Segmentation and Shape Extraction of 3D Boundary Meshes

Ariel Shamir

Efi Arazi School of Computer Science
The Interdisciplinary Center, Herzliya
arik@idc.ac.il

Abstract

In this report we present the state of the art on segmentation, or partitioning techniques used on boundary meshes. Recently, these have become a part of many mesh and object manipulation algorithms in computer graphics. We formulation the segmentation problem as an optimization problem and identify two primarily distinct types of mesh segmentation, namely parts *segmentation and* patch *segmentation. We classify previous segmentation solutions according to the different segmentation goals, the optimization criteria and features used, and the various algorithmic techniques employed. We also present generic algorithms for the major techniques of segmentation.*

1. Introduction

Mesh segmentation (or mesh partitioning) has become a key ingredient in many mesh manipulation algorithms and applications in recent years. These include parametrization, texture mapping, shape matching, morphing, multi-resolution modeling, mesh editing, compression, animation and more. Furthermore, recent advances in geometry processing aims at extracting shape features and structure from 3D meshes to enhance semantic-based shape representations (see e.g. [aim]). Numerous techniques presented for segmentation were developed. Some were borrowed from related fields such as image segmentation, finite element meshes partitioning, unsupervised machine learning and other fields. In this report we will survey the different techniques used for various purposes, and classify them to a small set of general algorithms. These can then provide understanding as to the strengths and weaknesses of each technique and assist in future choices for different applications.

We first present a formulation of mesh segmentation problem as an optimization problem, and distinguish between the two major types of segmentations. *Part-type* segmentation, where the 3D object is partitioned into meaningful components, which are mostly volumetric, and *surface-type* segmentation, where the boundary surface of the object is partitioned into charts (Figure 3).

Further we present the different criteria used for guiding the segmentation of a mesh. These can be defined as *features* or properties of the mesh and must be extracted prior to the segmentation of the mesh. These include simple surface measures such as area, size or length, various differential properties such as curvature, normal direction, some distance measures such as geodesic distances, distance to the medial axis, or the shape diameter, and more.

Lastly, we concentrate on the algorithmic side and present the major algorithms used in mesh partitioning. We classify these algorithms to several approaches and provide the link to general clustering algorithms.

2. Definitions

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

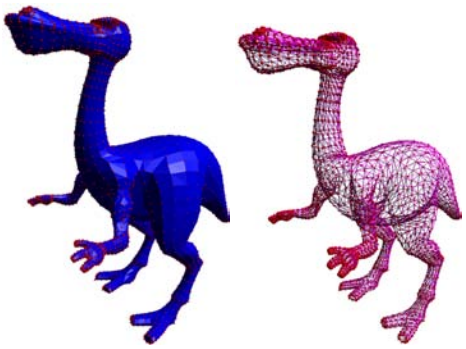

Figure 1: *Vertices, faces and edges in a 3D boundary mesh.*

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 sub-sets.

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 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 C can be empty.

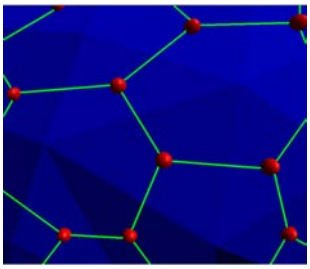

Figure 2: *Part of the face-adjacency dual-graph of a mesh.*

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. These are image segmentation [ZHPZ96, TM98, CM02], finite-element and simulation meshes partitioning [KK98, KK99, NN99, MK01], and point-sets clustering in statistics and machine learning [AHD96, Rob97, DHS00]. 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 [Got03]. 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 [GJS76]). Furthermore, if $|\Sigma| = k$ and $|S| = n$, then a complete enu-

meration 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 analysis.
5. Graph-Cut techniques.
6. 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.

3. Segmentation Type and Objectives

The type of mesh segmentation desired and the criterion function definition for optimization are affected by the segmentation objective. Although there are various objectives, we have found that there is a distinction between two different principal types of mesh segmentations. The major distinction between the two is based on a different point of view on the object being partitioned – either a 3D volumetric view or a 2D surface view (Figure 3). Hence, the first type, which we will term *part-type* segmentation, is targeted more at partitioning the *object* defined by the mesh into meaningful or 'semantic' components [Bie87], creating in general volumetric parts. The second type, which we will term *surface-type* segmentation, uses mostly surface geometric properties of the mesh such as planarity or curvature to creates *surface* patches. Obviously, there are also times when 'semantic' components are used by surface-type segmentation, e.g. in CAD oriented segmentations [SAKJ01], where an object is decomposed into geometric primitives such as planes, cylindrical patches, spherical parts etc. Similarly, there are times when surface-based attributes are used to partition an object into volumetric parts, such as minimum curvature in the minima-rule [HR84, HS97].

Although there are segmentation objectives that are shared by both segmentation types, in general surface-type and part-type segmentation imply different objectives. Hence, in the following we list the different objectives based on the two segmentation types.

Figure 3: *Two different types of mesh segmentation: part-type segmentation (left, taken from [LLS*05]) and surface-type segmentation (right, taken from [SSGH01])*

3.1. Surface-type Segmentation

Surface-type segmentation is often used for texture mapping [SSGH01, SCOGL02, ZMT05], building charts [LPRM02, ZSGS04] and geometry-image creation [SWG*03]. 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 [ITA*01].

