Computer Science and Artificial Intelligence Department, University of Jeddah, Jeddah 23890, Saudi Arabia*

**Abstract** A huge amount of research work has been devoted in recent years to segmentation of 3D meshes composed of planar triangular faces. In particular, spectral segmentation has had a fair share of this work because it is extremely faster than other segmentation techniques, especially those based on AI and machine learning. However, existing spectral segmentation techniques suffer from complex processing and heavy computation due to dealing directly with these faces. The present article is an attempt to address this issue by proposing an effective technique based on grouping the faces skillfully into higher-level structures called patches. Specifically, each patch is made of two neighbor faces, effectively cutting the number of low level structures processed by the segmentation technique into almost half. However, since the constituent mesh structures have changed from face to patch, the normal spectral segmentation methodology is altered to suit the new geometry. This alteration is reflected on the number of elements of both the eigenvectors and weight matrix, both reduced almost by 50%. We have validated the proposed technique by segmenting numerous 3D meshes from public repositories. The resulting segments are colored in order to distinguish visually between different parts of the same 3D mesh. The experimental results indicate, both visually and quantitatively, that the proposed technique matches the performance of the best state-of-the-art methodologies, but at about half the time and space cost.

**Keywords** 3D mesh, spectral segmentation, eigen vector, Laplacian matrix

**DOI:** 10.19139/soic-2310-5070-2515

## 1. Introduction

Basically, there are two *representation* types of *3D surfaces, Point cloud and polygonal meshes* [1]. Compared to point cloud data, meshes has many advantages, such as high-resolution image texture, sharp geometrical structure [1], higher efficiency and greater flexibility in capturing non-uniform shapes [2]. As such, meshes have become the standard for discretely representing 3D shapes. The problem, however, is that they, unlike images or point cloud data [3], have irregular connectivity, and therefore require careful attention to carry out key operations such as segmentation. Segmentation (also known as partitioning or decomposition) of 3D meshes is crucial for many applications, such as shape analysis, computer graphics, geometric modeling, computer vision, multimedia, digital shape reconstruction, medical imaging, shape compression, texture mapping, skeleton extracting and computer aided design. It basically means partitioning arbitrary 3D objects into constituent parts that are structurally meaningful [4]. It is an important step towards model understanding, and acts as a useful tool for different model processing applications, such as reverse engineering and modeling by example. It is also used to provide a high level representation of raw 3D data which is key for CAD, CAM and CAE applications. However, the non-uniformity of 3D meshes leads to irregularity in the mesh structure, making the segmentation of these meshes particularly challenging.

[*]Correspondence to: Hamed Nassar (Email: nassar@ci.suez.edu.eg), Computer Science Department, Suez Canal university, Ismailia 41522, Egypt.

ISSN 2310-5070 (online) ISSN 2311-004X (print)

There are many segmentation techniques based on a variety of theories, such as random walk [5] and *spectral clustering* [6]. With the increasing availability of mesh data, techniques based on machine learning, especially neural networks, have started to proliferate [7]. In this case, the segmentation method takes as input the vertices and edges of the surface mesh of an object, and at the output returns an integer value of the class for each vertex or edge of the provided mesh. Deep learning-based approaches for shape understanding and processing tasks have attracted considerable attention.

Some algorithms, especially those based on non-local shape features, seem to produce segmentation that closely resemble ones made by humans [8]. However, there is no one technique that is suitable for all types of shapes. It is evident that a successful segmentation depends on the application at hand and should take into account the requirements needed for the task. This means that a segmentation technique can produce meaningful results only if the corresponding features optimally support all criteria suitable for the chosen application.

*Semantic segmentation* is a promising approach for 3D scene understanding, especially important in the field of computer vision. It is typically implemented through convolutional neural network (CNNs) or transformers [9]. The procedure of 3D mesh semantic segmentation entails assigning distinct semantic labels to each triangle, thus dividing the entire scene into various categories. With 3D meshes, semantic segmentation is the classification of its constituent element into specific classes or categories. With point cloud data, it is an important way to obtain 3D geo-information [1]. Semantic segmentation is hampered by the irregular structure of 3D mesh data. Due to the complex geometry representation and lack of efficient utilization of image texture information, this technique is still immature, requiring further research [10]. The trouble with AI segmentation techniques in general is the long training time and to a lesser extent segmentation time. It has been reported [11] that training the AI model took 5 hours for a 3D mesh in the SHREC11 collection, and 12 hours for the human-body. In addition, the segmentation time is excessively larger than with mathematical methods such as spectral segmentation used in the present article.

Spectral clustering [6], the technique adopted in the present article, was first proposed in 2004 [12]. Given a set of mesh faces, an affinity matrix that encodes the likelihood of each pair of faces belonging to the same group is first constructed. Spectral methods then use selected eigenvectors [13] of the affinity matrix, or its closely related graph Laplacian, to obtain data representations that can be more easily clustered. Inspired by human perception, an algorithm segments the 3D mesh along concave regions, resulting in natural segmentation with smooth boundaries. It was shown that clustering in the embedding space set up by the leading eigenvectors of the normalized affinity matrix is easier than clustering with respect to the affinity matrix itself, provided that an appropriate number of eigenvectors are chosen. It was also shown that in the embedding space, there is less ambiguity in terms of face groupings. Thus the fuzzy region constructed therein is expected to be smaller than the one constructed from the original affinities. With spectral clustering one can obtain very good results on relatively clean meshes. Reinforcing the notion that fully automatic 3D mesh segmentation is difficult, choices for the number of clusters and the number of eigenvectors used in the embedding are usually ad-hoc and require manual intervention. At any rate, spectral segmentation is simple to implement, can be handled efficiently by standard linear algebra software, and often outperforms traditional clustering algorithms such as the k-means algorithm. At first glance spectral clustering appears somewhat mysterious, with no obvious indication of why it even works.

Admittedly, a colossal amount of research work has been devoted to 3D mesh segmentation. However, existing techniques for 3D shape segmentation suffer from complex geometry processing and heavy computation due to using low-level features and fragmented results emanating from the lack of global consideration. The present article is an attempt to address this point by introducing a novel technique aiming to reduce the computational cost of the segmentation. Specifically, we propose constructing "patches", out of the existing faces, in an effort to improve both the segmentation output and the computational cost. The purpose of this work is the segmentation of a 3D mesh composed of triangular planar meshes. By carefully choosing the faces of each patch, we show via extensive experimental work that the technique is both effective and efficient, dividing the segmentation time almost by 2. Experiments on standard datasets show that the proposed technique outperforms the state-of-the-art unsupervised methodologies and is comparable to the best supervised approaches. Final obtained parts are colored in order to distinguish between different parts of the 3D mesh.

The rest of this article is organized as follows. In the next section, we review recent related work, indicating that the proposed technique is novel. In Section 3, we present the technique. In Section 4, we present the experimental results and discussions. Concluding remarks and suggestions for future work are given in the last section.

## 2. Related work

As was mentioned above, considerable work has been devoted to the subject of 3D mesh segmentation. Early on, in [14], the authors present a variational mesh decomposition algorithm to partition a mesh into a prescribed number of segments. The algorithm extends the Mumford-Shah model to 3D meshes that contains a data term measuring the variation within a segment using eigenvectors of a dual Laplacian matrix whose weights are related to the dihedral angle between neighbor triangles and a regularization term measuring the length of the boundary between segments. On the other hand, in [15], the authors present an automatic mesh segmentation algorithm that exploits the shape concavity information. The method locates concave creases and seams using a set of concavity-sensitive scalar fields. These fields are computed by solving a Laplacian system with a concavity-sensitive weighting scheme.

Another early segmentation attempt [16] focused on pairwise affinities for spectral segmentation. It was an attempt to learning a full range of pairwise affinities gained by integrating local grouping cues for spectral segmentation. These pairwise affinities are then used to cluster all region nodes into visually coherent groups. Also, in [8], the authors present an automatic mesh segmentation framework that achieves 3D segmentation in two stages, hierarchical spectral analysis and isoline-based boundary detection. During the hierarchical spectral analysis stage, a segmentation field is defined to capture a concavity-aware decomposition of eigenvectors from a concavity-aware Laplacian. Specifically, a sufficient number of eigenvectors is first selected and simultaneously partitioned into sub-eigenvectors through spectral clustering. Furthermore, in [17], the authors study segmentation of 3D meshes using p-spectral clustering. They propose an approach to get the optimal segmentation of a 3D mesh as a human can perceive using the minima rule and spectral clustering. This method is fully unsupervised and provides a hierarchical segmentation via recursive cuts. They introduce a new concept of the adjacency matrix based on cognitive studies.

In [18], the authors present an Unsupervised Spectral Mesh Segmentation Driven by Heterogeneous Graphs. They introduce a spectral framework where local geometry affinities are coupled with surface patch affinities. A heterogeneous graph is constructed combining two distinct graphs: a weighted graph based on adjacency of patches of an initial over-segmentation, and the weighted dual mesh graph. The partitioning relies on processing each eigenvector of the heterogeneous graph Laplacian individually, taking into account the nodal set and nodal domain theory.

