A Formulation of Boundary Mesh Segmentation

Ariel Shamir
The Interdisciplinary Center, Herzliya
arik@idc.ac.il

Abstract

We present a formulation of boundary mesh segmentation as an optimization problem. Previous segmentation solutions are classified according to the different segmentation goals, the optimization criteria and the various algorithmic techniques used. We identify two primarily distinct types of mesh segmentation, namely parts *segmentation and* patch *segmentation. We also define generic algorithms for the major techniques used for segmentation.*

1 Introduction

Mesh segmentation (or partitioning) has become a key ingredient in many mesh manipulation algorithms in recent years. These include parametrization, texture mapping, shape matching, morphing, multi-resolution modeling, mesh editing, compression and more. Numerous techniques presented for segmentation were developed and some were borrowed from image segmentation, finite element meshes partitioning, unsupervised machine learning and other fields. Based on a survey of the different techniques, this paper introduces a unified formulation of the problem and presents classification of the different approaches.

A three dimensional boundary mesh M is defined as a tuple $\{V,E,F\}$ of *vertices* $V = \{p_i | p_i \in R^3, 1 \leq i \leq m\}$, *edges* $E = \{(p_i,p_j) | p_i,p_j \in V\}$, and *faces* F, which are usually triangles $F = \{(p_i,p_j,p_k) | p_i,p_j,p_k \in V\}$, but can also include other types of planar polygons (Figure 1). We use the term boundary mesh to distinguish these meshes from 3D volumetric meshes (e.g. tetrahedral), and to emphasize the fact that these meshes represent a 2D surface embedded in 3D. There are many constraints on the relations between the different elements (e.g. vertices, edges and faces) of the mesh which impose a valid representation. For example, in a 2-manifold mesh the neighborhood of every point which lays on the mesh is homeomorphic to a disk. In water-tight meshes the mesh will not contain any boundary edges. Generally we will restrict our discussion to 2-manifold boundary mesh representation, although many of the techniques reviewed do not directly rely on such constraints to work correctly.

Our basic definition of mesh segmentation is as follows:

> **Mesh segmentation Σ:** Let M be a 3D boundary-mesh, and S the set of mesh elements which is either V,E or F. A segmentation Σ of M is the set of sub-meshes $\Sigma = \{M_0,\ldots,M_{k-1}\}$ induced by a partition of S into k disjoint subsets.

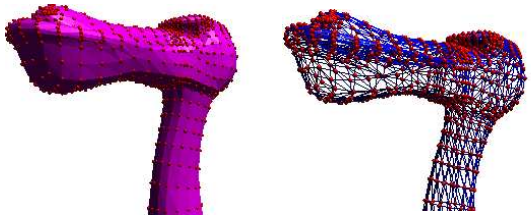

Figure 1: Vertices, faces and edges in a 3D boundary mesh.

Using a sub-set of elements $S' \subset S$, an induced sub-mesh $M' \subset M$ can be created by choosing all vertices which are included in S' as V', and then defining $M' = \{V',E',F'\}$. Where $E' = \{(p_i,p_j) \in E | p_i,p_j \in V'\}$ are all edges in which both vertices are a part of V', and F' is defined similarly as $F' = \{(p_i,p_j,p_k) \in F | p_i,p_j,p_k \in V'\}$. As can be seen, S can either be the vertices, edges or faces of the mesh and the partitioning of S induces a segmentation of M. Most mesh segmentation algorithms partition the faces of the mesh (i.e. $S = F$), some partition the vertices ($S = V$), and few the edges ($S = E$).

The key question in all mesh segmentation problems is how to partition the set S. Obviously, this relies heavily on the application in mind. However, one can formulate a mesh segmentation problem as an optimization problem by defining a specific criterion function $J : 2^S \rightarrow R$ which is a function of the partitioning of S. This is done in the following manner:

> **Mesh segmentation as an optimization problem:** Given a mesh M and the set of elements $S \in \{V,E,F\}$, find a disjoint partitioning of S into $S_0,\ldots,S_{k-1}$ such that the criterion function $J = J(S_0,\ldots,S_{k-1})$ be minimized (or maximized) under a set of constraints $\mathbb{C}$.

The set of constraints can give conditions both on the partitioning subsets S_i such as a limit on the number of elements, and on the segmentation sub-meshes M_i induced by the partition. For instance, that each sub-mesh be connected or be homeomorphic to a disk. In the simplest case $\mathbb{C}$ can be empty.