Other applications where surface-type segmentation is used are remeshing and simplification [EDD*95, KT96, GWH01, She01, ZTS02, BM03, CSAD04]. In most of those, each patch is replaced either by one or a set of planar polygons, hence planarity is the desired property of the patches. More recently, other types of proxies have been used to replace mesh patches defining different types of patch properties for spherical, cylindrical, and rolling ball blends [WK05, AFS06]. For actual reconstruction and creation of physical models and toys, strips and quasi-developable patches are built in [MS04, JKS05, STL06]. Other surface-type decompositions impose convexity constraint [CDST97] or constant curvature [MW98, MW99].

In morphing, complex transformations between shapes can be simplified by a reduction to transformations between sub-patches [GSL*99, ZSH00, ZTS02]. Similarly, the transfer of details, movement or deformation from one mesh to another can be achieved if there is a map between them. Finding such a mapping on the whole object is difficult and is often simplified by segmentation and matching parts or by simultaneous parameterizations [KS04, SAPH04].

For compression purposes by spectral analysis in [KG00] 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 surface-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 [GWH01], and animation with subdivision surfaces [DKT98].

3.2. Part-type Segmentation

Part-type segmentation objective is rooted in the study of human perception. Examining human image understanding many works indicate that recognition and shape understanding are based on structural decomposition of the shape into smaller parts [HS97, Bie87, HR84]. Towards this end, part-type segmentation decomposes a 3D object into sub-meshes which often correspond to physical 3D "semantic parts" of the object.

In [MPS*03, MPS*04], part-type semantic segmentation is created based on analyzing the intersection curves of a ball centered around each vertex, and the mesh. This analysis segments a surface into connected components that are either body parts or elongated features, that is, handle-like and protrusion-like features.

Part-type segmentation was used for modeling by assembling parts of objects to create new designs from existing ones [FKS*04]. It was also used to create bead-style toys in [RGS04].

Decomposing and, later on, recognizing and matching object sub-parts can assist shape matching and retrieval, and shape reconstruction [ZTS02, PAKZ03, Bia03]. Such part matching can also be utilized for morphing [STK02]. Object part decomposition has also facilitated object skeleton creation [MPS*03, KT03, WML*06], which in turn was used for deformations and animation. Lastly, bounding boxes defined around whole object parts can assist in fast collision detection calculations [LTTH01].

A recent comparative study on various part-type segmentation technique can be found in [AKM*06].

4. Attributes and Partitioning Criteria

No matter what algorithm is used for mesh segmentation, the most important factor affecting the result is the criteria for deciding which elements belong to the same segment and the constraints imposed on the partitioning process. These criteria are usually based on attributes extracted from the mesh. Although there are many different attributes chosen based on the goal of segmentation, some reoccur frequently in several works. Hence, we present those independently of the algorithms they appear in, and the final goal of segmentation. We will first describe some of the constraints used on partitions and then some of the attributes commonly used for segmentation.

4.1. Constraints

There are three major types of possible constraints for segmentation: cardinality constraints, geometric constraints and topological constraints. Some typical cardinality constraints regarding the partition sets are:

- A bound on the maximum and/or minimum number of elements in each part. This is often used to eliminate too small or too large partitions.
- A bound on the ratio between the maximum and minimum number of elements in all parts. This is used to create a more balanced partition.
- When applicable (i.e. when this number is not set a-priori) A bound on the maximum or minimum number of segments may also be used to balance the partition.

Geometric constraints are imposed on the sub-mesh induced by the partitioning. Some typical geometric constraints are:

- Maximum/minimum area of sub-mesh.
- Maximum/minimum length of diameter or perimeter of sub-mesh.
- More complex constraints that are either hard constraints (convexity) or add a bias towards specific shapes. For instance, maximum or minimum ratio of diameter or perimeter to area can provide a bias towards roundly shaped sub-meshes.

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

- Restriction of the segment to a single connected component.
- Restriction of the segment to be topologically equivalent to a disk.

4.2. Mesh Attributes

The most important factor that governs the segmentation results is the criterion function used for clustering mesh elements. We will discuss the following set of attributes extracted from the mesh:

1. Planarity of various forms.
2. Other geometric proxies (spheres, cylinders, developable surfaces).
3. Difference in normals of vertices or dihedral angles between faces.
4. Slippage.
5. Geodesic distances on the mesh.
6. Curvature.
7. Medial axis and related functions.

One of the leading criteria used for segmentation is planarity. This criteria assists segmentation goals such as parametrization, simplification, texture mapping and other algorithms. Different works have used different types of

norms to define planarity of segments. Assuming each segment is a cluster of elements best represented by a plane $ax+by+cz+d=0$, most criteria are a variants of the following:

L_∞ **distance norm:** given a cluster representative plane, 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, 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, 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, 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.

To cluster non-planar regions, other cluster representatives must be used and other criteria must be defined. Several works use primitives such as spheres and cylinders and try to find the best fitting primitive in least square sense. Other types of regions include rolling ball blends, triangle strips, and cones as quasi-developable surfaces (see Section 5.3 for specific examples). A more straightforward approach to cluster non-planar regions is simply to measure the differences in normal direction or in dihedral angles between mesh elements. Depending on the tolerance of this difference, either almost planar or also curved parts can be created.