In [19], the authors present spectral-based mesh segmentation. This article focuses on the topological aspect (graph spectrum), rather than the geometrical aspect, of mesh segmentation, in the belief that this tool has not been fully exploited. They pre-process the mesh to obtain an edge-length homogeneous triangle set and its Graph Laplacian is calculated. They then produce a monotonically increasing permutation of the Fiedler vector (2nd eigenvector of Graph Laplacian) for encoding the connectivity among part feature sub-meshes.

In [20], the authors present 3D mesh segmentation by region growing based on discrete curvature. The curvature value is computed for each vertex of the mesh then labeled with a value belonging to a curvature class. An iterative and recursive region growing process builds regions of neighbor vertices belonging to the same curvature class and starting from a seed vertex. Then, another iterative merging process eliminates poor regions, and particularly merges neighbor regions with the same curvature class.

In [21], a spectral segmentation method for large meshes is given. Building on edge collapse operators and progressive mesh representations, they first devise a feature-aware simplification algorithm that can generate a coarse mesh which keeps the same topology as the input mesh and preserves as many features of the input mesh as possible. Also, in [22], the authors perform 3D shape segmentation using a Medial Axis Transform (MAT) of the input shape. Specifically, they use the geometrical and structural information encoded in the MAT, they develop a simple and principled approach to effectively identify the various types of junctions between different parts of a 3D shape.

In [23], the authors study spectral mesh Segmentation via $\ell_0$ gradient minimization. Based on the local geometric and topological information of a given mesh, they build a Laplacian matrix whose Fiedler vector is used to characterize the uniformity among elements of the same segment. Specifically, they use the Fiedler vector to reformulate the mesh segmentation problem as a gradient minimization problem. Also, in [24], the authors study mesh segmentation using feature-aware region fusion. They introduce a segmentation algorithm for 3D meshes, consisting of two stages: over-segmentation and region fusion. In the first stage, adaptive space partition is applied to perform over-segmentation, which is very efficient. In the second stage, they define a new intra-region difference, inter-region difference, and fusion condition with the help of various shape features and propose an iterative region fusion method.

In addition, in [2], the authors study learning on 3D meshes with Laplacian encoding and pooling. Unlike images, 3D meshes have irregular connectivity, requiring careful design to capture relations in the data. To utilize the topology information while staying robust under different triangulations, they propose to encode mesh connectivity using Laplacian spectral analysis, along with mesh feature aggregation blocks (MFABs) that can split the surface domain into local pooling patches and aggregate global information among them.

Geometric deep learning has sparked a rising interest in computer graphics to perform shape understanding tasks, such as shape classification and semantic segmentation. When the input is a polygonal surface, one suffers from the irregular mesh structure. For example, in [7], the authors introduce Laplacian2mesh: Laplacian-based mesh understanding. Motivated by the geometric spectral theory, they introduce Laplacian2Mesh, a novel and flexible CNN framework for coping with irregular triangle meshes (where vertices may have any valence). Also, in [25], the authors present new designs of graph convolutional neural networks (GCNs) on 3D meshes for 3D object segmentation and classification. They use the faces of the mesh as basic processing units and represent a 3D mesh as a graph where each node corresponds to a face. To enhance the descriptive power of the graph, they introduce a 1-ring face neighborhood structure and derive novel multi-dimensional spatial and structural features to represent the graph nodes.

Furthermore, in [26], surface mesh segmentation based on geometry features is studied. The authors represent 3D geometry using signed distance function, and encode local geometric features to use them as an input for the CNN. The most important advantage of this work is its ability to learn from few examples. It is also able to handle thin and closely spaced surfaces, as it works with the geometry features and does not depend on the size of the surface mesh elements. In [27], the authors study a data-centric unsupervised 3D mesh segmentation method. The method uses a semi-supervised learning algorithm, to create vector embedding representations for each node in a 3D mesh graph. K-Means clustering is then used to cluster each node according to their node embedding information. K-means clustering is also used in [28], where mesh segmentation is implemented interactively according to the user requirements with the K value set as a control variable. In [29], the authors study developable mesh segmentation by detecting curve-like features on Gauss images. In [30], the authors present an open MVS-based texture reconstruction method based on the fully automatic plane segmentation for 3D meshes. They remedy the flaw caused by the fact that the Markov Random Field (MRF) energy function, provided by Open MVS-based 3D texture reconstruction algorithms, considers only the image label of the neighbor triangle face for the smoothness term and ignores the planar-structure information of the model. Furthermore, in [6], the authors study Part-to-surface mesh segmentation for mechanical models based on multi-stage clustering.

In [31], the authors propose a zero-shot method, called Segment Any Mesh (SAMesh), for 3D mesh segmentation that overcomes the limitations of shape analysis-based, learning-based, and current zero-shot approaches. SAMesh operates in two phases: multimodal rendering and 2D-to-3D lifting. However, we will show in the experimental results that our technique outperforms this method. In particular, considering the Rand index and cut discrepancy metrics, our patch technique results smaller errors than this method.

A recent paradigm in the field is semantic segmentation, where in each facet is assigned a semantic label [32]. It can be done with supervised machine learning, usually using CNNs, but in such case it requires large corpora of annotated training datasets. The problem with this is that in some domains, such as the geospatial domain, such datasets are quite scarce. But if such datasets are available, the paradigm gives impressive results. In [33], the authors present benchmark on semantic segmentation of high-resolution 3D point clouds and textured meshes from UAV LiDAR and Multi-View-Stereo. In [34], the authors perform 3D Semantic segmentation for Images and

Videos. They propose an active learning based 3D semantic labeling method for large-scale 3D mesh generated from images or videos. Taking as input a 3D mesh reconstructed from an image based 3D mesh, coupled with calibrated images, the method outputs a fine 3D semantic mesh in which each facet is assigned a semantic label. In [1], the authors study A Texture Integrated Deep Neural Network for Semantic Segmentation of Urban Meshes. In [9], the authors present a graph transformer for semantic segmentation of 3D meshes, motivated by the success of transformers in both NLP and computer vision, where they have achieved performance comparable to CNN models. Also, in [35], the authors present a deep learning framework, Laplacian Mesh Transformer to extract the critical structure and geometry features, for 3D mesh classification and segmentation.

Finally, for surveys on 3D mesh segmentation in general, one can consult [36], [37], [4], [10], [38], whereas for 3D mesh segmentation of point cloud data one can look at [39].

## 3. Basic algorithm and extensions

In this section we provide a description of the proposed technique. In the first step, create patches from the set of faces of the 3D mesh under consideration. Then, construct the corresponding affinity weight matrix $W^P$. Next, initial segmentation is used to create initial surfaces that will be used later in a heterogeneous graph, incorporating useful geometric relations. Lastly, spectral segmentation of the 3D mesh is carried out using the eigenvectors of $W^P$, which are based on the newly developed patches.

### *3.1. Patch generation*

Traditional segmentation techniques operate on individual faces of the mesh under consideration. The trouble with these techniques is that they can lead to high memory usage and excessive processing time, particularly with complex models containing thousands of faces. By grouping neighbor faces into patches, our technique effectively reduces the total number of elements considered in subsequent steps, allowing for a more efficient segmentation process. Each patch contains basically two neighbor faces. However, when a neighbor is not available to complete a patch, the latter will contain only one face. The move from faces to patches creates some challenges, which are addressed in the present article.

We introduce an efficient algorithm to generate patches from the given set of faces. The algorithm starts with a predetermined face then selects a neighbor face so that both faces form the first patch. The algorithm then moves to another face and repeats the same work. We will call a neighbor face that is available to be contained in a patch, a *free* neighbor. The algorithm keeps processing the faces, incorporating them into patches, until there are no more free neighbors. Clearly, as the algorithm keeps processing the mesh, the number of patches keeps increasing while the number of faces keeps decreasing.

Assume that the 3D mesh under consideration has $L$ faces, $f_1, f_2, \ldots, f_L$, and let $I = \{1, 2, \ldots, L\}$ denote the set of indices of these faces. Let $\widehat{n}_i$ be the normal unit vector of face $f_i$, and let

$$d_{i,j} = \left\|\widehat{n}_i - \widehat{n}_j\right\| \tag{1}$$

be the distance between faces $f_i$ and $f_j$. Let $N_i$ be the set of indices of the free neighbors of face $f_i$. Obviously, at any time during patch generation we will have $0 \le |N_i| \le 3$, where $|x|$ is the cardinality of set $x$. If face $f_i$ has $|N_i| \ge 1$, then its patch will contain two faces, whereas if it has $|N_i| = 0$, its patch will contain only one face—$f_i$ itself. Consequently, the algorithm ends up forming $M \ge \lceil L/2 \rceil$ patches if it processes a mesh with $L$ faces. In the sequel, patch $i$ will be considered a node and represented by a positive integer $p_i$ to identify the patch.