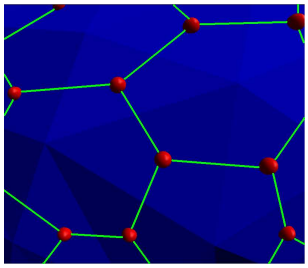

Figure 2: Part of the face-adjacency dual-graph of a mesh.

There are at least three closely related fields in computer science where similar segmentation or partitioning problems are encountered and where there is a large body of literature on these subjects. These are image segmentation [1, 2, 3], finite-element and simulation meshes partitioning [4, 5, 6, 7], and point-sets clustering in statistics and machine learning [8, 9, 10]. As we would like to concentrate on recent results in 3D boundary mesh segmentation, it is out of the scope of this paper to review these fields. Furthermore, although similar techniques can be applied in these fields, there are also some notable differences between them and 3D boundary mesh segmentation. Images are highly regular and are not embedded in higher dimensional space. Volumetric meshes for simulation are also full dimension meshes, hence their geometric properties are different than boundary meshes. Furthermore, the goal of their partitioning is usually to increase load balancing of computation between processors and reduced their communication. This means that the geometry of the mesh does not play as central role as in boundary embedded meshes. Point-sets in statistics are often defined in higher dimensions representing abstract notions and do not hold any explicit connectivity relation and hence are different in nature than 3D meshes.

A most useful analogy of mesh segmentation and graph partitioning is often introduced by defining the dual graph of the mesh [11]. Let S be the set of elements partitioned in M. We build the dual graph G of M by representing each element in S by a node in G and defining the edges in G by the adjacency relation in M of the elements of S. For instance, if $S = F$ then each node in G will represent a face in M and each edge will connect adjacent faces (Figure 2). When $S = V$ each node in G will represent a vertex in M, and the edges in G will in fact be the edges in M.

Using such a representation, a mesh segmentation problem can be cast as a (constrained) graph partitioning problem. In fact, by examining this analogy one can conclude that mesh segmentation is at least an NP-complete problem and often NP-hard (partitioning of a graph into approximately equal subsets of nodes so that the number of cut edges between the subsets is minimized is NP complete [12]). Furthermore, if $|\Sigma| = k$ and $|S| = n$, then a complete enumeration of all possible segmentations is unfeasible as the search space is of order k^n. This means we must resort to approximate solutions in feasible computation time.

We have classified the possible approximate solutions for mesh segmentation according to the approaches taken as follows:

1. Region growing.
2. Hierarchical clustering.
3. Iterative clustering.
4. Spectral clustering.
5. Other approaches.

In the following sections we elaborate on each of these approaches, define a generic algorithm for the main approaches, and classify the different mesh segmentations techniques found in literature. We have tried to detach the technique from the goal of segmentation and the criterion functions used. This view enhances the commonality of different works. Nevertheless, We also examine the different technique in view of their application domain or segmentation objective, and present constraints and optimization criteria which are frequently used in several algorithms.

2 Segmentation Type and Objectives

The type of mesh segmentation desired and the criterion function definition for optimization are affected by the segmentation objective. We distinguish between two different principal types of mesh segmentation. The first, which we will term *patch-type* segmentation, creates disk-like patches which obey certain geometric properties such as planarity, size or convexity. The second, which we will term *part-type* segmentation, is targeted more at partitioning the object defined by the mesh into meaningful components (Figure 3).

2.1 Patch-type Segmentation

Patch-type segmentation is often used for texture mapping [14, 15], building charts [16] and geometry-image creation [17]. In such applications the sub-mesh patch must be topologically equivalent to a disk and must not impose large distortion after parametrization onto 2D. Parametrization driven segmentations are also used in [18].

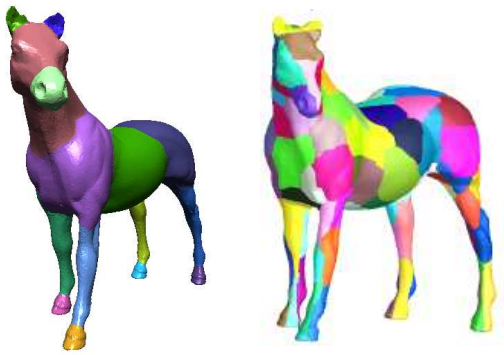

Figure 3: Two different types of mesh segmentation: part-type segmentation (left, taken from [13]) and patch-type segmentation (right, taken from [14])

Other applications where patch-type segmentation is used are remeshing and simplification [19, 20, 21, 22, 23, 24, 25]. In such applications, each patch is replaced either by one or a set of planar polygons, hence planarity is the desired property of the patches. Other patch-type decompositions impose convexity constraint [26] or constant curvature [27, 28].