A different approach is presented in [GG04] for slippage analysis. slippable motions are rigid motions which, when applied to a shape, slide the transformed version against the stationary version without forming any gaps. slippable shapes include rotationally and translationally symmetrical shapes such as planes, spheres, and cylinders, which are often found as components of mechanical parts. A slippable motion of each point P must be tangential to the surface at that point. Hence, by posing this as a minimization problem one can search for an instantaneous motion vector $[r,t]$ that, when applied to P minimizes the motion along the normal direction at each point:

$$\min_{[rt]}\sum_{i=1}^{n}((r\times p_i+t)\cdot n_i)^2$$

This equation leads to least-squares problem whose minimum is the solution of a linear system. Hence, the slippable motions of a local neighborhood of a point can be determined by computing its eigenvalues.

Two most useful functions in various algorithms are surface properties of the mesh. The first is a differential property of the mesh - curvature (Figure 4, left), while the second, averages geodesic distances (AGD), depends more on global embedded geometry (Figure 4, middle). There are many variations for curvature calculations either using discrete approximations or by locally fitting a quadratic function and taking its curvature as the curvature at the fitting point. Some examples can be found in [MDSB02, ACSD*03]. The AGD, also sometimes called centricity, is taken as the average geodesic distance from each point to all other points on the mesh. This means that points in the center of the object will have low AGD value, and points on the periphery will have a large value. Calculating the AGD is usually done by finding the geodesic distances from all vertices to all vertices. This can be done using Dijsktra algorithm for all pairs shortest path on the mesh graph. A more accurate approach is to use the fast marching method of [KS98].

The medial axis and medial axis transform (MAT) are an important topological attributes of the object [ACK01, DZ02, CCM97]. They carry information on the structure and size of the object and can often be used as guidelines for segmentation. Another related function is defined in [SSCO05] as the shape diameter function (SDF). This function measures the local diameter of the object at points on its boundary instead of the local radius (distance to the medial axis). The function values on a point laying on the mesh surface are averaged from sampling the length of rays sent from the point inward to the other side of the object's mesh (Figure 4, right).

5. Segmentation Techniques

In this section we classify previous mesh segmentation algorithms according to the approximation technique used to reach a solution, i.e. the technique used to find the approximation for the segmentation optimization problem.

5.1. Region Growing

The simplest of all possible approaches for segmentation 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
Loop until all elements are clustered
   Choose a seed element 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
Merge small clusters into neighboring ones
```

The main difference between various algorithms which

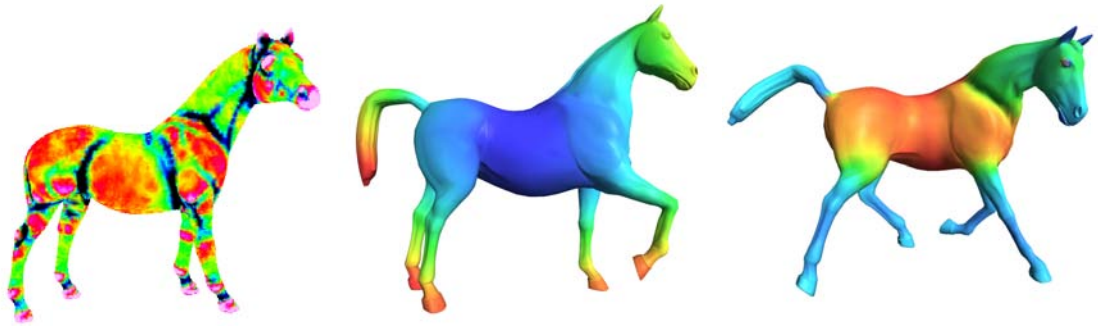

Figure 4: *Example of mesh attributes used for partitioning. Left: minimum curvature, middel: average geodesic distance, right: shape diameter function.*

use region growing is the criterion which determines if an element can be added to an existing cluster. The priority used in the priority-queue is usually tightly coupled with this criterion 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 [KT96] 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 [CDST97]. An additional size constraint was added to the convexity criteria to achieve better decompositions.

A common variation of the region growing algorithm starts from multiple source seeds and advances from all of them in parallel. For instance, for the purpose of creating a base triangle mesh with subdivision connectivity, a multiple source region growing is employed in [EDD*95]. 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.

```
Multiple Source Region Grow
Initialize a priority queue Q of pairs
Choose a set of seed elements {s_i}
Create a cluster C_i from each seed s_i
Insert the pairs < s_i,C_i > to Q
Loop until until Q is empty
   Get the next pair < s_k,C_k > from Q
   If s_k is not clustered already and
   s_k can be clustered into C_k
      Cluster s_k into C_k
      For all un-clustered neighbors s_i of s_k
         insert < s_i,C_k > to Q
Merge small clusters into neighboring ones
```

A method which simultaneously segments the mesh and defines a parametrization is defined in [SCOGL02]. 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 [LPRM02] 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 in fact a region growing algorithm with multiple sources. The seeds for growing are found based on the definition of a height function on the mesh. The algorithm finds and labels all local minima of this function. Each minimum serves as the initial seed for a surface region. Next, a region is grown incrementally from each seed until it reaches

a ridge or maxima in the function, thus partitioning the function terrain into regions.

The watershed region growing algorithm can be found in many variations, where the main difference between them is the definition of the feature energy or the height function in which "the water rises". For instance, in [ZH04] the average geodesic distance function is used for the height function definition. In [WL97] a simulation of electrical charge distribution over the mesh is used. 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 [MW98, MW99] the function is based on vertex discrete curvature calculations [MDSB02, PRF01]. In [SPP*02] 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 [ZTS02] dihedral angles between faces is used. A more elaborate functional is used in [PKA03] 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 [PAKZ03] this height function is further quantized into discrete values preventing spills from one region to another.

The major drawback in region growing is its dependence on the initial seed selection. Using watershed formulation this is solved by starting at function minima, e.g. in [ZH04] the critical points of the average geodesic distance of the vertices are used as seeds. However, often in practice when a height function cannot be determined, random seed selection is used and may result in bad segmentation. Multiple source region growing is often used also as a sub-routine in the variational approach of iterative clustering (Section 5.3). There, the seed selection problem is alleviated since the seeds are replaced in each iteration to better reflect their cluster. A different approach that lets the data values 'lead' the clustering of segments is given by the hierarchical clustering algorithm.

5.2. 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:

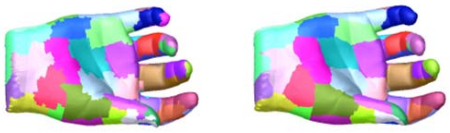

Figure 5: *Raw segmentation results may require post-processing to smooth the boundary between patches (example taken from [SSGH01]).*