Once the mesh has been converted from a set of faces to a set $P$ of patches, henceforth referred to as nodes, we search for pairs of patch nodes which are neighbors. A pair of nodes, $i$ and $j$, are neighbors if they have a common edge, denoted by $\widehat{e}_{ij}$. Let $f_{p_i}$ and $f_{p_j}$ be the two faces, in the two neighbor nodes $p_i$ and $p_j$, respectively, which share the common edge $\widehat{e}_{ij}$. Then the mesh can then be represented by an undirected graph $G = (P, E)$, where $P = \{p_i\}$ is a set of the $M$ nodes and $E = \{\widehat{e}_{ij}\}$ is a set of the common edges between the neighbor nodes of the graph.

**Algorithm 1:** Patch Generation.

```
Input  : Set $F = \{f_1, f_2, \dots, f_L\}$ of $L$ faces ($F$ is a set of sets).
Output: Set $P = \{p_1, p_2, \dots, p_M\}$ of $M$ patches ($P$ is a set of sets).
1  $M = 0$
2  $I = \{1, 2, \dots, L\}$
3  while $I \neq \phi$ do
4      $M = M + 1$
5      Select an arbitrary index $i \in I$.
6      $N_i$: set of the indices of the free neighbors of face $i$.
7      $\widehat{n_i}$: normal unit vector of face $i$.
8      if $N_i = \phi$ then
9          $p_M = \{f_i\}$
10         $I = I \setminus \{i\}$
11     end
12     else
13         $j = \text{argmax}_{k \in N_i}(\|\widehat{n_i} - \widehat{n_k}\|)$ //Identify the index $j$ which maximizes the distance $d_{i,j}$.
14         $p_m = \{f_i, f_j\}$
15         $I = I \setminus \{i, j\}$
16     end
17 end
```

Define $l_{ij}$ as the length of edge $\widehat{e_{ij}}$ and $\bar{l}$ as the average length of all edges $\widehat{e_{ij}}$ for all $i$ and all $j$. Denote by $d_{i,j}$ the distance between these the two faces $i$ and $j$. Then, we can find the affinity parameter

$$x_{ij} = \begin{cases} \frac{c^2}{d_{i,j}^2} & \text{if } d_{i,j} \geq c \\ 1 & \text{otherwise} \end{cases} \tag{2}$$

where $c$ is a constant which will be 0.08 in the present article, as this value in particular has been found to yield a reasonable concavity threshold.

With the above in mind, we will now construct, for the graph, $G$, the patch-to-patch affinity matrix

$$W^P = \begin{bmatrix} w_{11}^P & w_{12}^P & \cdots & w_{1M}^P \\ w_{21}^P & w_{22}^P & \cdots & w_{2M}^P \\ \vdots & \vdots & \ddots & \vdots \\ w_{M1}^P & w_{M2}^P & \cdots & w_{MM}^P \end{bmatrix} \tag{3}$$

where

$$w_{ij}^P = \begin{cases} \frac{l_{ij} x_{ij}}{\bar{l}} & \text{if } p_i \text{ and } p_j \text{ are neighbor} \\ 0 & \text{otherwise} \end{cases} .$$

is the weight of edge $\widehat{e_{ij}}$ and represents the affinity between nodes $p_i$ and $p_j$.

### 3.2. Heterogeneous graph construction

A heterogeneous graph is one that contains different types of nodes and different types of edges. In the context of the present work, we aim to create a heterogeneous graph from the given graph by incorporating geometric relationships between patches and initial segmented regions. This approach improves the eigenvectors used in the final segmentation process.

The initial segmentation is produced using the $k$-means clustering algorithm [40] to generate $N$ regions, each a set of patches. Let $R = \{r_1, r_2, \ldots, r_N\}$ be the set of regions generated by the $k$-means algorithm, with each $r_i$ a node referring to the set of patches it contains.

An illustration of the resulting heterogeneous graph is shown in Figure (1), where there are $M + N$ nodes of two types, those representing patches and those representing regions, and 3 types of arcs, those representing patch-to-patch, those representing patch-to-region, and those representing region-to-region.

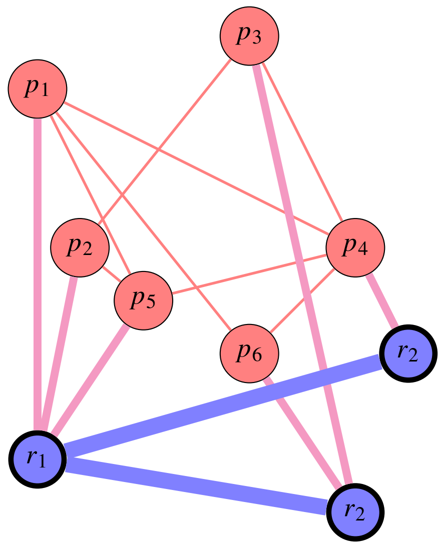


Figure 1. Heterogeneous graph model. Thin nodes represent patches and thick nodes represent regions. Thin, medium and thick lines are used to represent patch to patch, patch to region and region to region connections, respectively.

Now we are in a position to construct the region-to-region affinity matrix $W^R$. Using (3), we can write

$$W^R = \begin{bmatrix} w^R_{11} & w^R_{12} & \cdots & w^R_{1N} \\ w^R_{21} & w^R_{22} & \cdots & w^R_{2N} \\ \vdots & \vdots & \ddots & \vdots \\ w^R_{N1} & w^R_{N2} & \cdots & w^R_{NN} \end{bmatrix} \tag{4}$$

where

$$w^R_{mn} = \frac{\sum_{i \in r_m, j \in r_n} (w^P_{ij})}{L},$$

with $L$ being the number of edges common to the two neighbor regions $r_m$ and $r_n$.

Finally, using (3) and (4), we can construct for the entire heterogeneous graph, the $(M+N)\times(M+N)$ heterogeneous weight matrix

$$W^H = \begin{bmatrix} w^P_{11} & w^P_{12} & \cdots & w^P_{1M} & a_{11} & a_{12} & \cdots & a_{1N} \\ w^P_{21} & w^P_{22} & \cdots & w^P_{2M} & a_{21} & a_{22} & \cdots & a_{2N} \\ \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots \\ w^P_{M1} & w^P_{M2} & \cdots & w^P_{MM} & a_{M1} & a_{M2} & \cdots & a_{MN} \\ a_{11} & a_{21} & \cdots & a_{M1} & w^R_{11} & w^R_{12} & \cdots & w^R_{1N} \\ a_{12} & a_{22} & \cdots & a_{M2} & w^R_{21} & w^R_{22} & \cdots & w^R_{2N} \\ \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots \\ a_{1N} & a_{2N} & \cdots & a_{MN} & w^R_{N1} & w^R_{N2} & \cdots & w^R_{NN} \end{bmatrix} \tag{5}$$

where the $a_{ij}$ are non-zero if patch $i$ is in region $j$ and is zero otherwise, i.e.

$$a_{ij} = \begin{cases} c & \text{if} \quad p_i \in r_j \\ 0 & \text{otherwise,} \end{cases}$$

with $c = 0.01$ found [18] to yield a reasonable concavity threshold.

This completes the construction of the heterogeneous graph, which forms the foundation for the improved spectral segmentation.

### 3.3. Spectral segmentation

Now, we compute $K$ eigenvectors, arbitrarily labeled $U_1, U_2, \ldots, U_K$ and $K$ corresponding eigenvalues, $\lambda_1, \lambda_2, \ldots, \lambda_K$ eigenvalues. The value of $K$ is user defined, but we have found that $K = 10$ provides good segmentation results. The computation of the $U_i$ and $\lambda_i$ is attained by solving

$$L^H U_i = \lambda_i U_i \tag{6}$$

where the $ij$ element of $L^H$ is given by

$$l^H_{ij} = \begin{cases} \sum_{i=1}^{i=M+N} w^H_{ij} & \text{if } i = j \\ -w^H_{ij} & \text{otherwise} \end{cases} \tag{7}$$

are the elements of the unnormalized Laplacian $L^H$ of the heterogeneous affinity matrix $W^H$. When the eigenvectors are obtained, only the first $M$ elements of each eigenvector $U_i$ are considered, corresponding to the $M$ patches in the graph. It should be noted that each element $U_{i_j}$ of $U_i$ corresponds to a certain patch in the mesh.

Given a segment $s_k$, to divide it into two segments $g_1$ and $g_2$, using an eigenvector $U_i = (u_{i_1}, ui_2, \ldots, u_{i_M})$, we first identify the set $\Gamma$ of the elements of $U_i$ that correspond to the patches of segment $s_k$. Next we try to partition $\Gamma$ into two subsets $\Gamma_1$ and $\Gamma_2$ obtained by

$$g_{1,2} = \operatorname*{argmax}_{\Gamma_1,\Gamma_2} \sum_{j,k} |U_{i_{j\in\Gamma_1}} - U_{i_{k\in\Gamma_2}}|\theta, \tag{8}$$

where $\theta = 0$ if the patches corresponding to $j$ and $k$ are not neighbors, $\theta = \exp(-w^P(j,k))$ if the patches corresponding to $j$ and $k$ are neighbors and the common edge between them is concave, and $\theta = 0.01$ otherwise. The value 0.01 has been found to give good segmentation results.

