DOI: 10.1111/j.1467-8659.2007.01103.x

COMPUTER GRAPHICS *forum*
Volume 27 (2008), number 6 pp. 1539–1556

A survey on Mesh Segmentation Techniques

Ariel Shamir

Efi Arazi School of Computer Science, The Interdisciplinary Centre, Herzliya, Israel

Abstract

We present a review of the state of the art of segmentation and partitioning techniques of boundary meshes. Recently, these have become a part of many mesh and object manipulation algorithms in computer graphics, geometric modelling and computer aided design. We formulate the segmentation problem as an optimization problem and identify two primarily distinct types of mesh segmentation, namely part *segmentation and* surface-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 some generic algorithms for the major segmentation techniques.*

Keywords: mesh segmentation, mesh partioning, clustering

ACM CCS: 1.3.5 [Computing Methodologies]: Computer Graphics Computational Geometry and Object Modeling; I.3.6 [Computing Methodologies]: Computer Graphics Methodology and Techniques

1. Introduction

Mesh segmentation (or mesh partitioning) has become a key ingredient in many geometric modelling and computer graphics tasks and applications. Segmentation assists parametrization, texture mapping, shape matching, morphing, multi-resolution modelling, mesh editing, compression, animation and more. Moreover, shape understanding and semantics based object representation must rely on feature extraction and structure extraction from 3D meshes that represent these objects and shapes (see e.g. [Aim]).

Techniques developed for segmentation borrow from related fields such as image segmentation, finite element meshes partitioning, unsupervised machine learning and others. In this report, we survey the different techniques used for various purposes, and illustrate how they can be classified into a small set of generic algorithms. This provides better understanding as to the strengths and weaknesses of each technique and can assist in future choices for different applications.

As there is not a single criterion to evaluate mesh segmentation results, our attempt is to formulate the segmentation problem as an optimization problem [Sha04] using different criteria for different applications to define its energy. These criteria are based on various mesh properties or *features*, that are often extracted prior to the process of segmentation. They include simple surface measures such as area, size or length, various differential properties such as curvature and normal direction, some distance measures such as geodesic distances, distance to the medial axis, or the shape diameter, and more. We survey some of these in Section 3.

The quality of segmentation is often application dependent. In fact, we distinguish between two general types of mesh segmentation which are inherently different (Section 4). In *part-type* segmentation, the goal is to segment the object represented by the mesh into meaningful, mostly volumetric, parts, and in *surface-type* segmentation the objective is to partition the surface mesh into patches under some criteria (Figure 5).

Using our formulation, we illustrate how the different algorithms used for segmentation can be cast as various approximation techniques for optimization. We classify these algorithms to several approaches and provide the link to general clustering algorithms (Section 5).

2. Preliminaries

A three dimensional boundary mesh M is definedas a tuple $\{V, E, F\}$ of *vertices* $V = \{p_i \mid p_i \in \mathbb{R}^3, 1 \leq i \leq m\}$, *edges*

Submitted October 2006
Revised May 2007
Accepted September 2007

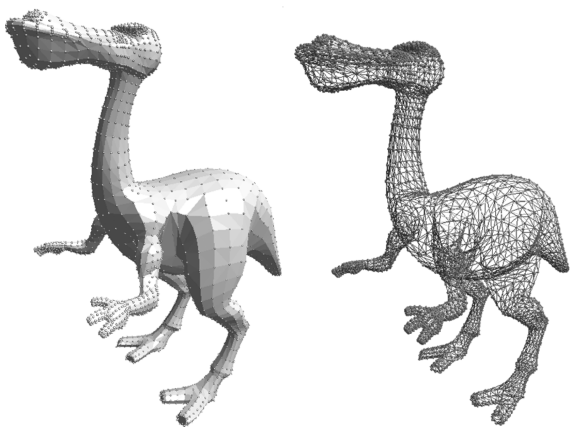

Figure 1: *Vertices, faces and edges in a 3D boundary mesh.*

$E = \{e_{ij} = (p_i, p_j) \mid p_i, p_j \in V, i \neq j\}$, and *faces* F, which are usually triangles $F = \{f_{ijk} = (p_i, p_j, p_k) \mid p_i, p_j, p_k \in V, i \neq j, i \neq k, j \neq k\}$, 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 used in simulations), 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 of the mesh (i.e. vertices, edges and faces), which impose a valid representation. For example, in a two-manifold mesh the neighbourhood of every point which lays on the mesh is homeomorphic to a disk. In watertight meshes, the mesh will not contain any boundary edges. Generally, we will restrict our discussion to two-manifold watertight mesh representation, although some of the techniques reviewed do not directly depend on such constraints for correctness.

