Segmentation of 3D Meshes through Spectral Clustering

Rong Liu Hao Zhang

GrUVi Lab, School of Computing Science
Simon Fraser University, Burnaby, BC, Canada
E-mail: *lrong, haoz@cs.sfu.ca*

Abstract

We formulate and apply spectral clustering to 3D mesh segmentation for the first time and report our preliminary findings. Given a set of mesh faces, an affinity matrix which encodes the likelihood of each pair of faces belonging to the same group is first constructed. Spectral methods then use selected eigenvectors of the affinity matrix or its closely related graph Laplacian to obtain data representations that can be more easily clustered. We develop an algorithm that favors segmentation along concave regions, which is inspired by human perception. Our algorithm is theoretically sound, efficient, simple to implement, and can achieve high-quality segmentation results on 3D meshes.

1. Introduction

Digital 3D models will become ubiquitous in imaging and multimedia applications in the near future. With the rapid advances in computer hardware and our never-ending pursuit for realism, highly detailed models obtained through 3D data capture have taken the dominant role over synthesized shapes. Highly affordable 3D digitizing devices have emerged which can produce 3D geometry data in the form of dense triangle meshes. Digital geometry processing is an active field of research in computer graphics, primarily aimed at developing signal-processing style concepts and algorithms to model, process, and analyze such mesh data.

Often as an early step in digital geometry processing, a triangle mesh is decomposed or segmented into a number of sub-parts. Proper segmentation of a 3D mesh into meaningful parts is useful in at least two aspects. First of all, a computation-intensive problem can often be solved more efficiently by a divide-and-conquer strategy. This is the general promise offered by the parallel computation paradigm, for which the problem of graph partitioning has received a great deal of attention [1, 8, 13, 14]. Here a network of processors are to be partitioned to achieve load-balancing while having only a small number of communication links between partitions. In spectral mesh compression [11] and watermarking [20], a large mesh is first decomposed into many small patches so that the cost of computing the eigenvectors becomes manageable. Secondly, the segmented parts can reveal useful structural properties of a 3D shape to facilitate further processing and analysis, e.g.,

- in geometric morphing applications [24] and for many higher-level vision-type tasks [3], mesh segments are used to establish shape correspondence;
- in mesh parameterization, the segmented mesh patches are preferably flat or they can approximate simple shapes such as cylinders or cones since this facilitates flattening of the patches to reduce stretching [26];
- in collision detection, a proper segmentation allows efficient bounding-volume computation [17];
- and finally, meaningful decomposition of a mesh also helps control skeleton extraction in animation [12].

Our initial interest in mesh segmentation comes from object recognition. That is, we wish to decompose a mesh into visually meaningful parts. To this end, we rely on the theory of *object recognition by parts* given by Hoffman and Richards [9] for our perception-based mesh segmentation. This theory stipulates that humans perform object recognition by first decomposing an object into several parts and that such a decomposition is derived from the so-called *minima rule*. The minima rule [9] states that human vision defines part boundaries along negative minima of principal curvatures, i.e., *concave creases*, on surfaces. Hoffman and Singh [10] point out further that the deeper the concavity, the more salient the part boundaries would be.

1.1. Related work

The use of the minima rule or deep concavity for 3D mesh segmentation is not new but rather recent [21, 24, 12]. Page et al. [21] propose a fast marching algorithm where the height map used for their watershed algorithm impedes climbing up negative principal curvature hills, honoring the

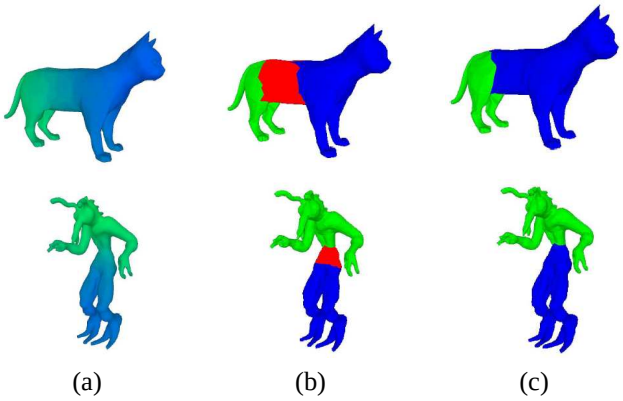
(a) (b) (c)

Figure 1. Segmenting an L-shaped object. (a) The base of the top part is surrounded by concavity (image from [21]) and both watershed and clustering can segment properly. (b) Watershed fails. (c) Our algorithm for 2-way clustering still works.

minima rule. Shlafman et al. [24] and Katz et al. [12] define pairwise distances between mesh faces in their clustering-based segmentation algorithms so that faces separated by deep concave regions are considered further apart and are less likely to be grouped into the same patch.

In contrast, the earlier watershed-based scheme by Magan and Whitaker [18] favors partition boundaries along high curvature regions and does not single out concavity. As many have pointed out [24, 12], one common problem with the watershed approach is over-segmentation. Even with a post-processing step involving patch merging, counterintuitive segmentation can still result. A more serious problem with watershed however is that although the height map definition can discourage the algorithm from crossing certain region, e.g., region of deep concavity, it is difficult to prevent the algorithm from using close-by, alternative route to just avoid the "danger zone" and flooding a region it is not supposed to venture into. This is illustrated in Figure 1.