In morphing, complex transformations between shapes can be simplified by a reduction to transformations between sub-patches [29, 30, 23].

For compression purposes by spectral analysis in [31] the set of mesh vertices is partitioned. The main motivation for breaking the mesh into smaller sub-meshes is to reduce the size of the Laplacian matrix of each sub-mesh for eigenvector computation.

Other applications which benefit from patch-type segmentation include radiosity, where the form-factor calculations usually uses planar patches, collision-detection, where bounding boxes are used on whole sub-mesh patches for efficiency [21], and animation with subdivision surfaces [32].

2.2 Part-type Segmentation

Part-type segmentation creates larger sub-meshes which often correspond to physical 3D parts of the object. This type of segmentation can assist shape matching and shape reconstruction [23, 33] by recognizing object parts. Such part matching can also be utilized for morphing [34]. Object part decomposition has also facilitated object skeleton definition which in turn was used for deformations and animation [35]. Lastly, bounding boxes defined around whole object parts can assist in fast collision detection calculations [36].

3 Constraints and Partitioning Criteria

No matter what algorithm is used for mesh segmentation, the most important factor affecting the result is the criteria used for partitioning and the constraints imposed on the process. The different criteria and constraints should be chosen based on the goal of segmentation. Nevertheless, some are used more frequently, hence we present them independent of the algorithm and goal of segmentation.

Some typical constraints regard the cardinality of the partition element sets:

- To eliminate too small or too large partitions, a bound on the maximum and minimum number of elements in each part is imposed.
- To create a more balanced partition, a bound on the ratio between the maximum and minimum number of elements in all parts is used.

Other constraints are defined on the geometry of the sub-meshes induced by the partitioning:

- Maximum/minimum area of sub-mesh.
- Maximum/minimum length of diameter or perimeter of sub-mesh.
- Maximum/minimum ratio of diameter or perimeter to area (a bias towards round sub-meshes).
- Convexity.

Lastly, topological constraints are also used to restrict the sub-mesh shape:

- Restriction to a single connected component.
- Restriction to a disk topology.

In terms of the criterion function, several segmentation algorithms choose planarity as the leading criteria to optimize. This criteria assists parametrization, simplification, texture mapping and other algorithms. Different works have used different types of norms to define planarity. Nevertheless, these are mostly a variants of the following:

L_∞ **distance norm:** given a cluster representative plane $ax+by+cz+d=0$, for any vertex $v=(v_x,v_y,v_z)$ it measures the maximum distance from the plane: $|(v_x,v_y,v_z,1)\cdot(a,b,c,d)| \leq \varepsilon$

L_2 **distance norm:** given a cluster representative plane $ax+by+cz+d=0$, and a set of vertices v_i it measures the average distance from plane: $\frac{1}{k}\sum_{i=1}^{k}((v_x,v_y,v_z,1)_i\cdot(a,b,c,d))^2 \leq \varepsilon$

L_∞ **orientation norm:** given a cluster representative plane $ax+by+cz+d=0$, for any face (or vertex) normal $n=(n_x,n_y,n_z)$ it measures the maximum difference of normals: $(1-(n_x,n_y,n_z)\cdot(a,b,c))\leq\varepsilon$

L_2 **orientation norm:** given a cluster representative plane $ax+by+cz+d=0$, and a set of face (or vertices) normals n_i it measures the average difference of normals: $\frac{1}{A}\sum_{i=1}^{k}\frac{1}{A_i}(1-(n_x,n_y,n_z)_i\cdot(a,b,c))\leq\varepsilon$, where A_i is a weighting factor for the region of the normal and $A=\sum_i A_i$. For instance A_i could be the area of the face for face normals, or simply 1 for uniform averaging.

In order to cluster non planar regions which are still similar geometrically a number of other measures are often used:

- Geodesic distance on the mesh.
- Difference in normals.
- Difference in the dihedral angles between faces.
- Differences in curvature.

4 Segmentation Techniques

In this section we classify previous mesh segmentation algorithms according to the approximation technique used to reach a solution.