Using a sub-set of elements from the faces, edges or vertices, an induced sub-mesh $M' \subset M$ can be created as follows. Let M be a 3D boundary-mesh, and S the set of mesh elements which is either V, E or F. Let $S' \subset S$ be a sub-set of mesh elements, and let V' be the set of all vertices which are included in (or are) the elements in S'. A *sub-mesh* M' is defined as the mesh $M' = \{V', E', F'\}$, where $E' = \{(p_i, p_j) \in E \mid p_i, p_j \in V'\}$ are all edges in which both vertices are a part of V', and $F' = \{(p_i, p_j, p_k) \in F \mid p_i, p_j, p_k \in V'\}$ are all faces in which all vertices are a part of V'.

Our basic definition of a mesh segmentation is therefore:

> *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.

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. Segmentation algorithms usually partition the faces of the mesh (i.e. $S = F$), some partition the vertices ($S = V$), and few the edges ($S = E$). Note that if $S = V$ or $S = E$ then some faces (that include vertices from different parts of S) will not be part of any sub-mesh M_i, and must be joined to one of the adjacent parts.

The key question in all mesh segmentation problems is how to partition the set S. Obviously, this relies heavily on the application in mind. We pose the segmentation problem as an optimization problem by defining a criterion function of the partitioning of S, $J : 2^S \rightarrow R$ for each application 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.

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 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. General clustering in statistics often involve points in higher dimensions representing abstract notions which are non-geometric, and do not hold any explicit connectivity relation, hence, they 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 [Del99]. 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,

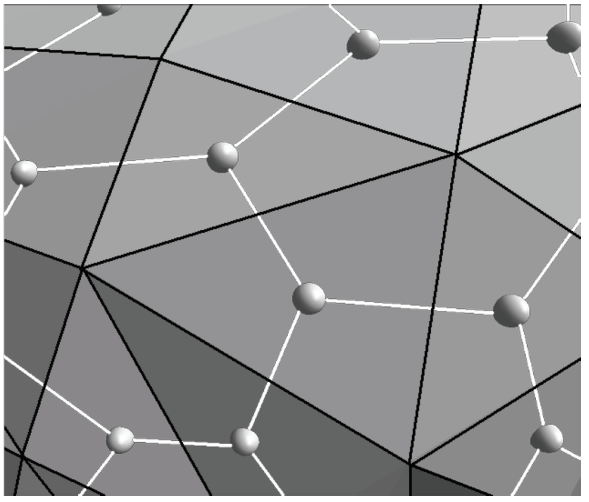

Figure 2: *Part of the face-adjacency dual-graph of a mesh.*

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.

Partitioning of a general graph into approximately equal subsets of nodes so that the number of cut edges between the subsets is minimized is NP complete [GJS76]. Although meshes are not general graphs, and the objective functions that we present below are different than equal-size partitioning, if we use the dual representation, the mesh segmentation problem can be cast as a (constrained) graph partitioning problem. This analogy can hint on the complexity of mesh segmentation. In fact, under certain conditions (that the patches are convex) it is shown in [CDST97] that the problem is indeed NP complete. In general, 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 major 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 and
5. Implicit methods.

In the following, 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. Some techniques use a combination of these approaches, while others do not fit into these categories, and we will try to signify those in the text.

It is important to note that although we distinguish between two major types of segmentations, namely part-type and patch-type, there is no technique which is more suitable for one or the other. Almost all the techniques presented were used to achieve both types of segmentation. Therefore, in our review we have tried to detach the technique from the goal of segmentation and promote two orthogonal views on the subject: the segmentation objective (Section 4) and the segmentation technique (Section 5).

We have also identified a number of geometric attributes and partitioning criteria that are commonly used by many segmentation techniques. The decision which attribute to use has a significant effect on the segmentation results and is strongly linked to the goal of segmentation. We therefore begin with an overview of the possible attributes and constraints used in segmentation algorithms in the next section.

3. 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 choice of constraints imposed on the partitioning process. These criteria are usually based on attributes extracted from the mesh a priori. Hence, we present them independently of the algorithms that use them, and of 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.