Other mesh segmentation algorithms include convex decomposition [4], face merging in the context of mesh decimation [6], and skeletonization and space sweep [17]. Their pros and cons have been well documented by Katz et al. [12] and their work seems to be able to produce the best qualitative result for 3D mesh segmentation to date. This algorithm improves upon their earlier work [24] on K-means clustering on mesh faces. It first defines pairwise distances between mesh faces and iteratively refines a set of k representatives, one for each patch in the decomposition yet to be constructed, where k is determined using a heuristic. Each face is then assigned a probability of belonging to a patch for all patches. Thresholding these probabilities results in an initial grouping of the faces while leaving some number of ungrouped faces, those with fuzzy probabilities, in *fuzzy regions*, as shown in Figure 2. Precise partitioning over the fuzzy regions are constructed using a graph min-cut procedure applied to each pair of neighboring clusters in the initial grouping. This approach can be made hierarchical easily via recursion. It can also accommodate very large models by performing clustering on a simplified mesh and propagating the result to the full model.



(a) (b) (c)

Figure 2. Steps of fuzzy clustering [12] in the binary case (images from [12]). (a) Probabilities. (b) Fuzzy decomposition from initial grouping, where red indicates fuzzy region. (c) Final decomposition after graph min-cut.

Neither data clustering nor min-cut graph partitioning is a new problem. In fact, they are closely related and have received a great deal of attention in the computer vision [3, 23, 25], machine learning [2, 19, 22], graph drawing [14], and parallel computing literature [1, 8, 13]. One of the better known and effective approaches is spectral clustering, where the leading eigenvectors of an affinity or weighted graph Laplacian matrix are used to construct a *low dimensional embedding* for which the clustering problem is more easily solved.

Spectral embedding has been exploited by Gotsman et al. [7] recently to solve the spherical parameterization problem. But to the best of our knowledge, it has not been seriously considered for 3D mesh segmentation. The only mentioning of the use of spectral method for mesh segmentation we have found is in [12], where they implemented the normalized cut algorithm of Shi and Malik [23] and reported varied results. Note that this algorithm uses the second smallest generalized eigenvector of the graph Laplacian and it can be considered as a special case of the general spectral embedding approach. It has been shown that more eigenvectors tend to produce better clustering results [1].

1.2. Our approach

In this paper, we apply spectral clustering techniques to 3D mesh segmentation for the first time. We first construct an affinity matrix which encodes the likelihood of each pair of mesh faces belonging to the same mesh par-

tition. We utilize the distance matrix used by Katz et al. [12] to discourage faces separated by deep concave regions from being grouped together; this essentially models the minima rule [9]. Then an appropriate number k of the leading eigenvectors of the affinity matrix are computed. Since only a small number of eigenvectors are needed, we use ARPACK [15] for this task. These eigenvectors are used to obtain an embedding of the mesh faces onto the k-dimensional unit sphere. The remaining task involves straightforward K-means clustering on points embedded in the sphere, where we propose an effective heuristic to obtain a good starting point for faster convergence and avoidance of bad local minima.

In hindsight, while our approach performs K-means in the embedding space, the clustering algorithm of Katz et al. [12] can be seen as a variation of K-means with two main modifications. First of all, the distance metric used is no longer Euclidean but a combination of geodesic and angular distance over the mesh surface. Moreover, the cluster "centroid" is constrained to be one of the mesh faces — this is the patch representative being sought for. The use of spectral clustering can help accomplish two things:

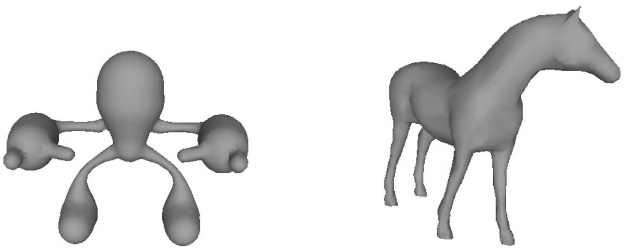
1. The nontrivial K-means clustering mentioned above is transformed into a clustering problem on the points distributed over the k-dimensional unit sphere, obtained by eigenvectors of the affinity matrix, for which a standard K-means algorithm is applicable.
2. With this transformation, the clustering problem becomes easier since the embedded data points tend to exaggerate the underlying group structures — this is implied by the so-called Polarization Theorem [2].

In Section 2, we discuss these issues in more details, showing from a theoretical point of view why spectral embedding and clustering can be expected to work well for 3D mesh segmentation.