4.1 Region Growing

The simplest of all possible approaches for approximation is the local-greedy approach which we term *region growing*. The algorithm for region growing starts with a seed element from S and grows a sub-mesh incrementally as follows:

Region Growing Algorithm

```
Initialize a priority queue Q of elements
Choose a seed and insert to Q
Create a cluster C from seed
Loop until Q is empty
   Get the next element s from Q
   If s can be clustered into C
      Cluster s into C
      Insert s neighbors to Q
```

The main difference between various algorithms which use region growing is in the criteria which determines if an element can be added to an existing cluster. The priority used in the queue is usually tightly coupled to this criteria as well. Other issues in region growing include the seeds selection mechanism, dealing with too small regions (for example if a single face cannot be clustered to any of its neighboring clusters), and post-processing of the segmentation borders for smoothing or straightening.

The super-face algorithm [37, 20] uses a region growing algorithm with a set of representative planes for the cluster approximated by an ellipsoid. The clustering criteria used are an L_∞ face-distance (distance of all face vertices) and a variant of the face-normal criteria along with a geometric constraint that prevents a face from 'folding-over' it's representative planes. The seed faces are chosen randomly. The borders between the segments are straightened in a post processing stage. Convex decomposition of the mesh also uses region growing with random starting faces [26]. An additional size constraint was added to the convexity criteria to achieve better decompositions.

For the purpose of creating a base triangle mesh with subdivision connectivity, a multiple source region growing is employed in [19]. The main idea is to create Voronoi-like patches on the mesh and then use the dual of the patches as the base triangular mesh. This imposes three constraints on the patches: 1. A patch must be homeomorphic to a disk, 2. Two patches cannot share more than one consecutive boundary, and 3. Not more than three patches can meet at a vertex. An approximation of geodesic distance between faces is used as the priority for selecting faces. The algorithm starts with one seed and then iteratively adds another seed in places where one of the constraints are violated, until the above constraints are met.

A method which simultaneously segments the mesh and defines a parametrization is defined in [15]. The seed faces are chosen randomly and greedy region growing is initialized which is capable of optimizing different criteria. For parametrization the criteria for adding a face to a region measures the distortion caused to a triangle during flattening to 2D. This is done using the singular values of the Jacobian of the affine transformation between the original 3D triangle and its counterpart in the plane.

Texture Atlas Generation in [16] uses region growing but instead of using seed faces and growing outward, the algorithm first extracts feature contours and uses them as boundaries between charts to grows the region inward. This also simplifies the test criteria which determines if an element can be added to an existing cluster since the boundaries are somehow pre-determined.

The watershed algorithm, originally used for images segmentation, is based on the definition of a height function on the mesh. The algorithm first finds and labels all local minima in the function. Each minimum also serves as the initial seed for a surface region. A region is grown incrementally from each seed until it reaches a ridge or maxima in the function, hence partitioning the function

terrain into regions. This region growing algorithm returns in many variations where the main difference between them is the definition of the feature energy or the height function in which "water rises".

A simulation of electrical charge distribution over the mesh is used in [38] for the height function definition. The charge density is very high and very low at sharp convexities and concavities, respectively. Thus, the object part boundary can be located at local charge density minima. In [27, 28] the function is based on vertex discrete curvature calculations [39, 40]. In [41] the algorithm approximates the feature strength of each vertex based on "normal-voting", i.e. the surface normal variation within a neighborhood of a vertex, and in [23] dihedral angles between faces is used. A more elaborate functional is used in [42] by defining a directional curvature height function between each two adjacent vertices u and v using the Euler's formula: $f_{uv} = \kappa_{max}\cos^2\theta + \kappa_{min}\sin^2\theta$, where κ_{max} and κ_{ming} are the maximum and minimum curvatures at u, and θ is the angle between the maximum principal direction and the vector connecting u to v in the tangent plane of u. In [33] this height function is further quantized into discrete values preventing spills from one region to another.

4.2 Hierarchical Clustering

The search for local optimum of each region separately may sometimes create unsatisfactory global results. For example, the number of regions depends heavily on the choice of initial seeds. Furthermore, there are times when a hierarchical segmentation structure is beneficial for specific applications. Hierarchical clustering, while still a greedy approach, can be seen as "global-greedy" since it always chooses the best merging operation for all clusters and doesn't concentrates on growing one:

Hierarchical Clustering Algorithm

```
Initialize a priority queue Q of pairs
Insert all valid element pairs to Q
Loop until Q is empty
   Get the next pair (u,v) from Q
   If (u,v) can be merged
      Merge (u,v) into w
      Insert all valid pairs of w to Q
```

Similar to region-growing, the difference between various hierarchical clustering algorithms lies mainly in the merging criteria and the priority of elements in the queue.