3.1. Constraints

There are three major types of possible constraints for segmentation: *cardinality* constraints, *geometric* constraints and *topological* constraints.

3.1.1. *Cardinality constraints*

Some typical cardinality constraints regard the set of partition elements S:

- A bound on the maximum and/or minimum number of elements in each part S_i. 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 (i.e. on |S|) may also be used to balance the partition.

Table 1: *The list of abbreviations used in Table 2.*

Techniques:	Attributes:
RG: Region grow (Algorithm 5.1)	Lnr = Linear planar characteristics.
MG: Multiple source region grow (Algorithm 5.2)	Qdr = Quadrics & primitives of degree >1.
WS: Water-shed (also Algorithm 5.2)	Ang = Dihedral angles or normal angles.
HR: Hierarchical clustering (Algorithm 5.3)	Crv = Curvature based.
TD: Top-down hierarchical (Algorithm 6.2)	Gds = Geodesic distances.
IT: Iterative clustering (Algorithm 5.4)	Skl = Skeleton based and related.
SP: Spectral analysis methods (Section 5.5)	Top = Topological.
IM: Implicit methods (Section 6)	Cnv = Convexity.
GC: Graph-cut (Section 6)	Par = Parametrization distortion.
SK: Inferred from a skeleton (Section 6.3)	Mtn = Motion characteristics.
	Bub = Intersection of the surface with a sphere.
Segmentation type:	Elc. = Electrical charge simulation.
P = Part-type	Slp = Slippage.
S = Surface-type	Sym = Symmetry.

3.1.2. *Geometric constraints*

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 such as convexity of either 2D patch or volumetric 3D part and
- Soft constraints in the form of a bias towards specific shapes. For instance, maximum or minimum ratio of diameter or perimeter to the area of the sub-mesh can provide a bias towards compact, roundly shaped sub-meshes.

3.1.3. *Topological constraints*

Topological constraints are also used to restrict the sub-mesh shape:

- Restriction of each S_i to be topologically equivalent to a disk.
- Restriction of each S_i to be a single connected component.

3.2. Mesh attributes

The function used in the optimization process is defined using specific mesh attributes that depends on the application in mind. We mention the following frequently used attributes for partitioning (Table 2 summarizes which attributes are used by each technique):

1. Planarity of various forms,
2. Higher degree geometric proxies (spheres, cylinders, cones, Quadrics, developable surfaces),
3. Difference in normals of vertices or dihedral angles between faces,
4. Curvature,
5. Geodesic distances on the mesh,
6. Slippage,
7. Symmetry,
8. Convexity,
9. Medial axis and shape diameter and
10. Motion characteristics.

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: $|(v_x, v_y, v_z, 1) \cdot (a, b, c, d)| \leq \epsilon$.
> L_2 *distance norm:* given a cluster representative plane, and a set of vertices v_i it measures the average distance from plane: $\frac{1}{k}\sum_{i=1}^{k}((v_x, v_y, v_z, 1)_i \cdot (a, b, c, d))^2 \leq \epsilon$.
> 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: $(1 - (n_x, n_y, n_z) \cdot (a, b, c)) \leq \epsilon$.
> 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: $\frac{1}{A}\sum_{i=1}^{k}\frac{1}{A_i}(1 - (n_x, n_y, n_z)_i \cdot (a, b, c)) \leq \epsilon$, 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

Table 2: *Summary of automatic segmentation techniques (the abbreviations are summarized in Table 1).*