```
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 [GWH01] 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. This 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. More recently [AFS06] use a finite set of fitting primitives (planes, spheres, cylinders) and the cost of merging a set of triangles into a single cluster is the minimum of the approximation errors computed against all possible primitives. Segmentation based on slippage analysis [GG04] also uses hierarchical clustering to merge points to larger regions based on slippage similarity scoring.

Charts creation based on hierarchical clustering uses Mean squared distance of a patch to the best fitted plane in [SSGH01]. 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 5).

When working on the dual graph of the mesh such as in [She01], an edge contraction in the graph is equivalent to a merge of two clusters of faces in the original mesh. Hence this is in fact equivalent to hierarchical clustering. The priority of edges used in the algorithm for clustering is a combination of geometric and topological costs including size, shape, curvature and more.

5.3. Iterative Clustering

The two previous methods are often described as non-parametric, as the number of resulting clusters is unknown in advance. In a *parametric* search, the number of clusters is given a-priori. The segmentation can then be formulated as a variational problem of finding the optimal segmentation by iteratively searching for the best segmentation for the given number of clusters. The basis of this approach is the k-means algorithm, sometimes referred to as Lloyd or Lloyd-Max algorithm [Llo82, DHS00]. 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. It is interesting to note that most iterative clustering algorithms on meshes use region growing as a sub-routine. The reason for this is that the elements (faces or vertices) lie on a manifold mesh embedded in 3D. Therefore, one cannot use Euclidean distances between elements to assign an element to a cluster (or a representative of a cluster). Geodesic distances are more appropriate for measuring distances on the mesh. However, calculating geodesic distances on-the-fly is extremely expensive. Therefore, using the representatives as seeds for a region growing algorithm alleviates the computational cost. This also provides the advantage of constraining the clusters to be connected.

To create compatible segmentation of two objects for morphing purposes, a k-means based face-clustering algorithm is proposed in [STK02]. 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.

$$\mathrm{Dist}(F_1,F_2) = (1-\delta)\cos^2(\alpha) + \delta\mathrm{PhysDist}(F_1,F_2)$$

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 [CSAD04] 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.

$$L^2(R_i,P_i) = \int\int_{x\in R_i} ||x-\pi_i(x)||^2 dx$$

A superior metric both in terms of results and in terms of calculation cost and simplicity 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.

$$L^{2,1}(R_i,P_i) = \int\int_{x\in R_i} ||n(x)-n_i||^2 dx$$

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.

An extension of the possible proxies to other surface elements was defined in [WK05] where planes, spheres, cylinders and rolling ball blend patches are used. The motivation for this choice is mainly due to the fact that most technical CAD objects consist of patches from these four categories. For instance, for sphere fitting robust least-square method of [Pra87] is used where the sphere is represented implicitly as:

$$f(x,y,z) = A(x^2+y^2+z^2) + Bx + Cy + Dz + E$$

.

To geometrically fit a cylinder to a region the curvature tensor field is used to determine the direction d_i of the cylinder axis. If the region is indeed anisotropic, the barycenters of the region triangles are projected onto the plane passing through the origin with normal d_i and are fitted with a 2D circle. Since the fitting process for all types of proxies can be time consuming the algorithm progresses by first fitting planes and only then cylinders and spheres and lastly rolling ball blend patches.

A different variation on the iterative clustering algorithm uses quasi-developable patches as proxies in [JKS05]. The detection mechanism is actually narrowed to a subset of developable surfaces, i.e. unions of uni-axial conics. A surface is a union of conics with aligned axes and the same cone angle if and only if the angle between the normal to the surface at every point and a common axis is constant. Hence,

to measure how well a given triangle t with a normal n_t fits into a given developable chart C with normal N_C and angle θ_C, the fitting error is defined as:

$$F(C,t) = (N_C \cdot n_t - \cos\theta_C)^2$$

Mesh charts are also defined in [SWG*03] for geometry image creation using iterative clustering. This algorithm also ensured connectivity by adding only neighboring faces to existing charts. The cost of adding a face 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. N_C: When λ is usually 1:

$$\text{cost}(F,F') = (\lambda - (N_C \cdot N_{F'}))(P_{F'} - P_F)$$

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.

5.4. Spectral Analysis

Spectral graph theory [Chu97, SM00] states the relationship between the combinatorial characteristics of a graph and the algebraic properties of its Laplacian [Got03]. 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 [AY95, Got03].