Hierarchical clustering starts initially when each face is its own cluster. Each pair of clusters is assigned a cost for merging. Hierarchical face clustering [21] uses L_2 distance and orientation norms from representative planes as a measure of planarity, but formulates them using quadric error metric for efficient computation. The algorithm also uses a bias term to create circular compact cluster shapes by using the ratio between the square of the perimeter and $4\pi A$ where A is the area of the cluster.

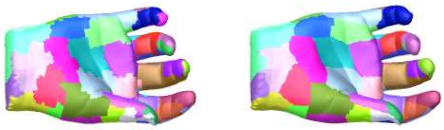

Figure 4: Raw segmentation results may require post-processing to smooth the boundary between patches (example taken from [14]).

Mean squared distance of a patch to the best fitted plane is also used in [14] for the creation of charts. However, the measure is integrated on all patch faces and not only on vertices. Compactness of patches is measured simply as the squared perimeter length. Additional tests are performed before merging two clusters to take care of topology constraints such that each clustered patch remain homeomorphic to a disk. In post processing smooth boundaries between the charts are created calculating constrained shortest path (Figure 4).

Although working on the dual graph of the mesh in [22], edge contracting according to a priority is similar to hierarchical clustering. This is due to the fact that an edge contraction in the graph is equivalent to a merge of two clusters. The priority of edges used in the algorithm is a combination of geometric and topological costs including size, shape, curvature and more.

4.3 Iterative Clustering

In the two previous methods the number of resulting clusters is unknown in advance. A different type of search for optimal segmentation is defined by iteratively searching for the best clustering given that the number of clusters is fixed. The basis of this approach is the k-means algorithm, sometimes referred to as Lloyd or Lloyd-Max algorithm [43, 10]. The iterative process begins with k representatives representing k clusters. Each element is then assigned to one of the k clusters. Subsequently, the k representatives are re-calculated from the k-clusters and the assignment process begins again. The process terminates when the representatives stop changing:

Iterative Clustering Algorithm

```
Initialize k representatives of k clusters
Loop until representatives do not change
   For each element s
      Find the best representative i for s
      Assign s to the i^th cluster
   For each cluster i
      Compute a new representative
```

The key issue concerning iterative clustering algorithm is convergence. The measure of 'best' representative for an element and the computation of new representatives from clusters should be chosen with care so that the process converges. Other issues such as the choice of initial representative can also affect the convergence and the final result.

To create compatible segmentation of two objects for morphing purposes, a k-means based face-clustering algorithm is proposed in [34]. A distance measure between faces is defined as a weighted combination of the approximate geodesic distance (the sum of distance from centroid to the center of edge) and the difference in dihedral angle.

After representatives are chosen each face is assigned to the cluster of its closest representative. New representatives are chosen as the faces which minimize the sum of distances to all other faces in the cluster

Another variant of k-means algorithm is presented in [25] for the creation of planar shape proxies. Two different error metrics are defined. L^2 measures the integral over a patch of the squared error between point on the patch and its planar proxy. The point-difference is the distance between the point on the patch and its orthogonal projection on the proxy.

A more superior metric both in terms of results and in terms of simplicity of calculation is $L^{2,1}$, which is defined simply as the L^2 norm on the normal field of the mesh. This means the error is an integral over the difference between the normal of a point in the patch and the proxy normal.

These metrics are used also to define new proxy representatives in each iteration. In order to keep the clustered regions connected and non-overlapping, only triangles adjacent to currently grown regions are inserted to the queue.

Mesh charts for geometry image creation are defined in [17] using iterative clustering. This algorithm also ensured connectivity by adding only neighboring triangles to existing charts. The cost of adding is a measure of geometric distance between the face and its neighboring face in the chart, and difference between the face normal and the chart normal.

The new seeds for the next iteration are simply the central faces in each chart. To assure the disk topology of all charts some face assignments are disallowed. This may lead to a possibility of an orphan face left not clustered. The solution to this is to add this face as a seed in the next iteration, hence enlarging k by one. This idea is also used to initialize the seed set by adding the last face assigned in the previous iteration as a new seed in the next iteration, starting from 1 seed until k seeds are created.

4.4 Spectral Analysis

Spectral graph theory [44] states the relationship between the combinatorial characteristics of a graph and the algebraic properties of its Laplacian [11]. If A is the adjacency matrix of a graph G and D is a diagonal matrix which holds the degree (valance) of vertex i as $d_{i,i}$, then the Laplacian of G is defined as the matrix $L = D - A$.