1.3. Contribution and limitation of our approach

Our contribution is the realization of spectral clustering for 3D mesh segmentation, making aware of the great potential of this well-practiced approach to solve a relatively new problem and opening the door for further investigation. Through careful study of various existing spectral clustering techniques [1, 19, 22, 23, 25], which all vary slightly from each other, we arrive at an algorithm appropriate for mesh segmentation. Some advantages and features of our spectral clustering approach are:

- Efficiency: Computing a few leading eigenvectors is quite efficient using ARPACK. Nevertheless, it is well known that eigenvector/eigenvalue computations can be made significantly faster through parallelization or multilevel approaches [8, 14]. We would like to look into this in the near future. Also, we have used the full affinity matrix in this paper without thresholding. Since great speedup can be achieved if the matrix can be made sparse, we would also like to exploit this since for spectral clustering, the precision requirement on the eigenvectors is not very high [22].
- Simplicity: Compared to the fuzzy clustering approach of Katz et al. [12], our algorithm is much simpler to implement. There is no need for the sophisticated iterations or constrained graph min-cut, an eigensolver and an implementation of K-means clustering are sufficient. The whole algorithm can be programmed in Matlab with less than 100 lines of code. Our C++ implementation is also very compact.
- Spectral clustering can be made hierarchical by recursion and programmed to work for very large meshes with the aid of mesh simplification.



(a) Toy figure. (b) Horse.

Figure 3. Some meshes to be segmented.

Our current algorithm can produce high-quality mesh segmentation on reasonably "clean" models, i.e., models on which the visually salient segmentation boundaries are mostly around concave regions, e.g., the toy figure model in Figure 3(a). In these cases, our algorithm gives visually meaningful segmentation with smooth boundaries. The more difficult situation arises when the intended partition boundary, i.e., one that a human would pick, has to pass through a convex or otherwise featureless region. For example, consider the shoulder joint of a horizontally stretched arm or the leg joint of the horse model in Figure 3(b). At this point, our algorithm is not guaranteed to find the best boundary, especially when a greater portion of this boundary lies in featureless regions. Note that the fuzzy clustering approach of Katz et al. [12] would only work when the best boundary happens to be a min-cut; this is not always going to be the case however, as we shall explain later.

Finally, when the mesh model is noisy and the intended boundary is tampered with nearby concave edge sequences, the computed segmentation boundary would most likely be jaggy using either our approach or that of Katz et al. [12]. In this case, one may need to perform explicit smoothing or shortening of the boundary, e.g., using 3D snakes [16].

1.4. Organization of the paper

In the next section, we present some theoretical underpinnings of spectral embedding and motivate its use for clustering. In Section 3, we give a detailed description of the spectral mesh segmentation algorithm we use and discuss how the various parameters can be chosen. In Section 4, we demonstrate the effectiveness of spectral clustering for mesh segmentation through numerous examples and provide some timing analysis. We also illustrate some limitations of our current algorithm. Finally, we conclude in Section 5 and suggest possible future work.

2. Polarization: spectral clustering works

In this section, we outline the Polarization Theorem of Brand and Huang [2], thus motivating the use of spectral embedding for clustering and mesh segmentation. Consider a mesh with n faces that we would like to segment. Throughout this paper, we assume that the mesh graph is connected. If the mesh has more than one connected components, each component can be segmented separately. Suppose that we have constructed a symmetric affinity matrix $W \in \mathbf{R}^{n\times n}$, where $0 \leq W_{ij} \leq 1$ for all i and j. Thus W_{ij} can encode the likelihood that face i and face j can be clustered into the same patch. One may view the affinity matrix as the adjacency matrix of a complete (weighted) graph whose nodes are the mesh faces.

It is customary to normalize the affinity matrix W. The most obvious way to do this is to divide each affinity value W_{ij} by the total affinity, or vertex weight, at node i. If we denote by D the (diagonal) degree matrix whose i-th diagonal element is the sum of the i-th row of W, then the resulting normalized affinity matrix $L = D^{-1}W$; this is the matrix used by Shi and Malik [23] for their normalized cut algorithm. However, L is no longer symmetric in general. In order to obtain a symmetric normalized affinity matrix, which would possess more desirable properties, we use

$$N = D^{-1/2}WD^{-1/2}$$

in the spirit of spectral graph theory [5] and other spectral clustering approaches [19, 22, 25]. Note that then for each i and j, $N_{ij} = W_{ij}/\sqrt{D_{ii}D_{jj}}$.

Let $\mathbf{e}_1, \mathbf{e}_2, \ldots, \mathbf{e}_n$ be the orthonormal eigenvectors of N corresponding to eigenvalues $\lambda_1 \geq \lambda_2 \geq \ldots \geq \lambda_n$. Denote by Λ the diagonal matrix whose diagonal elements are the eigenvalues in descending order. Let $V \in \mathbf{R}^{n\times k}$ be formed by the k *eigenvalue-scaled* leading eigenvectors $\sqrt{\lambda_1}\mathbf{e}_1, \ldots, \sqrt{\lambda_k}\mathbf{e}_k$ of N. It is well known from linear algebra that the $n \times n$ matrix $Q = VV^T$, where V^T is the transpose of V, is the best rank-k approximation of N with respect to the Frobenius norm; equivalently, the most energy-preserving projection of N to rank k.