The Laplacian matrix of the vertex adjacency graph was used for mesh compression purposes in [KG00]. 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 [KK98] graph partitioning application was used.

Using a slightly different formulation in [LZ04] 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. Later, [ZL05] utilize a novel sampling scheme to make effective use of Nyström approximation at a sample size of two. The algorithm also adopts a different optimization criterion, based on part salience [HS97], that is specific for mesh segmentation.

An interesting observation is provided in [ZSGS04] on the properties of spectral analysis of the normalized geodesic distance matrix of vertices on the mesh. The geodesic distance distortion of multi-dimensional scaling to 2D based on spectral analysis is found to give good results also in stretch minimization criterion for parameterizations. This is used to define simultaneous chartification and parametrization of 3D meshes. When the distortion is too large, the mesh is segmented using region growing, where the candidate seed vertices are selected based on the spectral analysis of the geodesic distance matrix.

5.5. Graph Cuts & Image-Based Techniques

Graph cuts have been used extensively on images for segmentation and feature extraction [BJ01, LSTS04]. For the purpose of mesh partitioning, this technique is often used as a refinement step of the borders between two or more clusters on the mesh. This idea has been used in [KT03] to define a hybrid algorithm between iterative clustering and graph cut. At the initial stage iterative clustering is used to create general partitioning. However this partition remains fuzzy around the boundary regions of the segments. The final decomposition is created using graph cut inside the fuzzy region to refine the borders between the segments. The algorithm is also capable of creating hierarchical decomposition by top down binary partitioning.

Another scheme which targets the segment boundaries instead of building the segments by clustering is proposed in [LLS*05] for intelligent scissoring. Following the minima-rule from perception [HR84, HS97], 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.

An approach based on image segmentation is presented in [BM03]. The problem of 3D boundary mesh segmentation is reduced to image segmentation by using geometry images [GGH02] to represent the mesh. The portioning of the image imposes a mesh segmentation in 3D.

5.6. Other Methods

There are several other methods which do not fall into one of the above mentioned classifications. An approach based on skeletonization is proposed in [LTTH01]. 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.

Lastly, fully automatic segmentation remains a hard problem especially since it concerns semantics of shape and form. Therefore, there are several manual segmentation and partitioning techniques found in literature [GSL*99, ZSH00, FKS*04, LLS*04]. Segmentation is often created using cut that define the boundaries between segments either explicitly or by designating some vertices and calculating shortest path between them. More recently These are concerned more with user interface design issues which imitate the physical notion of cutting [SBSCO06].

5.7. Advanced Algorithms

More recently, the basic problem of mesh segmentation has been extended in several directions. For instance, when the object being segmented is flexible or dynamic and may come in various poses such as a human or animal models, it is important that the object's segmentation remain consistent despite the pose changes.

To construct a pose-invariant segmentation, multi-dimensional scaling (MDS) to 3D is used on a coarse approximation of the mesh in [KLT05]. MDS finds an embedding of the mesh into an Euclidean space where Euclidean distances approximate well the geodesic distances between points on the mesh. Often this means that different poses of the same object will map to similar poses using MDS. Later, feature points which are, intuitively, points that reside on tips of prominent components of a given model, are extracted. Each prominent component of the object is defined by one or more of these feature points. Using spherical mirroring, the core of the object is extracted and then the other segments. A final refinement stage uses graph cut to finalize the boundaries of the segments.

A different approach was taken in [SSCO05]. It turns out that the shape-diameter-function (SDF) remains largely consistent through pose changes of the same object and can thus guide pose-invariant segmentations. Using iso-contours of this function on the mesh along with graph cut refinement, a segmentation algorithm is presented. Another direction for extension is the segmentation of multiple meshes compatibly. This enables correspondence between objects, which is important to motion transfer, shape matching or editing. The SDF function [SSCO05] maintains similar values in analogue parts of different objects, allowing correspondence between parts on different objects to be developed using the signature of various parts.

6. 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 surface-type segmentations and part-type segmentations. This difference originates from a different point of view on the object - either 2D surface or 3D object, and is reflected in the segmentation goal. For this reason, it is difficult to assess quality and compare the different results, and we have focused more on extracting and formulating the major algorithmic techniques used to date.

Although there are already numerous techniques for mesh segmentation, it seems that directions to address this problem are only beginning, and we have tried to present possible extensions as well.

Acknowledgments

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

References

[ACK01] AMENTA N., CHOI S., KOLLURI R.: The power crust, unions of balls, and the medial axis transform. *Computational Geometry: Theory and Applications 19*, 2-3 (2001), 127–153.

[ACSD*03] ALLIEZ P., COHEN-STEINER D., DEVILLERS O., LEVY B., DESBRUN M.: Anisotropic polygonal remeshing. *ACM Transactions on Graphics. Special issue for SIGGRAPH conference* (2003), 485–493.

[AFS06] ATTENE M., FALCIDIENO B., SPAGNUOLO M.: Hierarchical mesh segmentation based on fitting primitives. *The Visual Computer 22*, 3 (2006), 181–193.

[AHD96] ARABIE R., HUBERT L., DESOETE G. (Eds.): *Clustering and Classification*. World Scientific Publishers, River Edge, NJ, 1996.

[aim] A.i.m.a.t.s.h.a.p.e. – advanced and innovative models and tools for the development of semantic-based systems for handling, acquiring, and processing knowledge embedded in multidimensional digital objects. http://www.aim-at-shape.net/.