Let $\{\xi_0, \xi_1, \ldots, \xi_{n-1}\}$ be the eigenvectors of L. By embedding the graph G into the space R^d using d first eigenvectors, one can reduce the combinatorial graph partitioning problem to a geometric space-partitioning problem [45, 11].

The Laplacian matrix of the vertex adjacency graph was used for mesh compression purposes in [31]. Due to high computation cost the mesh was segmented into smaller sub-meshes and each one treated separately. However, these sub-meshes should be balanced in size and the edge straddling the different sub-meshes should be minimized in order to reduce the visual effects. These conditions are similar to FEM mesh decomposition and hence MaTiS [4] graph partitioning application was used.

Using a slightly different formulation in [46] a symmetric affinity matrix $W \in R^{n \times n}$ is constructed where for all i, j, W_{ij} encodes the probability that face i and face j can be clustered into the same patch $0 \leq W_{ij} \leq 1$. This matrix may be viewed as the adjacency matrix of a complete (weighted) graph whose nodes are the mesh faces. The Spectral analysis of this matrix creates a partitioning which induces a segmentation of the mesh.

4.5 Other Methods

A hybrid algorithm between iterative clustering and graph cut is proposed in [35]. At the initial stage iterative clustering is used to create general partitioning. However this partition remains fuzzy around the boundary regions of the segments and a final decomposition is created using graph cut flow algorithm. The algorithm is also capable of creating hierarchical decomposition by top down binary partitioning.

An approach based on skeletonization is proposed in [36]. First an approximation of the skeleton of the mesh is extracted. Next, a plane perpendicular to the skeleton branches is sweeped over the mesh and critical points are identified. Each critical skeleton point is used to define a cut using the sweep plane which segments the mesh to different parts. Using this scheme, the segmentation is defined implicitly by the creation of cuts.

Another scheme which targets the segment boundaries instead of building the segments by clustering is presented as mesh-scissoring in [13]. Following the minima-rule from perception [47], minimum curvature feature-

contours are extracted from the mesh. These contours are then closed to form loops around mesh parts. Finally snakes are used to smooth the cuts which define a part-type segmentation of the object (Figure 3).

An approach based on image segmentation is presented in [24]. The problem of 3D boundary mesh segmentation is reduced to image segmantation by using geometry images [48] to represent the mesh. The portioning of the image imposes a mesh segmentation in 3D.

Lastly, several manual segmentation and partitioning are found in literature [29, 30, 49]. These usually define the boundaries between segments either explicitly or by designating some vertices and calculating shortest path between them.

5 Concluding Remarks

We have presented the main approaches for boundary mesh segmentation and identified different optimization criteria used. It is obvious that the key factor in choosing both the algorithm and the criteria is the application in mind. For example, we have identified a distinct difference between the results of patch-type segmentations and part-type segmentations. This difference is mainly due to the differences in the goals of segmentation. For this reason, it was difficult to assess quality and compare the different results, and we focused more on extracting and formulating the major algorithmic techniques used to date. Although there are already numerous techniques to create mesh segmentations, it seems that directions to address this problem are only beginning.

6 Acknowledgments

The author would like to thank Daniel Cohen-Or for fruitful discussions and comments, and Hugeus Hoppe for permission to use images of his work.

References

[1] X. Zhuang, Y. Huang, K. Palaniappan, and Y. Zhao, "Gaussian mixture density modeling: Decomposition and applications," *IEEE Transactions on Image Processing*, vol. 5, pp. 1293–1302, September 1996.

[2] C. Tomasi and R. Manduchi, "Bilateral filtering for gray and color images," in *Proc. of the sixth international Conference of Computer Vision*, pp. 839–846, 1998.

[3] D. Comaniciu and P. Meer, "Mean shift: A robust approach towards feature space analysis," *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 24, pp. 603–619, May 2002.

[4] G. Karypis and V. Kumar, "Metis: A software package for partitioning unstructured graphs, partitioning meshes, and computing fill-reducing orderings of sparse matrices.." http://wwwusers.cs.umn.edu/ karypis/metis/metis.html, 1998.

[5] G. Karypis and V. Kumar, "Multilevel algorithms for multi-constraint graph partitioning," in *Proceedings of the 36th ACM/IEEE conference on Design automation conference*, (New Orleans, Louisiana), pp. 343 – 348, 1999.

[6] C. Nikos and D. Nave, "Simultaneous mesh generation and partitioning for delaunay meshes," in *Proceedings of the 8th International Meshing Roundtable*, (South Lake Tahoe), pp. 55–66, 1999.