One may view the rows of V, each being a k-dimensional vector, as embeddings of the original data points into a k-dimensional feature space. Now we normalize each row of V to unit length to obtain a new embedding $\hat{V}$. Let $\hat{Q} = \hat{V}\hat{V}^T$, which we refer to as the *association matrix*. Clearly, the rows of $\hat{V}$, $\hat{\mathbf{v}}_1, \ldots, \hat{\mathbf{v}}_n$, give an embedding of the original data points into the k-dimensional unit sphere centered at origin. Moreover, $\hat{Q}_{ij} = \hat{\mathbf{v}}_i\hat{\mathbf{v}}_j^T = cos\theta_{ij}$ gives the cosine of the angle between unit vectors $\hat{\mathbf{v}}_i$ and $\hat{\mathbf{v}}_j$.

The Polarization Theorem states that as the matrix N is projected to successively lower rank k, the sum of squared angle-cosines $\sum_{i,j}(cos\theta_{ij})^2$ is strictly increasing. Algebraically, this implies that the cosine values will be more and more polarized towards $+1$ and -1. Geometrically, this suggests that pairs of embedded points on the unit k-sphere will either be clustered closer or repulsed further. Furthermore, a corollary of the theorem ensures that points of high affinity, i.e., with a higher probability of being grouped together, will move toward each other as k decreases, while other pairs will move more and more apart. This implies that clustering in the k-dimensional embedding space would be easier than clustering the original set of data points. In addition, clustering of the embedded points $\hat{\mathbf{v}}_1, \ldots, \hat{\mathbf{v}}_n$ can be accomplished by the standard K-means clustering simply using the Euclidean distance metric.

Our experiments confirm the polarization phenomena for entries in the association matrix Q compared with entries in the original affinity matrix N and entries in the higher-rank approximation of N. In practice however, we find the *eigenvalue-scaled* embedded coordinates $\hat{\mathbf{v}}_1, \ldots, \hat{\mathbf{v}}_n$ to work unfavorably for mesh segmentation. Rather, embedded coordinates obtained without scaling the eigenvectors by eigenvalues work better; these embeddings have also been used by several authors [19, 22, 25] for spectral clustering with success. The association matrices derived from these also exhibit polarization, as shown in Figure 4.

Note that both the toy figure and the sword are easy to cluster in the embedding space as the association matrix is almost perfectly polarized. The horse model, on the other hand, is more difficult to deal with, even in the embedding space. In all cases however, clustering in the embedding space is clearly easier than using the affinity matrix.

Finally, it is worth noting that only favoring a low-dimensional embedding because of its better polarization property is not always desirable for mesh segmentation. This is because the less number of eigenvectors to use, the larger the distortion associated with the embedded coordinates. This may result in poor segmentation since the association matrix would no longer be an accurate enough approximation of the affinity matrix. In the next section, we discuss how to choose the number of eigenvectors for embedding and clustering, along with our spectral mesh segmentation algorithm.

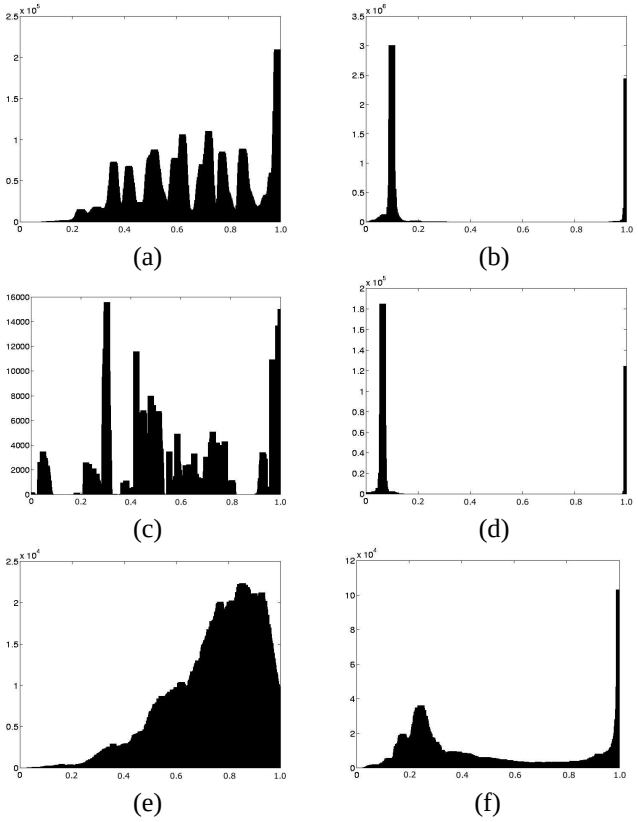


Figure 4. Histograms of entries in the affinity matrix vs. those in the association matrix. (a), (b): for the toy model in Figure 3(a). (c), (d): for the sword model given in Figure 5(e). (e), (f): for the horse model in Figure 3(b).

3. Spectral mesh segmentation

Spectral clustering takes as input an affinity matrix and uses its first k largest eigenvectors to compute an embedding and determine the grouping of mesh faces.

3.1. Affinity matrix

In order to partition a mesh along its edges, we group faces instead of vertices, i.e., we construct the affinity matrix with respect to the connectivity of the dual of the mesh graph. To perform intuitive segmentation, the affinity matrix W should encode mesh structural information which reflects how faces are grouped in accordance with human perception. The pairwise face distances used by Katz et al. [12] roughly model the minima rule and we use them to define the affinity matrix.