[AKM*06] ATTENE M., KATZ S., MORTARA M., PATANE G., SPAGNUOLO M., A.TAL: Mesh segmentation - a comparative study. In *Proceedings Shape Modeling International (SMI'06)* (2006), p. to appear.

[AY95] ALPERT C., YAO S.: Spectral partitioning: The more eigenvectors, the better. In *32nd ACM/IEEE Design Automation Conference* (San Francisco, 1995), pp. 195–200.

[Bia03] BIASOTTI S.: 3d shape matching through topological structures. In *Discrete Geometry for Computer Imagery* (2003), vol. LNCS 2886, Springer-Verlag, pp. 194–203.

[Bie87] BIEDERMAN I.: Recognition-by-Components: A theory of human image understanding. *Phicological Review 94* (1987), 115–147.

[BJ01] BOYKOV Y., JOLLY M.: Interactive graph cuts for optimal boundary & region segmentation of objects in n-d images.

In *International Conference on Computer Vision (ICCV)* (2001), vol. I, pp. 105–112.

[BM03] BOIER-MARTIN I. M.: Domain decomposition for multiresolution analysis. In *Proceedings of the Eurographics Symposium on Geometry Processing* (2003), pp. 29–40.

[CCM97] CHOI H., CHOI S., MOON H.: Mathematical theory of medial axis transform. *Pacific Journal of Mathematics 181*, 1 (1997), 57–88.

[CDST97] CHAZELLE B., DOBKIN D., SHOURHURA N., TAL A.: Strategies for polyhedral surface decomposition: An experimental study. *Computational Geometry: Theory and Applications 7*, 4-5 (1997), 327–342.

[Chu97] CHUNG F. R. K.: *Spectral Graph Theory*. No. 92 in CBMS Regional Conference Series in Mathematics. American Mathematical Society, 1997.

[CM02] COMANICIU D., MEER P.: Mean shift: A robust approach towards feature space analysis. *IEEE Trans. Pattern Analysis and Machine Intelligence 24* (May 2002), 603–619.

[CSAD04] COHEN-STEINER D., ALLIEZ P., DESBRUN M.: Variational shape approximation. *ACM Trans. Graph. 23*, 3 (2004), 905–914.

[DHS00] DUDA R. O., HART P. E., STORK D. G.: *Pattern Classification (2nd ed.)*. Wiley Interscience, October 2000.

[DKT98] DEROSE T., KASS M., TRUONG T.: Subdivision surfaces in character animation. In *ACM Computer Graphics, Proc. SIGGRAPH 1998* (1998), pp. 85–94.

[DZ02] DEY T. K., ZHAO W.: Approximating the medial axis from the voronoi diagram with a convergence guarantee. In *Algorithms - ESA 2002: 10th Annual European Symposium*. Springer-Verlag Heidelberg, 2002, pp. 387–398.

[EDD*95] ECK M., DEROSE T., DUCHAMP T., HOPPE H., LOUNSBERY M., STUETZLE W.: Multiresolution analysis of arbitrary meshes. In *Proceedings of ACM SIGGRAPH 1995* (1995), pp. 173–182.

[FKS*04] FUNKHOUSER T., KAZHDAN M., SHILANE P., MIN P., KIEFER W., TAL A., RUSINKIEWICZ S., DOBKIN D.: Modeling by example. *ACM Transactions on Graphics (Proceedings SIGGRAPH 2004) 23* (2004), to appear.

[GG04] GELFAND N., GUIBAS L. J.: Shape segmentation using local slippage analysis. In *SGP '04: Proceedings of the 2004 Eurographics/ACM SIGGRAPH symposium on Geometry processing* (New York, NY, USA, 2004), ACM Press, pp. 214–223.

[GGH02] GU X., GORTLER S., HOPPE H.: Geometry images. *ACM Transaction on Graphics, Special issue for SIGGRAPH conference 21*, 3 (2002), 355–361.

[GJS76] GAREY M., JOHNSON D., STOCKMEYER L.: Some simplified np-complete graph problems. *Theoretical Computer Science 1* (1976), 237–267.

[Got03] GOTSMAN C.: On graph partitioning, spectral analysis, and digital mesh processing. In *Proceedings of Shape Modeling International* (Seoul, 2003), pp. 165–169.

[GSL*99] GREGORY A., STATE A., LIN M., MANOCHA D., LIVINGSTON M.: Interactive surface decomposition for polyhedral morphing. *The Visual Computer 15* (1999), 453–470.

[GWH01] GARLAND M., WILLMOTT A., HECKBERT P.: Hierarchical face clustering on polygonal surfaces. In *Proc. ACM Symposium on Interactive 3D Graphics* (March 2001).

[HR84] HOFFMAN D., RICHARDS W.: Parts of recognition. *Cognition 18*, 1-3 (December 1984), 65–96.

[HS97] HOFFMAN D., SIGNH M.: Salience of visual parts. *Cognition 63* (1997), 29–78.

[ITA*01] INOUE K., TAKAYUKI I., ATSUSHI Y., TOMOTAKE F., KENJI S.: Face clustering of a large-scale cad model for surface mesh generation. *Computer Aided Design 33*, 3 (March 2001). The 8th International Meshing Roundtable Special Issue: Advances in Mesh Generation.

[JKS05] JULIUS D., KRAEVOY V., SHEFFER A.: D-charts: Quasi-developable mesh segmentation. *Computer Graphics Forum (Proceedings Eurographics 2005) 24*, 3 (2005), 981–990.