After partitioning a segment $s_k$ into two segments $g_1$ and $g_2$, it should be ensured that the two segments meet these two segmentation conditions:

- Size condition: For a segment to be valid, its size, i.e. the number of patches it contains must be no less than a certain number, $\nu M$. That is, both $|g_1| > \nu M$ and $|g_2| > \nu M$ should hold. From experimentation, we have found that $\nu = 0.05$ provides good segmentation results. That is, each of the two segments produced by the partitioning must contain at least 5% of the patches of the entire 3D mesh.
- Connectedness condition: The patches in a segment must be all connected, i.e. neighbors to one another.

If the size condition is violated, i.e. one of the two segments produced by the partitioning contains less than $\nu M$ of the patches of the entire 3D mesh, the partitioning is discarded and the original segment $s_k$ remains intact for possible future segmentation by another eigenvector. An illustration of this is shown in Figure (2).

As for the violation of the connectedness condition, there can be two cases, assuming that the partitioning of the original segment $s_k$ has resulted in two segments, $g_1$ and $g_2$. The first case is when one of the two segments, say $g_2$, is unconnected, e.g. made up of two segments $g_{2_1}$ and $g_{2_2}$. Then we have one of two cases. The first case is when $g_{2_1}$ and $g_{2_2}$ satisfy the size condition. In this case, they will be recognized as two legitimate segments, and stored in the segmentation set as such. This case is shown in Figure (3). The second case is when $g_{2_1}$ and $g_{2_2}$ does not satisfy the size condition, in which case the partitioning will be canceled.

Algorithm (2) shows the segmentation process in pseudo code based on spectral analysis and the patch data structure. The Algorithm starts with a single segment, represented by a set $S$, which is basically the entire 3D mesh and thus contains all the patches. As the algorithm progresses, it uses the eigenvectors to iteratively create new segments. Each eigenvector is processed individually to create two new segments if certain conditions are met. Each eigenvector is used in two phases: the UseMin phase, where the minimum value of the vector is utilized, and the UseMax phase, where the maximum value of the vector is utilized.

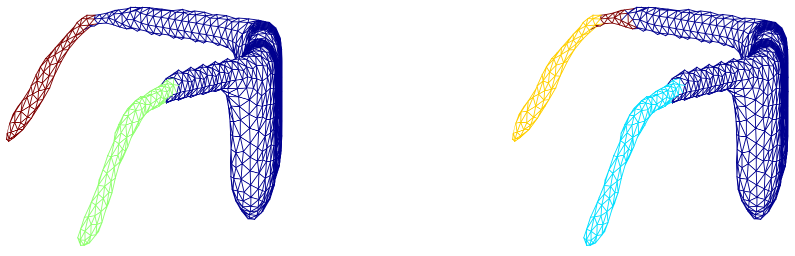

(a) (b)

Figure 2. Illustration of a failed segmentation attempt due to violating the size condition. (a) The blue segment $s_k$ is nominated for partitioning. (b) The segment $s_k$ has been partitioned into two segments, $g_1$ and $g_2$, but $g_1$ (brown segment) is too small (less than 5% of the 3D mesh) to meet the size condition, which results in discarding both segments.

The idea of spectral segmentation is to use the $K$ eigenvectors successively to partition a given 3D mesh iteratively. Each eigenvector will be used twice, once using the minimum element of the vector, henceforth called UseMin, and once using the maximum element of the vector, henceforth called UseMax. In the UseMin (resp. UseMax) of the eigenvector, the minimum (resp. maximum) element of the vector is located, the corresponding patch is identified, and the segment containing this patch is searched for and located. We always apply UseMin first and UseMax second, and in either one the located segment is partitioned using (8).

**Algorithm 2:** Spectral Segmentation.

```
Input  : Set $P = \{p_1, p_2, \dots, p_M\}$ of $M$ patches ($P$ is a set of sets)
         Set $U = \{U_1, U_2, \dots, U_K\}$ of $K$ eigenvectors ($U$ is a set of vectors)
Output: Set $S = \{s_1, s_2, \dots, s_n\}$ of $n$, $1 \leq n \leq 4K$, segments ($S$ is a set of sets)
1  $n := 1$
2  $s_n = P$
3  foreach $U_i \in U$ do
4      //UseMin Phase: Use the minimum value of eigenvector $U_i$
5      $j = \text{argmin}_{1 \leq k \leq M}(u_{i_k})$ //Identify the index $j$ of the minimum element of eigenvector
        $U_i = (u_{i_1}, u_{i_2}, \dots, u_{i_M})$.
6      Identify the index $k$ of the segment that contains patch $p_j$.
7      Partition $s_k$ into two segments, $g_1$ and $g_2$.
8      if $g_1$ and $g_2$ meet the size condition then
9          if $g_1$ and $g_2$ meet the connectedness condition then
10             $s_k = g_1$
11             $n = n + 1$
12             $s_n = g_2$
13         end
14         else
15             Assuming $g_1$ is unconnected, with two connected parts, each meeting the size condition, label
                the two parts $g_{1_1}$ and $g_{1_2}$.
16             $s_k = g_2$
17             $n = n + 1$
18             $s_n = g_{1_1}$
19             $n = n + 1$
20             $s_n = g_{1_2}$
21         end
22     end
23     //UseMax Phase: Use the maximum value of eigenvector $U_i$
24     $j = \text{argmax}_{1 \leq k \leq M}(u_{i_k})$ //Identify the index $j$ of the maximum element of eigenvector
        $U_i = (u_{i_1}, u_{i_2}, \dots, u_{i_M})$.
25     Identify the index $k$ of the segment that contains patch $p_j$.
26     Partition $s_k$ into two segments, $g_1$ and $g_2$.
27     if $g_1$ and $g_2$ meet the size condition then
28         if $g_1$ and $g_2$ meet the connectedness condition then
29             $s_k = g_1$
30             $n = n + 1$
31             $s_n = g_2$
32         end
33         else
34             Assuming $g_2$ is unconnected, with two connected parts, each meeting the size condition, label
                the two parts $g_{2_1}$ and $g_{2_2}$.
35             $s_k = g_1$
36             $n = n + 1$
37             $s_n = g_{2_1}$
38             $n = n + 1$
39             $s_n = g_{2_2}$
40         end
41     end
42 end
```

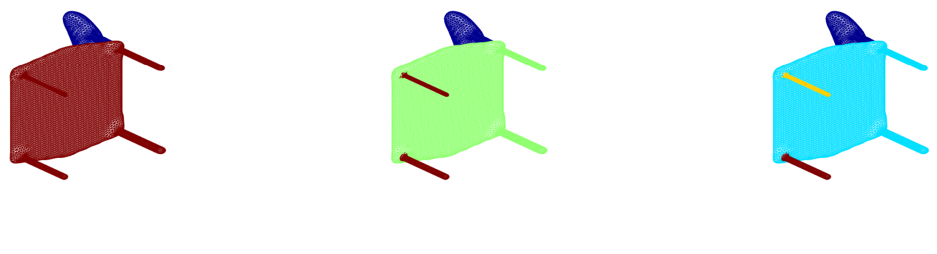

(a) (b) (c)

Figure 3. (a) The segment $s_k$, containing the seating area and the legs, is nominated for partitioning. (b) The partitioning produced two segments, $g_1$ and $g_2$, with $g_2$, the two legs, made up of two unconnected segments (the two legs). (b) After testing the size condition, each unconnected segment (i.e. each leg) came out to satisfy the condition, and therefore the three segments (the seating area and the two legs) will each be recognized as a segment.

Assuming the 3D mesh under consideration will eventually be segmented into $n, 1 \leq n \leq 4K$, segments, we define the segmentation set $S = \{s_1, s_2, \ldots, s_n\}$, where each $s_i$ is a segment made up of a set of patches. The set $S$ is initialized by setting $n = 1$, then setting $s_n = P$. This in effect initializes the segmentation set as having only one element, which is basically the entire 3D mesh.

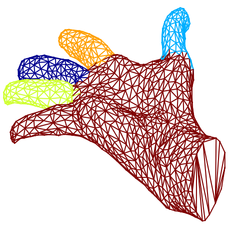
For Algorithm, in the first segmentation step, we set $s_1 = P$ and enter UseMin phase using $U_1$. However, we do not need to search for the segment to partition, since we have only one segment, $s_1$. Therefore, we set $k = 1$ and apply (8). Assuming that the partitioning produces two segments, $g_1$ and $g_2$, we investigate the two segmentation conditions. If both satisfy the two segmentation conditions, we set $s_k = g_1$, increment the segmentation set counter $n = n + 1$, then set $s_n = g_2$. This in effect, puts the first segment $g_1$ in place of the original segment which was partitioned, and inserts the other segment $g_2$ at the highest index of the segmentation set.