In the distance definition, both geodesic and angular distances between mesh faces are accounted for. There are two important parameters δ and η. The parameter δ is a real number between 0 and 1 controlling the relative importance of geodesic distance and angle distance for mesh segmentation. In most of our experiments, angle distance plays a more dominant role for visually meaningful segmentation. Hence, δ is set close to zero, e.g., $\delta \in [0.01, 0.05]$. A smaller value of η gives more weight to concavity in our segmentation. As we wish to exemplify the minima rule, we often set $0.1 \leq \eta \leq 0.2$.

Once the pairwise face distances are obtained, the affinity matrix is formed by a Gaussian kernel as

$$W(i,j) = e^{-dist(i,j)/2\sigma^2}.$$

Clearly, $0 < W(i,j) \leq 1$, and closer faces share larger affinities between them. Nonetheless, the use of a Gaussian introduces a practical problem of how to choose its width σ for proper segmentation. If σ is too small, it may separate faces which should have belonged to the same cluster. If σ is too large, faces that should be in different clusters may be falsely grouped together. In our experiments, we simply choose σ as an average, $\sigma = \frac{1}{n^2} \sum_{(1 \leq i,j \leq n)} dist(i,j)$, and have found it to work quite well in practice. Other weighting schemes for the distances and other choices of σ are possible, but to judge the usefulness of a particular choice for general mesh segmentation would require further study.

3.2. Spectral clustering for mesh segmentation

Generally speaking, all the spectral clustering algorithms proposed so far [2, 19, 22, 25] are intrinsically similar. They use the eigenvectors of the affinity matrix. These eigenvectors inherit information from the affinity matrix and enhance clustering of data points in the embedding space, as we have explained in Section 2. Our spectral mesh segmentation algorithm is given as follows.

1. Compute the affinity matrix W defined in Section 3.1.
2. Normalize W to $N = D^{-\frac{1}{2}} W D^{-\frac{1}{2}}$.
3. Compute the k largest eigenvectors $\mathbf{e}_1$, $\mathbf{e}_2$, ..., $\mathbf{e}_k$ of N and construct matrix $V = [\mathbf{e}_1, \mathbf{e}_2, ..., \mathbf{e}_k]$.
4. Compute $\hat{V}$ by normalizing each row of V to unit length.
5. Extract an initial guess of cluster centers for K-means, say, ω_1, ω_2, ..., ω_k, from $\hat{Q} = \hat{V}\hat{V}^T$, where k is the number of clusters desired.
6. Perform K-means clustering on the coordinates given by the rows of $\hat{V}$ to obtain grouping of corresponding mesh faces.

Steps 5 and 6 of our algorithm differ from previous ones [19, 22, 25]. Ng et al. [19] perform K-means clustering on $\hat{V}$ directly, but this can easily run into bad local minima, as our experiments can confirm. Weiss [25] suggests

to threshold the association matrix $\hat{Q}$ by first setting its entries that are sufficiently close to unity to 1. Then faces i and j are grouped if and only if $\hat{Q}_{ij} = 1$. But this procedure may lead to contradictory clustering results. Note that we can actually combine both techniques to get around their drawbacks. That is, we first threshold $\hat{Q}$ to obtain an initial guess for K-means clustering. The initial guess need not be really accurate yet it likely provides a good starting point for the subsequent K-means algorithm.

In practice however, thresholding would require an extra parameter and K-means is quite sensitive to the choice of the threshold. Therefore, we have elected to use a different heuristic to find an initial guess for K-means. Essentially, if we know the number c of clusters to segment the mesh into, we would use c embedded points that are *mutually furthest apart* as a good initial guess to start K-means clustering. These points can be found by a greedy algorithm which first locates the smallest entry in $\hat{Q}$, say, $\hat{Q}_{rs}$. Then r and s are furthest apart among all pairs of embedded points. The algorithm then iteratively finds the next point in a min-max fashion. That is, the new point is to minimize the maximum association from previously found points.

We use a similar scheme to determine the number of clusters, which relies on the affinity matrix. We note that there will typically be a point when adding a new representative face to the mix would dramatically increase the maximum affinity to previously included faces. This is when we stop adding faces and arrive at the number of clusters. In practice and as observed by others [19, 25], the number of eigenvectors chosen should be the same as the number of clusters. This way we have set the last parameter needed for our spectral mesh segmentation algorithm.

4. Experimental results

We have applied our algorithm to various 3D models and achieved some very good segmentation results; these are shown in Figure 5 and Figure 6. The face count for these meshes vary from about 50 to 4000. Evidently, our algorithm works quite well on relatively clean models, as we first stated in Section 1.3. Although we did not perform hierarchical decomposition, by changing the number of eigenvectors (same as the number of clusters), we see that the algorithm tends to segment in a hierarchical fashion. Note especially results for the toy and the sword.