[7] I. Moulitsas and G. Karypis, "Multilevel algorithms for generating coarse grids for multigrid methods," in *Proceedings of the 2001 ACM/IEEE conference on Supercomputing*, (Denver, Colorado), pp. 45–45, 2001.

[8] R. Arabie, L. Hubert, and G. DeSoete, eds., *Clustering and Classification*. River Edge, NJ: World Scientific Publishers, 1996.

[9] S. J. Roberts, "Parametric and non-parametric unsupervised cluster analysis," *Pattern Recognistion*, vol. 30, pp. 327–345, 1997.

[10] R. O. Duda, P. E. Hart, and D. G. Stork, *Pattern Classification (2nd ed.)*. Wiley Interscience, October 2000.

[11] C. Gotsman, "On graph partitioning, spectral analysis, and digital mesh processing," in *Proceedings of Shape Modeling International*, (Seoul), pp. 165–169, 2003.

[12] M. Garey, D. Johnson, and L. Stockmeyer, "Some simplified np-complete graph problems," *Theoretical Computer Science*, vol. 1, pp. 237–267, 1976.

[13] Y. Lee, S. Lee, A. Shamir, D. Cohen-Or, and H.-P. Seidel, "Intelligent mesh scissoring using 3d snakes." Submitted, 2004.

[14] P. Sander, J. Snyder, S. Gortler, and H. Hoppe, "Texture mapping progressive meshes," in *Proceedings of ACM SIGGRAPH*, pp. 409–416, 2001.

[15] O. Sorkine, D. Cohen-Or, R. Goldenthal, and D. Lischinski, "Bounded-distortion piecewise mesh parameterization," in *Proceedings of IEEE Visualization 2002*, 2002.

[16] B. Levy, S. Petitjean, N. Ray, and J. Maillot, "Least squares conformal maps for automatic texture atlas generation," in *ACM Computer Graphics, Proc. SIGGRAPH 2002*, pp. 362–371, 2002.

[17] P. Sander, Z. Wood, S. Gortler, J. Snyder, and H. Hoppe, "Multi-chart geometry images," in *Proceedings of the Eurographics Symposium on Geometry Processing*, pp. 146–155, 2003.

[18] K. Inoue, I. Takayuki, Y. Atsushi, F. Tomotake, and S. Kenji, "Face clustering of a large-scale cad model for surface mesh generation," *Computer Aided Design*, vol. 33, March 2001. The 8th International Meshing Roundtable Special Issue: Advances in Mesh Generation.

[19] M. Eck, T. DeRose, T. Duchamp, H. Hoppe, M. Lounsbery, and W. Stuetzle, "Multiresolution analysis of arbitrary meshes," in *Proceedings of ACM SIGGRAPH 1995*, pp. 173–182, 1995.

[20] A. Kalvin and R. Taylor, "Superfaces: Polygonal mesh simplification with bounded error," *IEEE Computer Graphics and Applications*, vol. 16, no. 3, 1996.

[21] M. Garland, A. Willmott, and P. Heckbert, "Hierarchical face clustering on polygonal surfaces," in *Proc. ACM Symposium on Interactive 3D Graphics*, March 2001.

[22] A. Sheffer, "Model simplification for meshing using face clustering," *Computer Aided Design*, vol. 33, pp. 925–934, 2001.

[23] E. Zuckerberger, A. Tal, and S. Shlafman, "Polyhedral surface decomposition with applications," *Computers & Graphics*, vol. 26, no. 5, pp. 733–743, 2002.

[24] I. M. Boier-Martin, "Domain decomposition for multiresolution analysis," in *Proceedings of the Eurographics Symposium on Geometry Processing*, pp. 29–40, 2003.

[25] D. Cohen-Steiner, P. Alliez, and M. Desbrun, "Variational shape approximation," *ACM Transactions on Graphics (Proceedings SIGGRAPH 2004)*, vol. 23, p. to appear, 1994.

[26] B. Chazelle, D. Dobkin, N. Shourhura, and A. Tal, "Strategies for polyhedral surface decomposition: An experimental study," *Computational Geometry: Theory and Applications*, vol. 7, no. 4-5, pp. 327–342, 1997.

[27] A. P. Mangan and R. T. Whitaker, "Surface segmentation using morphological watersheds," in *Proc. IEEE Visualization 1998 Late Breaking Hot Topics*, 1998.

[28] A. Mangan and R. Whitaker, "Partitioning 3d surface meshes using watershed segmentation," *IEEE Transactions on Visualization and Computer Graphics*, vol. 5, no. 4, pp. 308–321, 1999.