To continue the first segmentation step, we enter the UseMax phase for $U_1$. This means we search for the maximum element of $U_1$, and find the segment where the patch corresponding to that element is located. Since at this point we have only two elements of $S$, namely $s_1$ or $s_2$, the patch corresponding to the maximum element will be in either one, say $s_2$. Then, we set $k = 2$ and partition $s_k$ using (8). Assuming that we get from the partitioning two segments $g_1$ and $g_2$, then we first test if they meet the two segmentation conditions. If both come out to satisfy the two conditions, we set $s_k = g_1$, increment the segmentation set counter $n = n + 1$, then set $s_n = g_2$.

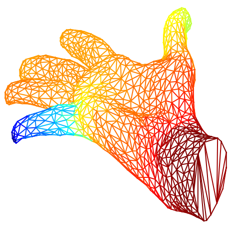
In the second segmentation step, we use $U_2$ entering UseMin first and UseMax second. Since the first step ended up with three segments, $s_1, s_2, s_3$, UseMax will choose for partitioning one of them, say $s_1$. Therefore, we set $k = 1$ and partition $s_k$ using (8). Assuming that the partitioning produces two segments, $g_1, g_2$, we then investigate if they meet the two segmentation conditions. Assuming that the two segments meet the size condition and $g_2$ is unconnected, with two connected parts $g_{2_1}$ and $g_{2_2}$, each meeting the size condition,we set $s_k = g_1$, increment the segmentation set counter $n = n + 1$, then set $s_n = g_{2_1}$, next increment the segmentation set counter $n = n + 1$, then set $s_n = g_{2_2}$. This in effect, puts the first segment $g_1$ in place of the segment $s_k$ which was partitioned, and inserts the other two segments $g_{2_1}$ and $g_{2_2}$ in the segmentation set, which at this point are $s_4$ and $s_5$.

To continue the second segmentation step, we enter the UseMax phase for $U_2$, to find the segment where the patch corresponding to the maximum element of $U_2$ is located. At this point we have five elements of $S$, namely $s_1, \ldots, s_5$, and hence the maximum element will select one of them, which will be handled as was done in the first segmentation step. Figure (4) shows the fine details of using a single eigenvector as a segmentation tool, where the two phases, UseMin and UseMax, are displayed. The steps repeat in the same way until all $K$ eigenvectors have been used, at which time the segmentation set $S$ will be completed, containing all segments of our 3D mesh.

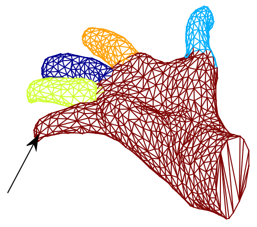
As can be seen, each of the two phases, UseMin and UseMax, can produce up to 2 new segments, making each eigenvector produce up to 4 new segments. However, the 2 new segments per phase occur only in exceptional cases when a newly produced segment turns out to have two unconnected parts which meet the two segmentation conditions. Accordingly, the final output of the algorithm will be $n \leq 4K$ segments of the original 3D mesh.



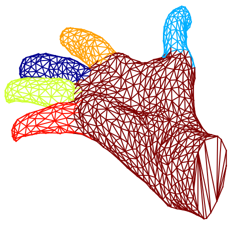
(a)



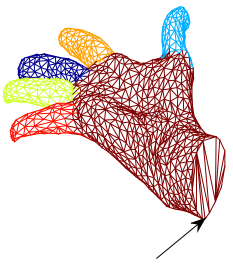
(b)



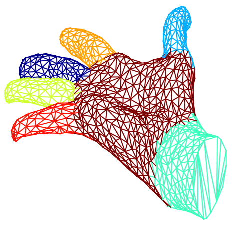
(c)



(d)



(e)



(f)

Figure 4. Segment creation using eigenvector. (a) 3D mesh segments before processing eigenvector $U_i$. (b) 3D mesh colored using eigenvector $U_i$. (c) UseMin phase: Arrow points to patch ( $p_j$) corresponding to the minimum value in $U_i$, brown segment represent the segment $s_k$. (d) By using cut condition $s_k$ is split into two segments $g_1$ and $g_2$, brown and black segments. (e) UseMax phase: Arrow points to patch ( $p_j$) corresponding to the maximum value in $U_i$, red segment represent the segment $s_k$. (f) Applying cut condition implies to split $s_k$ into $g_1$ and $g_2$.

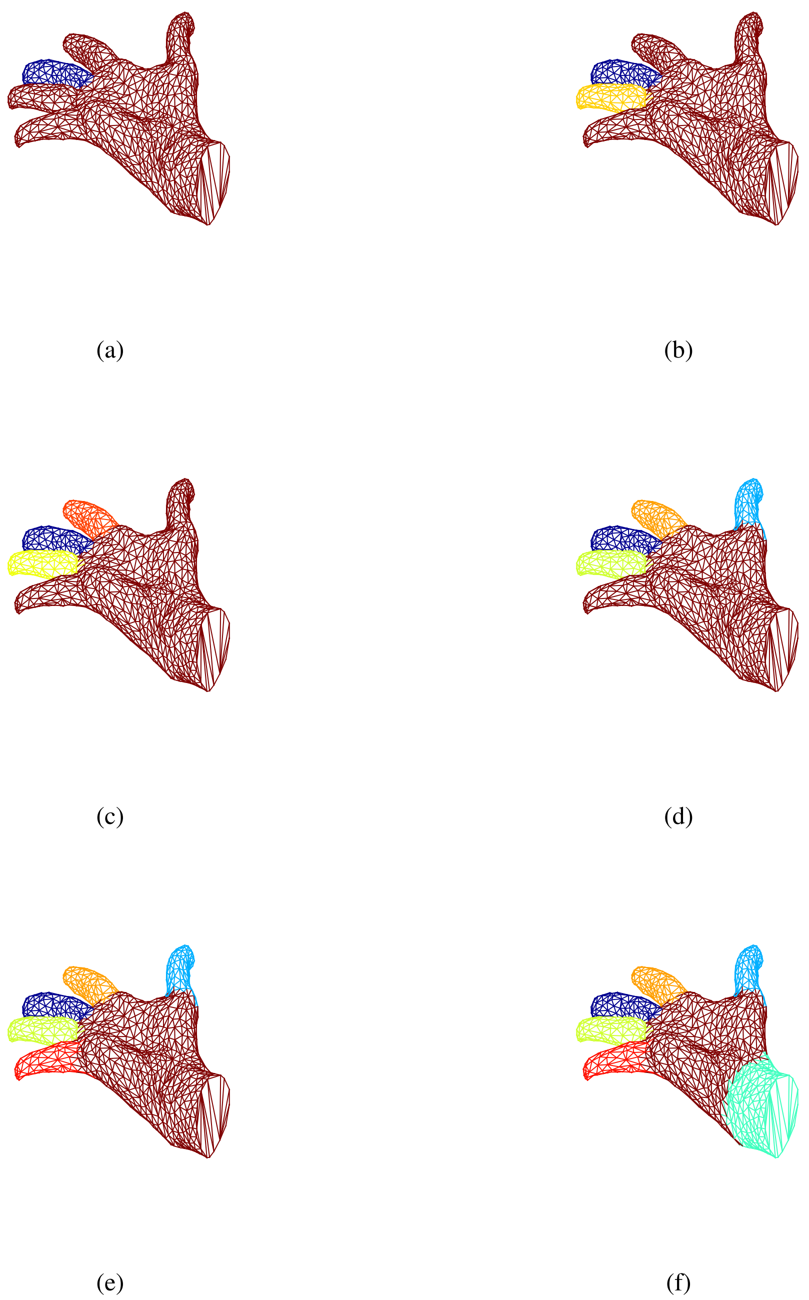


Figure 5. Hand segmentation sequence. (a) Middle finger segmented. (b) Ring finger segmented. (c) Index finger segmented. (d) Thumb segmented. (e) Little finger segmented. (f) Wrist segmented.

Figure (5) shows an example of segmenting a picture of a hand, displaying one new segment in each sub-figure. Figure (5a) shows the hand as two segments, the middle finger and the rest of the hand. Later sub-figures show the fingers appearing sequentially as the segmentation process progresses.

## 4. Experimental results and discussions

To validate the proposed patch technique, we coded both Algorithm 1 and Algorithm 2 in the MATLAB language, and then used the code to run experiments on 12 3D meshes from the Princeton [43] and COSEG [44] repositories. The experiments ran on a Windows 10 laptop powered by an Intel i7 CPU of 2.4 GHz, with 8 GB of main memory. The results indicate that the proposed technique provides segmentation quality that is no less than existing techniques, while outperforming these techniques in terms of both the time cost and the space cost. Figures (6) and (7) show the segmentation of two assortments of 3D meshes, from the Princeton and COSEG, respectively. It can be vividly seen that the technique has succeeded in partitioning each mesh into its distinguishable parts.