[KG00] KARNI Z., GOTSMAN C.: Spectral compression of mesh geometry. In *Proceedings of ACM SIGGRAPH 2000* (2000), pp. 279–286.

[KK98] KARYPIS G., KUMAR V.: 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.

[KK99] KARYPIS G., KUMAR V.: Multilevel algorithms for multi-constraint graph partitioning. In *Proceedings of the 36th ACM/IEEE conference on Design automation conference* (New Orleans, Louisiana, 1999), pp. 343 – 348.

[KLT05] KATZ S., LEIFMAN G., TAL A.: Mesh segmentation using feature point and core extraction. *The Visual Computer(Pacific Graphics) 21*, 8–10 (2005), 865–875.

[KS98] KIMMEL R., SETHIAN J.: Fast marching methods on triangulated domains. 8341–8435.

[KS04] KRAEVOY V., SHEFFER A.: Cross-parameterization. *ACM Transaction on Graphics (Proceedings SIGGRAPH 2004) 23*, 3 (2004), 861Ű–869.

[KT96] KALVIN A., TAYLOR R.: Superfaces: Polygonal mesh simplification with bounded error. *IEEE Computer Graphics and Applications 16*, 3 (1996).

[KT03] KATZ S., TAL A.: Hierarchical mesh decomposition using fuzzy clustering and cuts. *ACM Transactions on Graphics (Proceedings SIGGRAPH 2003) 22*, 3 (2003), 954–961.

[Llo82] LLOYD S.: Least square quantization in pcm. *IEEE Transactions on Information Theory 28* (1982), 129–137.

[LLS*04] LEE Y., LEE S., SHAMIR A., COHEN-OR D., SEIDEL H.-P.: Intelligent mesh scissoring using 3D snakes. In *Proceedings of the 12th Pacific Conference on Computer Graphics and Applications* (2004), pp. 279–287.

[LLS*05] LEE Y., LEE S., SHAMIR A., COHEN-OR D., SEIDEL H.-P.: Mesh scissoring with minima rule and part salience. *Computer Aided Geometric Design 22*, 5 (July 2005), 444–465.

[LPRM02] LEVY B., PETITJEAN S., RAY N., MAILLOT J.: Least squares conformal maps for automatic texture atlas generation. In *ACM Computer Graphics, Proc. SIGGRAPH 2002* (2002), pp. 362–371.

[LSTS04] LI Y., SUN J., TANG C.-K., SHUM H.-Y.: Lazy snapping. *ACM Trans. Graph. 23*, 3 (2004), 303–308.

[LTTH01] LI X., TOON T., TAN T., HUANG Z.: Decomposing polygon meshes for interactive applications. In *Proceedings of the 2001 symposium on Interactive 3D graphics* (2001), pp. 35–42.

[LZ04] LIU R., ZHANG H.: Segmentation of 3D meshes through spectral clustering. In *The 12th Pacific Conference on Computer Graphics and Applications (PG'04)* (Seoul, Korea, 2004), pp. 298–305.

[MDSB02] MEYER M., DESBURN M., SCHRÖDER P., BARR A. H.: Discrete differential - geometry operators for triangulated 2-manifolds. In *Proceedings VisMath '02* (Berlin, 2002).

[MK01] MOULITSAS I., KARYPIS G.: Multilevel algorithms for generating coarse grids for multigrid methods. In *Proceedings of the 2001 ACM/IEEE conference on Supercomputing* (Denver, Colorado, 2001), pp. 45–45.

[MPS*03] MORTARA M., PATANÈ G., SPAGNUOLO M., FALCIDIENO B., ROSSIGNAC J.: Blowing bubbles for multi-scale analysis and decomposition of triangle meshes. *Algorithmica 38*, 1 (2003), 227–248.

[MPS*04] MORTARA M., PATANÈ G., SPAGNUOLO M., FALCIDIENO B., ROSSIGNAC J.: Plumber: A method for a multi-scale decomposition of 3d shapes into tubular primitives and bodies. In *Proceedings of ACM Symposium on Solid Modeling and Applications* (2004), pp. 139–158.

[MS04] MITANI J., SUZUKI H.: Making papercraft toys from meshes using strip-based approximate unfolding. *ACM Transaction on Graphics (Proceedings SIGGRAPH 2004) 23*, 3 (2004), 259–263.

[MW98] MANGAN A. P., WHITAKER R. T.: Surface segmentation using morphological watersheds. In *Proc. IEEE Visualization 1998 Late Breaking Hot Topics* (1998).

[MW99] MANGAN A., WHITAKER R.: Partitioning 3D surface meshes using watershed segmentation. *IEEE Transactions on Visualization and Computer Graphics 5*, 4 (1999), 308–321.

[NN99] NIKOS C., NAVE D.: Simultaneous mesh generation and partitioning for delaunay meshes. In *Proceedings of the 8th International Meshing Roundtable* (South Lake Tahoe, 1999), pp. 55–66.

[PAKZ03] PAGE D., ABIDI M., KOSCHAN A., ZHANG Y.: 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* (2003), pp. 91–97.

[PKA03] PAGE D., KOSCHAN A., ABIDI M.: Perception-based 3D triangle mesh segmentation using fast marching watersheds. In *Conference on Computer Vision and Pattern Recognition (CVPR '03) - Volume II* (2003), pp. 27–32.

[Pra87] PRATT V.: Direct least-squares fitting of algebraic surfaces. *Computer Graphics (SIGGRAPH 1987) 21*, 4 (1987), 145–152.