[29] A. Gregory, A. State, M. Lin, D. Manocha, and M. Livingston, "Interactive surface decomposition for polyhedral morphing," *The Visual Computer*, vol. 15, pp. 453–470, 1999.

[30] M. Zockler, D. Stalling, and H.-C. Hege, "Fast and intuitive generation of geometric shape transitions," *The Visual Computer*, vol. 16, no. 5, pp. 241–253, 2000.

[31] Z. Karni and C. Gotsman, "Spectral compression of mesh geometry," in *Proceedings of ACM SIGGRAPH 2000*, pp. 279–286, 2000.

[32] T. DeRose, M. Kass, and T. Truong, "Subdivision surfaces in character animation," in *ACM Computer Graphics, Proc. SIGGRAPH 1998*, pp. 85–94, 1998.

[33] D. Page, M. Abidi, A. Koschan, and Y. Zhang, "Object representation using the minima rule and superquadrics for under vehicle inspection," in *Proceedings of the 1st IEEE Latin American Conference on Robotics and Automation*, pp. 91–97, 2003.

[34] S. Shlafman, A. Tal, and S. Katz, "Metamorphosis of polyhedral surfaces using decomposition," *Computer Graphics forum*, vol. 21, no. 3, 2002. Proceedings Eurographics 2002.

[35] S. Katz and A. Tal, "Hierarchical mesh decomposition using fuzzy clustering and cuts," *ACM Transactions on Graphics (Proceedings SIGGRAPH 2003)*, vol. 22, no. 3, pp. 954–961, 2003.

[36] X. Li, T. Toon, T. Tan, and Z. Huang, "Decomposing polygon meshes for interactive applications," in *Proceedings of the 2001 symposium on Interactive 3D graphics*, pp. 35–42, 2001.

[37] A. Kalvin and R. Taylor, "Superfaces: Polyhedral approximation with bounded error," in *SPIE Proceedings 2164*, pp. 2–13, 1994.

[38] K. Wu and M. Levine, "3d part segmentation using simulated electrical charge distributions," *IEEE transactions on pattern analysis and machine intelligence*, vol. 19, no. 11, pp. 1223–1235, 1997.

[39] M. Meyer, M. Desburn, P. Schröder, and A. H. Barr, "Discrete differential - geometry operators for triangulated 2-manifolds," in *Proceedings VisMath '02*, (Berlin), 2002.

[40] S. Pulla, A. Razdan, and G. Farin, "Improved curvature estimation for watershed segmentation of 3-dimensional meshes." manuscript, 2001.

[41] Y. Sun, D. L. Page, J. K. Paik, A. Koschan, and M. A. Abidi, "Triangle mesh-based edge detection and its application to surface segmentation and adaptive surface smoothing," in *Proceedings of the International Conference on Image Processing ICIP02, Vol. III*, (Rochester, N.Y.), pp. 825–828, 2002.

[42] D. Page, A. Koschan, and M. Abidi, "Perception-based 3d triangle mesh segmentation using fast marching watersheds," in *Conference on Computer Vision and Pattern Recognition (CVPR '03) - Volume II*, pp. 27–32, 2003.

[43] S. Lloyd, "Least square quantization in pcm," *IEEE Transactions on Information Theory*, vol. 28, pp. 129–137, 1982.

[44] F. R. K. Chung, *Spectral Graph Theory*. No. 92 in CBMS Regional Conference Series in Mathematics, American Mathematical Society, 1997.

[45] C. Alpert and S. Yao, "Spectral partitioning: The more eigenvectors, the better," in *32nd ACM/IEEE Design Automation Conference*, (San Francisco), pp. 195–200, 1995.

[46] R. Liu and H. Zhang, "Segmentation of 3d meshes through spectral clustering." Submitted, 2004.

[47] D. Hoffman and M. Signh, "Salience of visual parts," *Cognition*, vol. 63, pp. 29–78, 1997.

[48] X. Gu, S. Gortler, and H. Hoppe, "Geometry images," *ACM Transaction on Graphics, Special issue for SIGGRAPH conference*, vol. 21, no. 3, pp. 355–361, 2002.

[49] T. Funkhouser, M. Kazhdan, P. Shilane, P. Min, W. Kiefer, A. Tal, S. Rusinkiewicz, and D. Dobkin, "Modeling by example," *ACM Transactions on Graphics (Proceedings SIGGRAPH 2004)*, vol. 23, p. to appear, 2004.