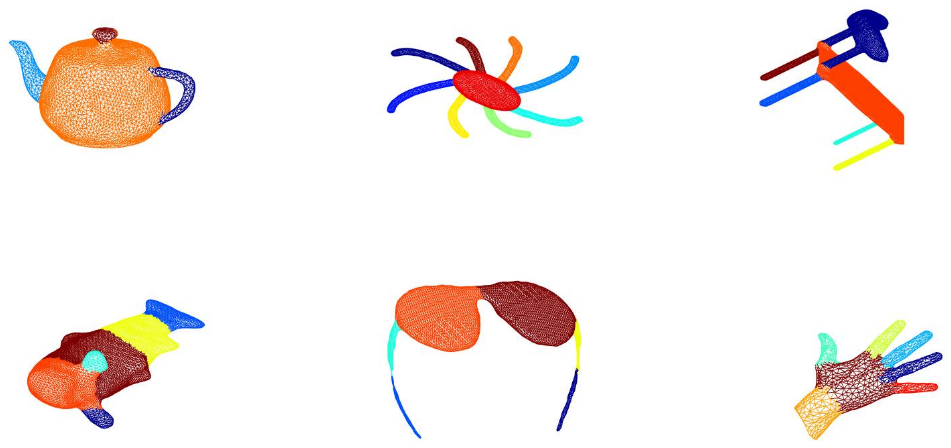


Figure 6. An assortment of 3D meshes from the Princeton segmentation repository [43] using the proposed technique.

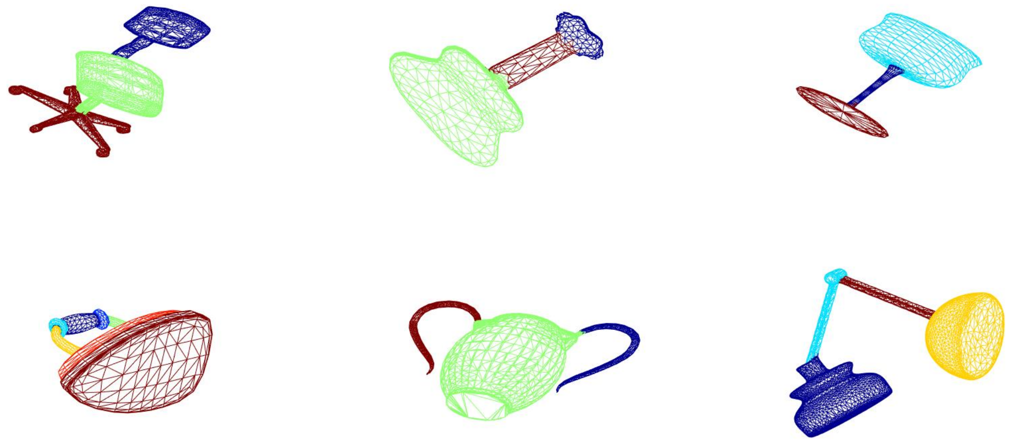


Figure 7. An assortment of 3D meshes from the COSEG repository [44] using the proposed technique.

Table 1. Numbers of faces and patches, as well patch creation time in seconds, for the 3D meshes of Figures (6) and (7).

| | Model | No of faces | No of patches | No of paired patches | No of unpaired patches | sec |
|---|---|---|---|---|---|---|
| Benchmark | Glassees | 12572 | 6440 | 6132 | 308 | 0.20 |
| | Chair | 24652 | 12443 | 12209 | 234 | 0.35 |
| | Octopus | 6198 | 3192 | 3006 | 186 | 0.09 |
| | Hand | 3026 | 1614 | 1412 | 202 | 0.05 |
| | Fish | 10496 | 5366 | 5130 | 236 | 0.16 |
| | Bearing | 3322 | 1716 | 1606 | 110 | 0.04 |
| COSEG | Chair | 9996 | 5406 | 4590 | 816 | 0.18 |
| | Guitar | 2328 | 1237 | 1091 | 146 | 0.05 |
| | Goblet | 1440 | 731 | 709 | 22 | 0.06 |
| | Iron | 2400 | 1241 | 1159 | 82 | 0.06 |
| | Vase | 7486 | 4067 | 3419 | 648 | 0.14 |
| | Lamp | 9996 | 5494 | 4502 | 992 | 0.21 |

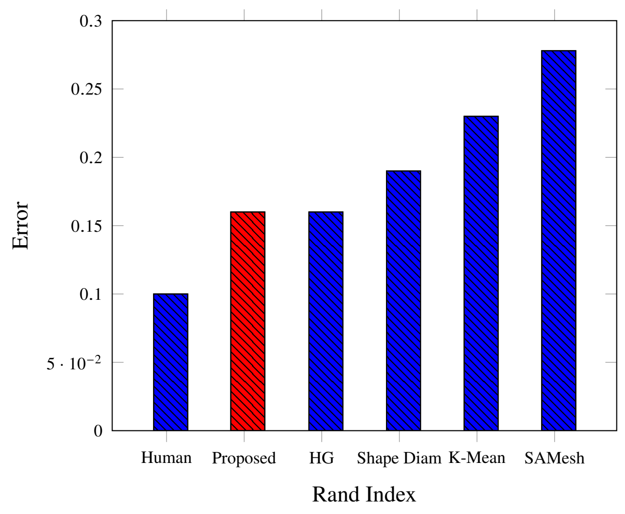
Table 2. Segmentation running times, in seconds, and eigenvector lengths for the 3D meshes of Figures (6) and (7), for the proposed technique (shown in **bold**) and the closest competitive technique.

| | | Segmentation time (s) | | EV length | |
|---|---|---|---|---|---|
| | Model | Proposed | HG [18] | Proposed | HG [18] |
| Benchmark | Glassees | **9.3** | 13.5 | **7771** | 14836 |
| | Chair | **159.1** | 335.9 | **8745** | 17306 |
| | Octopus | **6.9** | 10.4 | **1374** | 2682 |
| | Hand | **10.6** | 19 | **4826** | 9366 |
| | Fish | **38.2** | 79.8 | **5306** | 10428 |
| | Bearing | **9.6** | 16.6 | **1716** | 3322 |
| COSEG | Chair | **82** | 142.51 | **5406** | 9996 |
| | Guitar | **4.9** | 8.3 | **1237** | 2328 |
| | Goblet | **4.01** | 7.03 | **731** | 1400 |
| | Iron | **11.33** | 20.31 | **1241** | 2400 |
| | Vase | **47** | 84 | **4067** | 7486 |
| | Lamp | **75** | 118.77 | **5494** | 9996 |

Equally important to qualitative evaluation of segmentation, obtained visually, is quantitative evaluation of segmentation, obtained through measurements, especially concerning time and space requirements of the proposed technique. The experimental results show that these two requirements are around half of those of the existing face-based techniques.

Before introducing the time and space reduction results, we display in Table (1) the face and patch statistics for 3D meshes from 2 data sets, 6 from the the Princeton Segmentation Benchmark [43] and 6 from the Shape COSEG Dataset [44]. The table shows the number of original faces and the number of the patches created out of them. It is evident that the number of patches is around one half of the number of faces, which is the most significant achievement of the present work as this reduction is reflected on the number of elements of both the eigenvectors and weight matrix used in the spectral segmentation method. The last column of the table shows the time taken to convert faces into patches, and the values indicate that it is negligible (basically a fraction of a second), indicating that the proposed technique is not time costly.

Table (2) displays the segmentation running times, in seconds, and eigenvector lengths for the 12 3D meshes of Figures (6) and (7), for the proposed technique (shown in **bold**) and the closest competitive technique (HG [18]). Obviously, the proposed technique outperforms the competitive technique in all 12 3D meshes. In particular, for each 3D mesh the segmentation time of the proposed technique is almost half of the competitive technque.



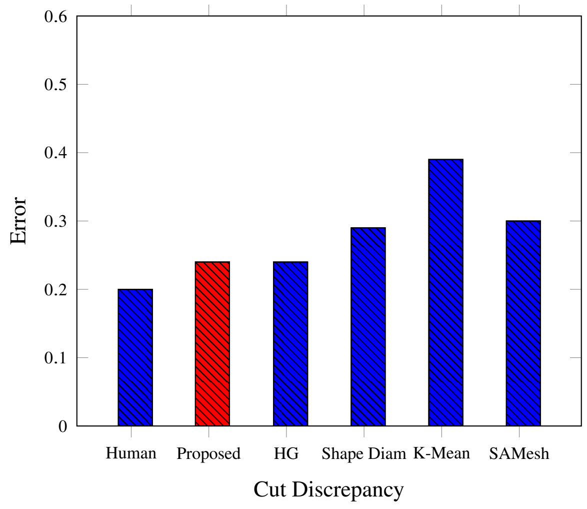


Figure 8. Comparison between the proposed technique and four other segmentation techniques, based on two error metrics, Rand Index and Cut Discrepancy Error, for the 6 3D meshes of Figure (6, which are taken from the Princeton Segmentation Benchmark [41]. Human segmentation, which the standard reference, is shown as the first column to the left.