[PRF01] PULLA S., RAZDAN A., FARIN G.: Improved curvature estimation for watershed segmentation of 3-dimensional meshes. manuscript, 2001.

[RGS04] RAAB R., GOTSMAN C., SHEFFER A.: Virtual woodwork: Generating bead figures from 3d models. *International Journal on Shape Modeling 10*, 1 (2004), 1–30.

[Rob97] ROBERTS S. J.: Parametric and non-parametric unsupervised cluster analysis. *Pattern Recognistion 30* (1997), 327–345.

[SAKJ01] SHAH J. J., ANDERSON D., KIM Y. S., JOSHI S.: A discourse on geometric feature recognition from cad models. *J. Comput. Info. Sci. Eng. 1*, 1 (2001), 41–51.

[SAPH04] SCHREINER J., ASIRVATHAM A., PRAUN E., HOPPE H.: Inter-surface mapping. *ACM Transaction on Graphics (Proceedings SIGGRAPH 2004) 23*, 3 (2004), 870–877.

[SBSCO06] SHARF A., BLUMENKRANTS M., SHAMIR A., COHEN-OR D.: Snap-paste: An interactive technique for easy mesh composition. *The Visual Computer (Proceedings Pacific Graphics 2006) to appear* (2006).

[SCOGL02] SORKINE O., COHEN-OR D., GOLDENTHAL R., LISCHINSKI D.: Bounded-distortion piecewise mesh parameterization. In *Proceedings of IEEE Visualization 2002* (2002).

[She01] SHEFFER A.: Model simplification for meshing using face clustering. *Computer Aided Design 33* (2001), 925–934.

[SM00] SHI J., MALIK J.: Normalized cuts and image segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence 22*, 8 (2000), 888–905.

[SPP*02] SUN Y., PAGE D. L., PAIK J. K., KOSCHAN A., ABIDI M. A.: 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., 2002), pp. 825–828.

[SSCO05] SHAPIRA L., SHAMIR A., COHEN-OR D.: *Consistent Partitioning of Meshes*. Tech. rep., The interdisciplinary Center, 2005.

[SSGH01] SANDER P., SNYDER J., GORTLER S., HOPPE H.: Texture mapping progressive meshes. In *Proceedings of ACM SIGGRAPH* (2001), pp. 409–416.

[STK02] SHLAFMAN S., TAL A., KATZ S.: Metamorphosis of polyhedral surfaces using decomposition. *Computer Graphics forum 21*, 3 (2002). Proceedings Eurographics 2002.

[STL06] SHATZ I., TAL A., LEIFMAN G.: Paper craft models from meshes. *The Visual Computer (Proceedings Pacific Graphics 2006) to appear* (2006).

[SWG*03] SANDER P., WOOD Z., GORTLER S., SNYDER J., HOPPE H.: Multi-chart geometry images. In *Proceedings of the Eurographics Symposium on Geometry Processing* (2003), pp. 146–155.

[TM98] TOMASI C., MANDUCHI R.: Bilateral filtering for gray and color images. In *Proc. of the sixth international Conference of Computer Vision* (1998), pp. 839–846.

[WK05] WU J., KOBBELT L.: Structure recovery via hybrid variational surface approximation. *Computer Graphics Forum (Proceedings Eurographics 2005) 24*, 3 (2005), 277–284.

[WL97] WU K., LEVINE M.: 3D part segmentation using simulated electrical charge distributions. *IEEE transactions on pattern analysis and machine intelligence 19*, 11 (1997), 1223–1235.

[WML*06] WU F.-C., MA W.-C., LIANG R.-H., CHEN B.-Y., OUHYOUNG M.: Domain connected graph: the skeleton of a closed 3d shape for animation. *The Visual Computer 22*, 2 (2006), 117–135.

[ZH04] ZHOU Y., HUANG Z.: Decomposing polygon meshes by means of critical points. In *MMM '04: Proceedings of the 10th International Multimedia Modelling Conference* (Washington, DC, USA, 2004), IEEE Computer Society, p. 187.

[ZHPZ96] ZHUANG X., HUANG Y., PALANIAPPAN K., ZHAO Y.: Gaussian mixture density modeling: Decomposition and applications. *IEEE Transactions on Image Processing 5*, 9 (September 1996), 1293–1302.

[ZL05] ZHANG H., LIU R.: Mesh segmentation via recursive and visually salient spectral cuts. In *Proceedings of Vision, Modeling, and Visualization* (November 2005), pp. 429–436.

[ZMT05] ZHANG E., MISCHAIKOW K., TURK G.: Feature-based surface parameterization and texture mapping. *ACM Transaction on Graphics 24*, 1 (2005), 1–27.

[ZSGS04] ZHOU K., SYNDER J., GUO B., SHUM H.-Y.: Isocharts: stretch-driven mesh parameterization using spectral analysis. In *SGP '04: Proceedings of the 2004 Eurographics/ACM SIGGRAPH symposium on Geometry processing* (New York, NY, USA, 2004), ACM Press, pp. 45–54.

[ZSH00] ZOCKLER M., STALLING D., HEGE H.-C.: Fast and intuitive generation of geometric shape transitions. *The Visual Computer 16*, 5 (2000), 241–253.

[ZTS02] ZUCKERBERGER E., TAL A., SHLAFMAN S.: Polyhedral surface decomposition with applications. *Computers & Graphics 26*, 5 (2002), 733–743.