The horse model, shown in Figure 6(e), is more challenging. The result we obtain is about the same as fuzzy clustering [12]. The bird example in Figure 6(h) illustrates the potential problem of placing too much emphasis on angle distance. Since the faces between the back and body of the bird form a consistent sharp convex region, these two parts are separated inappropriately. Deceasing the importance of angle distance or convexity can fix this problem. Result shown for the hand model in Figure 6(i) appears to be excellent, but this is rather deceiving since at the back, see Figure 6(j), our algorithm does not perform well at all. This is due to the presence of a shallow concave region around the knuckles. We believe that this is a manifestation of a common problem for all the algorithms [21, 12, 24] that place a great deal of emphasis on concavity. To obtain a more natural segmentation, e.g., cutting away a finger at its base, one may need to favor other factors, such as the cut length, over concavity. The use of graph min-cut to partition the fuzzy region [12] attempts at combining these factors, but its result will be sensitive to the interplay between geodesic and angle distances and it is generally hard to choose η and δ to ensure a natural segmentation.

Table 1. Execution times (in seconds) for our spectral mesh segmentation algorithm.

No. faces	Affinity	Eigen.	Clustering	Total
496	0.17	0.16	0.02	0.35
800	0.51	0.37	0.03	0.91
1200	1.7	0.52	0.09	2.31
1619	2.7	1.1	0.17	3.97
2000	4.76	1.44	0.29	6.49
4000	21.35	6.5	1.72	29.57

Finally, we report the timing of our algorithm in Table 1. These experiments were performed on an Intel Xeon 2.8 GHz machine with 1GB RAM. The majority of the time is actually spent on constructing the pairwise distance and affinity matrices. Comparatively, the actual clustering time is almost negligible.

5. Conclusion and future work

In this paper, we report results from our preliminary study on applying spectral clustering to 3D mesh segmentation. From a theoretical point of view, we have shown that clustering in the embedding space set up by the leading eigenvectors of the normalized affinity matrix should be easier than clustering with respect to the affinity matrix itself, provided that an appropriate number of eigenvectors are chosen — this is implied from the Polarization Theorem [2]. The polarization phenomenon suggests 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. This suggests that spectral embedding could serve as a preprocessing step for fuzzy clustering.

With no attempt at post-processing the result of spectral clustering, such as smoothing or graph min-cut [12], we are able to obtain very good results on relatively clean models. For more challenging models given in the paper, the algorithm is not always able to produce the most natural

segmentation with smooth boundaries. We believe the problem illustrated is common to all the approaches so far. Explicit smoothing or shortening of the segmentation boundary, e.g., using snakes [16], while not restricting them to lie along mesh edges, appears to be promising.

In terms of improving efficiency, there are a number of interesting directions to explore. These include the use of parallel or multilevel eigensolver to improve speed, utilization of the fast marching algorithm [21] to compute more accurate geodesic distance, and also an investigation into whether the affinity matrix can be made more sparse and how the precision of the eigenvector computations can influence the segmentation result.

Finally, we believe that fully automatic 3D mesh segmentation is still a rather difficult problem. Even with the above issues resolved, one still needs to choose parameters such as δ and η manually. Choices for the number of clusters and the number of eigenvectors used in the embedding are still ad-hoc and require more formal study.

Acknowledgments

This research is partially supported by an NSERC Discovery Grant (No. 611370) and a MITACS grant (No. 699112) held by the second author. The authors would like to thank Sagi Katz for the discussions and the anonymous reviewers for their helpful comments to improve the paper.

References

[1] C. J. Alpert and S. Z. Yao, "Spectral Partitioning: The More Eigenvectors, The Better," *Discrete Applied Mathematics* No. 90, pp. 3-26, 1999.

[2] M. Brand and K. Huang, "A Unifying Theorem for Spectral Embedding and Clustering," in C. M. Bishop and B. J. Frey (eds), *Proceedings of the Ninth International Workshop on Artificial Intelligence and Statistics,* Jan. 3-6, 2003.

[3] M. Carcassoni and E. R. Hancock, "Spectral Correspondence for Point Pattern Matching," *Pattern Recognition,* Vol. 36, pp. 193–204, 2003.

[4] B. Chazelle and L. Palios, "Decomposition Algorithms in Geometry," in *Algebraic Geometry and its Applications,* Springer-Verlag, C. C. Bajaj (eds), pp. 419-447, 1994.

[5] F. R. K. Chung, *Spectral Graph Theory,* CBMS, AMS, 1997.

[6] M. Garland, A. Willmott, and P. Heckbert, "Hierarchical Face Clustering on Polygonal Surfaces", *Proceedings of ACM Symposium on Interactive 3D Graphics,* pp. 49–58, 2001.

[7] C. Gotsman, X. Gu, and A. Sheffer, "Fundamentals of Spherical Parameterization for 3D meshes," *ACM Transaction on Graphics,* Vol. 22, No. 3, pp. 358-363, 2003.

[8] B. Hendrickson and R. Leland, "A Multilevel Algorithm for Partitioning Graphs," *Proceedings of the ACM/IEEE conference on Supercomputing*, Article No. 28, 1995.

[9] D. D. Hoffman and W. A. Richards, "Parts of Recognition," *Cognition,* Vol. 18, pp. 65–96, 1984.

[10] D. D. Hoffman and M. Singh, "Salience of visual parts," *Cognition,* Vol. 63, pp. 29-78, 1997.