Figure (8) shows a comparison between the proposed technique and four competitive segmentation techniques, based on two error metrics, Rand Index and Cut Discrepancy Error, for the 6 3D meshes of Figure (6, which are taken from the Princeton Segmentation repository [43]. The Rand index [41] is a metric that measures the likelihood that a pair of faces are either in the same segment or in two different segments in two segmentations. On the other hand, cut discrepancy [41] is a metric that sums the distances from points along the cuts in the computed segmentation to the closest cuts in the ground truth segmentation, and vice-versa. Intuitively, it is a boundary-based method that measures the distances between cuts. For both metrics, it can be seen that the proposed technique is as good as the best competitive technique, while greatly superseding the latter in terms of segmentation time, as displayed in Table (2).

Shown also in the Figure is the human segmentation, which the standard reference, as the first column to the left. It can be seen that the proposed technique gives values that are very close, equaling or superseding values provided by the four competitive techniques.

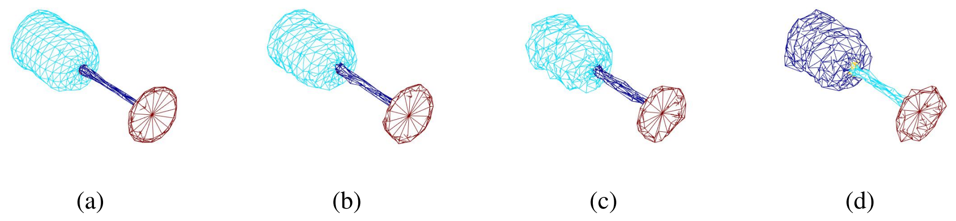


Figure 9. Illustration showing the resilience of the proposed technique to noise. Only when the noise parameter $\alpha$ exceeds 0.015 does the segmentation begin to suffer (tiny yellow segment appears at the top of the stem) (a)$\alpha$ = 0.005. (b)$\alpha$ = 0.01. (c)$\alpha$ = 0.015. (d)$\alpha$ = 0.017.

It should be noted that in all the above experiments, noise was neglected. However, to ensure that the robustness of the proposed technique, an extra experiment was carried out to investigate the effect of noise on the technique. Specifically, noise was inserted in the 3D mesh using the noise parameter $\alpha$, which changes the coordinates of all vertices, and then the proposed technique was applied to the noisy mesh. Initially, the noise parameter $\alpha = 0.005$ was used, then doubled, and finally tripled. For all these values of $\alpha$ the segmentation was successfully made by the technique, as shown in Figure (9). Only when the noise parameter reached the value $\alpha = 0.017$ did the technique suffer (little yellow segment at top of stem).

## 5. Conclusion

This study presents a novel technique to improve the performance of 3D mesh spectral segmentation using a patch-based geometrical structure. The proposed technique efficiently reduces the dimensionality of the spectral segmentation process by representing the mesh as a set of patches instead of faces. As a patch is nominally made of 2 faces, the dimensionality is reduced by almost 50%, resulting in a more computationally efficient affinity matrix construction and eigenvector size.

Our experimental results demonstrate significant reductions in both memory usage and segmentation time across a variety of 3D meshes, without compromising the segmentation quality. The technique's ability to reduce the length of the eigenvector highlights its scalability and adaptability to complex 3D meshes. The minute creation time of patches adds almost nothing to the overall segmentation process time cost.

The visual quality of the segmentation results is consistently accurate, with clearly defined boundaries that reflect the underlying structure of the 3D meshes. Additionally, . While our technique delivers improved results, it also opens avenues for future research. Adaptive patching techniques and further optimization for large-scale 3D meshes can be explored in future work to refine the approach and make it more versatile for a broader range of 3D meshes.

There are many venues to extend this work in the future. For example, the parameters $c = 0.08$ for affinity, $v = 0.05$ for size thresholds and $K$ for the number of eigenvectors of the affinity matrix are empirically chosen. A good point to study would be a theoretical analysis for both, employed to choose these parameters more justifiably. Another idea for future work is to consider massive geometries, instead of the normal geometries of this study. Obviously new algorithms tailored for modern architectures, e.g. parallel computing and quantum computing, will have to be developed to account for the gigantic computational work associated with massive geometries. In addition, the proposed technique deserves a meticulous theoretical analysis investigating it scalability. However, to do justice to such analysis, a separate study is planned for future work

## REFERENCES

1. Y. Yang, R. Tang, M. Xia, and C. Zhang, "A Texture Integrated Deep Neural Network for Semantic Segmentation of Urban Meshes," *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, vol. 16, pp. 4670–4684, 2023, doi:https://doi.org/10.1109/JSTARS.2023.3276977.
2. Yi-Ling Qiao, Lin Gao, Jie Yang, Paul L. Rosin, Yu-Kun Lai, and Xilin Chen. "Learning on 3D Meshes With Laplacian Encoding and Pooling." *IEEE Transactions on Visualization and Computer Graphics*, vol. 28, no. 2, pp. 1317–1327, Feb. 2022. doi:https://doi.org/10.1109/TVCG.2020.3014449. IEEE Educational Activities Department.
3. Waleed. Abuain, "Text-Line Segmentation Techniques for Arabic-Handwritten Documents: A Review," *Statistics, Optimization & Information Computing*, to appear in May 2025, DOI: https://doi.org/10.19139/soic-2310-5070-2466.
4. C. He and C. Wang, "A Survey on Segmentation of 3D Models," *Wirel. Pers. Commun.*, vol. 102, no. 4, pp. 3835–3842, Oct. 2018, doi:https://doi.org/10.1007/s11277-018-5414-1.
5. Yu Hou, Yong Zhao, and Xin Shan. "3D Mesh Segmentation via L0-Constrained Random Walks." *Multimedia Tools and Applications*, vol. 80, pp. 24885-24899, Apr. 2021. doi:https://doi.org/10.1007/s11042-021-10816-0. Springer.
6. A. Mu, Z. Liu, G. Duan, and J. Tan, "Part-to-Surface Mesh Segmentation for Mechanical Models Based on Multi-Stage Clustering," *Computer-Aided Design*, vol. 162, p. 103545, 2023, doi:https://doi.org/10.1016/j.cad.2023.10354510.1016/j.cad.2023.103545.
7. Qiujie Dong, Zixiong Wang, Manyi Li, Junjie Gao, Shuangmin Chen, Zhenyu Shu, Shiqing Xin, Changhe Tu, and Wenping Wang. "Laplacian2Mesh: Laplacian-Based Mesh Understanding." *IEEE Transactions on Visualization and Computer Graphics*, vol. 1, pp. 1–13, 2024. doi:http://dx.doi.org/10.1109/TVCG.2023.3259044. Institute of Electrical and Electronics Engineers (IEEE).
8. H. Wang, T. Lu, O. K.-C. Au, and C.-L. Tai, "Spectral 3D Mesh Segmentation with a Novel Single Segmentation Field," *Graph. Models*, vol. 76, no. 5, pp. 440–456, Sep. 2014, doi:https://doi.org/10.1016/j.gmod.2014.04.009.
9. G. Vecchio et al., "MeT: A Graph Transformer for Semantic Segmentation of 3D Meshes," *Computer Vision and Image Understanding*, vol.235, p.103773, 2023, doi = https://doi.org/10.1016/j.cviu.2023.103773, Publisher: Elsevier.
10. L. Gao et al., "Large-scale 3D Mesh Data Semantic Segmentation: A Survey," in *2023 9th International Conference on Big Data and Information Analytics (BigDIA)*, pp. 81–89, 2023, doi:https://doi.org/10.1109/BigDIA60676.2023.10429306.
11. Lahav, A. and Tal, A., *MeshWalker: deep mesh understanding by random walks*, *ACM Trans. Graph.*, 39(6), Article 263, December 2020. Association for Computing Machinery, New York, NY, USA. doi: https://doi.org/10.1145/3414685.3417806.

12. R. Liu and H. Zhang, "Segmentation of 3D Meshes through Spectral Clustering," in *12th Pacific Conference on Computer Graphics and Applications, 2004. PG 2004. Proceedings.*, pp. 298–305, 2004, doi:https://doi.org/10.1109/PCCGA.2004.1348360.
13. Y.Hu, Z.Wei, and G.Yuan, "Inexact Accelerated Proximal Gradient Algorithms For Matrix $\ell_{2,1}$-Norm Minimization Problem in Multi-Task Feature Learning," *Statistics, Optimization & Information Computing*, vol.2, no.4, pp.352–367, Nov.2014, DOI: https://doi.org/10.19139/soic.v2i4.106.
14. J. Zhang, J. Zheng, C. Wu, and J. Cai, "Variational Mesh Decomposition," *ACM Trans. Graph.*, vol. 31, no. 3, pp. 1–14, Jun. 2012, doi:https://doi.org/10.1145/2167076.2167079.
15. O. K.-C. Au, Y. Zheng, M. Chen, P. Xu, and C.-L. Tai, "Mesh Segmentation with Concavity-Aware Fields," *IEEE Transactions on Visualization and Computer Graphics*, vol. 18, no. 7, pp. 1125–1134, Jul. 2012, doi:https://doi.org/10.1109/TVCG.2011.131.
16. T. H. Kim, K. M. Lee, and S. U. Lee, "Learning Full Pairwise Affinities for Spectral Segmentation," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 35, no. 7, pp. 1690–1703, Jul. 2013, doi:https://doi.org/10.1109/TPAMI.2012.237.
17. M. Chahhou, L. Moumoun, M. El Far, and T. Gadi, "Segmentation of 3D Meshes Using p-Spectral Clustering," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 36, pp. 1687–1693, 2014, URL:https://api.semanticscholar.org/CorpusID:12834320.
18. P. Theologou, I. Pratikakis, and T. Theoharis, "Unsupervised Spectral Mesh Segmentation Driven by Heterogeneous Graphs," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 39, no. 2, pp. 397–410, Feb. 2017, doi:https://doi.org/10.1109/TPAMI.2016.2544311.
19. D. Mejia Parra, O. Ruiz, and C. Cadavid, "Spectral-Based Mesh Segmentation," *International Journal for Interactive Design and Manufacturing (IJIDeM)*, vol. 11, pp. 503–514, Aug. 2017, doi:https://doi.org/10.1007/s12008-016-0300-0 Publisher: Springer.
20. M. Y. Benzian and N. Benamrane, "3D Mesh Segmentation by Region Growing based on Discrete Curvature," in *2020 1st International Conference on Communications, Control Systems and Signal Processing (CCSSP)*, pp. 271–276, 2020, doi:https://doi.org/10.1109/CCSSP49278.2020.9151831.
21. X. Bao, W. Tong, and F. Chen, "A Spectral Segmentation Method for Large Meshes," *Communications in Mathematics and Statistics*, vol. 11, no. 3, pp. 583–607, 2023, doi:https://doi.org/10.1007/s40304-021-00265-4 Publisher: Springer.
22. Cheng Lin, Lingjie Liu, Changjian Li, Leif Kobbelt, Bin Wang, Shiqing Xin, and Wenping Wang. "SEG-MAT: 3D Shape Segmentation Using Medial Axis Transform." *IEEE Transactions on Visualization and Computer Graphics*, vol. 28, no. 6, pp. 2430-2444, June 2022. doi:https://doi.org/10.1109/TVCG.2020.3032566. IEEE.
23. W. Tong, X. Yang, M. Pan, and F. Chen, *Spectral Mesh Segmentation via $\ell_0$ Gradient Minimization*, IEEE Transactions on Visualization and Computer Graphics, vol. 26, no. 4, pp. 1807–1820, April 2020, doi:https://doi.org/10.1109/TVCG.2018.2882212.
24. L. Wu, Y. Hou, J. Xu, and Y. Zhao, "Robust Mesh Segmentation Using Feature-Aware Region Fusion," *Sensors*, vol. 23, no. 1, 2023, doi:https://doi.org/10.3390/s23010416.
25. Wenming Tang and Guoping Qiu. "Dense graph convolutional neural networks on 3D meshes for 3D object segmentation and classification." *Image and Vision Computing*, vol. 114, p. 104265, 2021. doi: https://doi.org/10.1016/j.imavis.2021.104265. Elsevier.
26. V. Dordiuk, M. Dzhigil, and K. Ushenin, "Surface Mesh Segmentation Based on Geometry Features," in *2023 IEEE Ural-Siberian Conference on Biomedical Engineering, Radioelectronics and Information Technology (USBEREIT)*, pp. 270–273, 2023, doi:https://doi.org/10.1109/USBEREIT58508.2023.10158888.
27. T. T. Sivri and Y. Sahillioğlu, "A Data-Centric Unsupervised 3D Mesh Segmentation Method," *The Visual Computer*, vol. 40, no. 4, pp. 2237–2249, 2024, doi:https://doi.org/10.1007/s00371-023-02913-y, Publisher: Springer.
28. J. Lian, H. Li, N. Li, and Q. Cai, "An Adaptive Mesh Segmentation via Iterative K-Means Clustering," in *Proceedings of 2021 Chinese Intelligent Systems Conference: Volume III*, pp. 193–201, 2022, doi:https://doi.org/10.1007/978-981-16-6320-8_20 Organization: Springer.
29. Z. Zeng, X. Jia, L. Shen, and P. Bo, "Developable Mesh Segmentation by Detecting Curve-Like Features on Gauss Images," *Computers & Graphics*, vol. 109, pp. 42–54, 2022, doi:https://doi.org/10.1016/j.cag.2022.10.003.
30. S. Li, X. Xiao, B. Guo, and L. Zhang, "A Novel OpenMVS-Based Texture Reconstruction Method Based on the Fully Automatic Plane Segmentation for 3D Mesh Models," *Remote Sensing*, vol. 12, no. 23, p. 3908, 2020, doi:https://doi.org/10.3390/rs12233908.
31. G. Tang, W. Zhao, L. Ford, D. Benhaim, and P. Zhang, "Segment Any Mesh: Zero-Shot Mesh Part Segmentation via Lifting Segment Anything 2 to 3D," 2024, doi: https://doi.org/10.48550/arXiv.2408.13679.
32. J. M. Adam et al., "Deep Learning-Based Semantic Segmentation of Urban-Scale 3D Meshes in Remote Sensing: A Survey," *International Journal of Applied Earth Observation and Geoinformation*, vol. 121, p. 103365, 2023, doi:https://doi.org/10.1016/j.jag.2023.103365.
33. M. Kölle, D. Laupheimer, S. Schmohl, N. Haala, F. Rottensteiner, J. D. Wegner, and H. Ledoux, "The Hessigheim 3D (H3D) Benchmark on Semantic Segmentation of High-Resolution 3D Point Clouds and Textured Meshes from UAV LiDAR and Multi-View-Stereo," *ISPRS Open Journal of Photogrammetry and Remote Sensing*, vol. 1, p. 100001, 2021, doi:https://doi.org/10.1016/j.ophoto.2021.100001.
34. M. Rong, H. Cui, Z. Hu, H. Jiang, H. Liu, and S. Shen, "Active Learning Based 3D Semantic Labeling From Images and Videos," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 32, no. 12, pp. 8101–8115, Dec. 2022, doi:https://doi.org/10.1109/TCSVT.2021.3079991.
35. X.-J. Li, J. Yang, and F.-L. Zhang, "Laplacian Mesh Transformer: Dual Attention and Topology Aware Network for 3D Mesh Classification and Segmentation," in *European Conference on Computer Vision*, pp. 541–560, 2022, doi:https://doi.org/10.1007/978-3-031-19818-2_31, Organization: Springer.

36. P. Theologou, I. Pratikakis, and T. Theoharis, "A Comprehensive Overview of Methodologies and Performance Evaluation Frameworks in 3D Mesh Segmentation," *Computer Vision and Image Understanding*, vol. 135, pp. 49–82, 2015, doi:https://doi.org/10.1016/j.cviu.2014.12.008.
37. M. Rashad, M. Khamiss, and M.-H. Mousa, "A Review on Mesh Segmentation Techniques," *International Journal of Engineering*, vol. 6, pp. 18–26, 2017, URL:https://api.semanticscholar.org/CorpusID:53681459.
38. R. Li and Q. Peng, "3D Shape Segmentation: A Review," *Recent Patents on Engineering*, vol. 16, no. 5, pp. 19–35, 2022, Publisher: Bentham Science Publishers, URL:https://doi.org/10.2174/1872212115666210203152106
39. D. Krawczyk and R. Sitnik, "Segmentation of 3D Point Cloud Data Representing Full Human Body Geometry: A Review," *Pattern Recognition*, vol. 139, p. 109444, 2023, doi:https://doi.org/10.1016/j.patcog.2023.109444.
40. U. Luxburg, "A Tutorial on Spectral Clustering," *Statistics and Computing*, vol. 17, no. 4, pp. 395–416, Dec. 2007, doi:https://doi.org/10.1007/s11222-007-9033-z.
41. X. Chen, A. Golovinskiy, and T. Funkhouser, "A Benchmark for 3D Mesh Segmentation," *ACM Trans. Graph.*, vol. 28, no. 3, pp. 73, Jul. 2009, doi:https://doi.org/10.1145/1531326.1531379.
42. Y. Wang, S. Asafi, O. van Kaick, H. Zhang, D. Cohen-Or, and B. Chen, "Active Co-Analysis of a Set of Shapes," *ACM Trans. Graph.*, vol. 31, no. 6, pp. 165, Nov. 2012, doi:https://doi.org/10.1145/2366145.2366184,
43. *A Benchmark for 3D Mesh Segmentation* . https://segeval.cs.princeton.edu. Accessed: 2025-01-01.
44. *The Shape COSEG Dataset*. https://irc.cs.sdu.edu.cn/~yunhai/public_html/ssl/ssd.htm. Accessed: 2025-01-01.