[11] Z. Karni and C. Gotsman, "Spectral Compression of Mesh Geometry," *Computer Graphics Proceedings, SIGGRAPH 2000*, pp. 279-286.

[12] S. Katz and A. Tal,"Hierarchical Mesh Decomposition Using Fuzzy Clustering and Cuts," *ACM Transactions on Graphics,* Vol. 22, No. 3, pp. 954–961, 2003.

[13] G. Karypis and V. Kumar, "A Fast And High Quality Multilevel Scheme For Partitioning Irregular Graphs," *SIAM Journal on Scientific Computing,* Vol. 20, No. 1, pp. 359-392, 1998.

[14] Y. Koren, L. Carmel, and D. Harel, "ACE: A Fast Multiscale Eigenvector Computation for Drawing Huge Graphs," *Proceedings of IEEE Symposium on Information Visualization,* pp. 137-144, 2002.

[15] R. Lehoucq, K. Maschhoff, D. Sorensen, and C. Yang, *ARPACK,* http://www.caam.rice.edu/software/ARPACK/.

[16] Y. Lee and S. Lee, "Geometric snakes for triangular meshes," *Computer Graphics Forum,* Vol. 21, No. 3, pp. 229–238, 2002.

[17] X. Li, T. Toon, T. Tan, Z. Huang, "Decomposing Polygon Meshes for Interactive Applications," *Proceedings of the Symposium on Interactive 3D Graphics,* pp. 35–42, 2001.

[18] A. Mangan and R. Whitaker, "Partitioning 3D Surface Meshes using Watershed Segmentation," *IEEE Transactions on Visualization and Computer Graphics,* Vol. 5, No. 4, pp. 309–321, 1999.

[19] A. Y. Ng, M. I. Jordan, Y. Weiss, "On Spectral Clustering: Analysis and An Algorithm," in *Advances in Neural Information Processing Systems* 14, pp. 857–864, 2002.

[20] R. Ohbuchi, S. Takahashi, T. Miyazawa, and A. Mukaiyama, "Watermarking 3-D Polygonal Meshes in the Spectral Domain," *Proceedings of Graphics Interface,* pp. 9-18, 2001.

[21] D. L. Page, A. F. Koschan, and M. A. Abidi, "Perception-Based 3D Triangle Mesh Segmentation Using Fast Marching Watershed," *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition,* pp. 27–32, 2003.

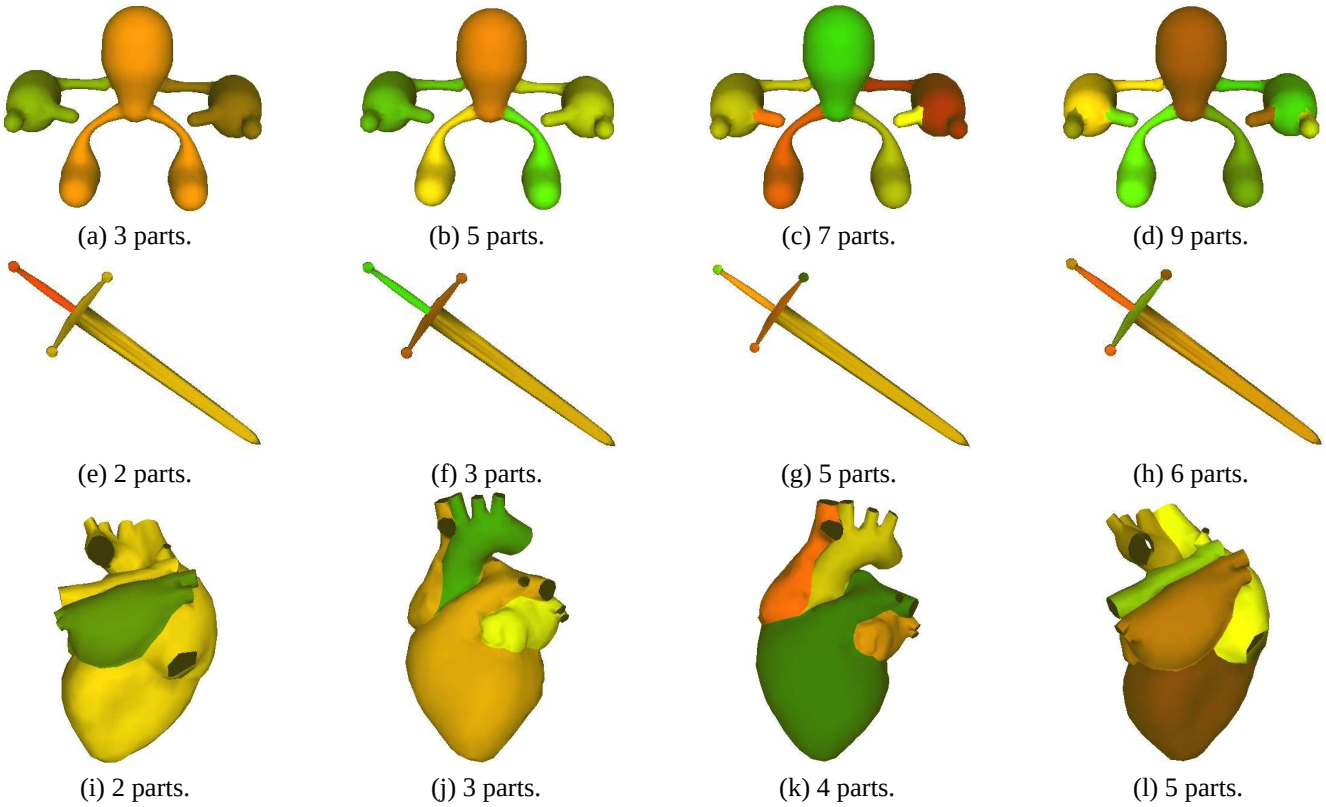
[22] G. L. Scott and H. C. Longuet-Higgins, "Feature Grouping by Relocalisation of Eigenvectors of the Proximity Matrix," in *Proceedings of the British Machine Vision Conference,* pp. 103–108, 1990.

[23] J. Shi and J. Malik, "Normalized Cuts and Image Segmentation," *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition,* pp. 731–737, 1997.

[24] S. Shlafman, A. Tal, and S. Katz, "Metamorphosis of Polyhedral Surfaces using Decomposition," *Eurographics 2002,* pp. 219–228, 2002.

[25] Y. Weiss, "Segmentation Using Eigenvectors: A Unifying View," *Proceedings IEEE International Conference on Computer Vision,* pp. 975–982, 1999.

[26] E. Zhang, K. Mischaikow, and G. Turk, "Feature-Based Surface Parameterization and Texture Mapping," *ACM Transactions on Graphics,* to appear.

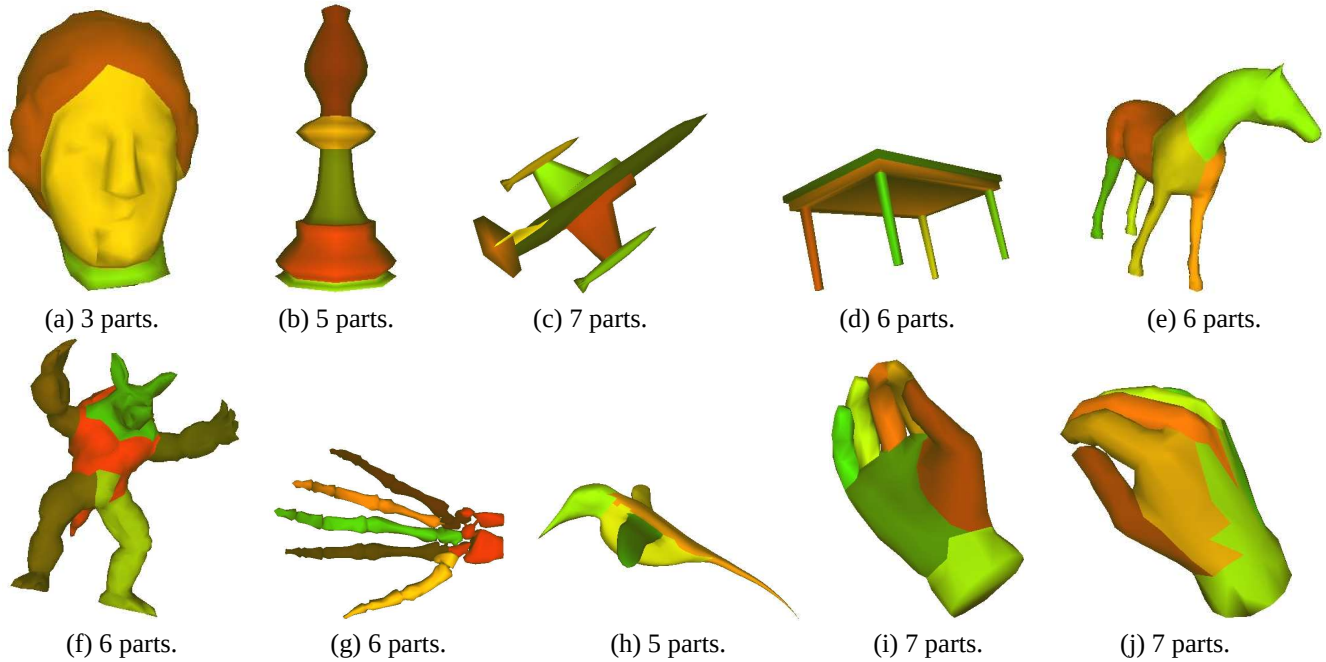


(a) 3 parts. (b) 5 parts. (c) 7 parts. (d) 9 parts.

(e) 2 parts. (f) 3 parts. (g) 5 parts. (h) 6 parts.

(i) 2 parts. (j) 3 parts. (k) 4 parts. (l) 5 parts.

Figure 5. Segmentation results showing hierarchical nature of spectral clustering.



(a) 3 parts. (b) 5 parts. (c) 7 parts. (d) 6 parts. (e) 6 parts.

(f) 6 parts. (g) 6 parts. (h) 5 parts. (i) 7 parts. (j) 7 parts.

Figure 6. More segmentation results.