Reference	Seg. type	Technique	Attributes										
			Lnr	Qdr	Ang	Crv	Gds	Skl	Top	Cnv	Par	Mtn	Other
[KT96]	S	RG	x		x								
[LDB05]	S	RG				x							
[CDST97]	S	RG								x			
[KJS06]	P	RG								x			
[LLYL05]	P	RG										x	
[ZMT05]	S	RG					x		x		x		
[EDD*95]	S	MG					x		x				
[SCOGL02]	S	MG							x		x		
[MPS*04]	P	MG				x		x	x				Bub
[LPRM02]	S	IM+MG			x		x		x		x		
[ZH04]	S/P	WS					x						
[WL97]	S/P	WS											Elc
[MW98]	S/P	WS				x							
[MW99]	S/P	WS				x							
[LDB05]	S/P	WS				x							
[SPP*02]	P	WS			x								
[ZTS02]	P/S	WS			x								
[PKA03]	S	WS			x	x							
[PAKZ03]	S	WS		x	x	x							
[GWH01]	S	HR	x										
[AFS06]	P	HR		x									
[ITA*01]	S	HR	x		x								
[GG04]	S/P	HR											Slp
[SSGH01]	S	HR	x						x		x		
[She01]	S/P	HR			x	x			x				
[GFW*06]	P	TD+IT										x	
[SSK05]	P	IT										x	
[STK02]	P	IT			x		x						
[CSAD04]	S	IT	x		x								
[WK05]	P	IT		x									
[JKS05]	P	IT		x									
[SWG*03]	S	IT			x								
[KT03]	P	IT+ GC			x		x						
[PSG*06]	P	IT+GC											Sym
[STL06]	S	IT		x					x				
[KG00]	P	SP					x						
[LZ04]	P	SP					x						
[ZL05]	P	SP					x						
[LZ07]	P	SP				x	x						
[ZSGS04]	S	SP					x				x		
[LLS*05]	P	IM				x							
[BM03]	S	IM											
[KS04]	S	IM					x		x				
[SAPH04]	S	IM					x		x				
[MS04]	S	IM			x		x						
[SSCO05]	P	GC						x					
[KLT05]	P	IM+GC					x			x			
[LKA06]	P	TD+SK						x	x	x			
[LTTH01]	P	SK						x	x				
[RGS04]	P	SK						x	x				
[WML*06]	P/S	SK						x					

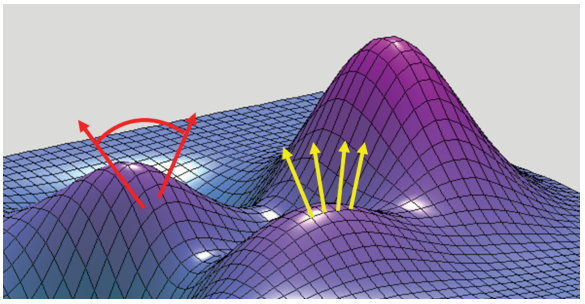

Figure 3: *Differences between normal directions of vertexes (right, yellow vectors) and dihedral angles, which is the angle between faces on the mesh (left, red angle) are differential geometric properties often used for partitioning the mesh.*

use primitives such as *spheres, cylinders* and *cones*, and try to find the best fitting primitive in least squares sense. Other types of regions include rolling ball blends, triangle strips and cones as quasi-developable surfaces (see Section 5.4 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 (Figure 3). Depending on the tolerance of this difference, either planar or curved parts can be created.

Two of the most useful functions in various segmentation algorithms are surface properties of the mesh. The first is a differential property of the mesh-curvature (Figure 4, left), while the second, average geodesic distance (AGD), depends more on global geometry and topology (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, CSM03. 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 the points in the centre 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].

A different approach presented in [GG04] is *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:

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

This equation leads to least-squares problem whose minimum is the solution of a linear system. Hence, the slippable motions of a local neighbourhood of a point can be determined by computing its eigenvalues. Local points and regions having similar slippable motions are clustered to form segments.

In a similar manner, *symmetry* analysis has been used in [PSG*06] to segment a mesh into components. First the m major symmetry planes of an object are found based on sampling. Next, for each face and for every symmetry planes, a measure of the degree to which the face contributes to the symmetry with respect to that plane is given. Hence, every face can now be described using these m values as a feature vector and clustered to segments.

As discussed in [HR84], volumetric *convexity* (or concavity) is often related to shape segmentation. The basic problem of convex decomposition of polyhedra has been addressed early on in [Cha81, BD92, CP94]. However, such decompositions can be costly to construct and can result in representations with an unmanageable number of components. Some approximation measure for convexity were defined to create more effective decompositions. An approximate convex decomposition (ACD) measures the concavity, i.e. the volume ratio between the actual part and it's convex hull, in [LA06], while [KJS06] measures the average distance from all part's triangles to the part's convex hull.

The *medial axis* and medial axis transform (*MAT*) is an important topological attribute 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. Similarly, various curve skeletons [CSM07] have been used to interpret the shape of objects. In fact, skeletonization and segmentation are often complemental, segmenting the object can guide skeleton extraction [KT03] and having a skeleton can guide segmentation.

An attribute related to the MAT 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). The SDF gives a good distinction between thick and thin parts of the object and thus has been used for part-type partitioning. Another advantage of the SDF is its pose-oblivious nature: the SDF values of points on the mesh remain largely invariant under pose changes of the object. Hence, it has been used to

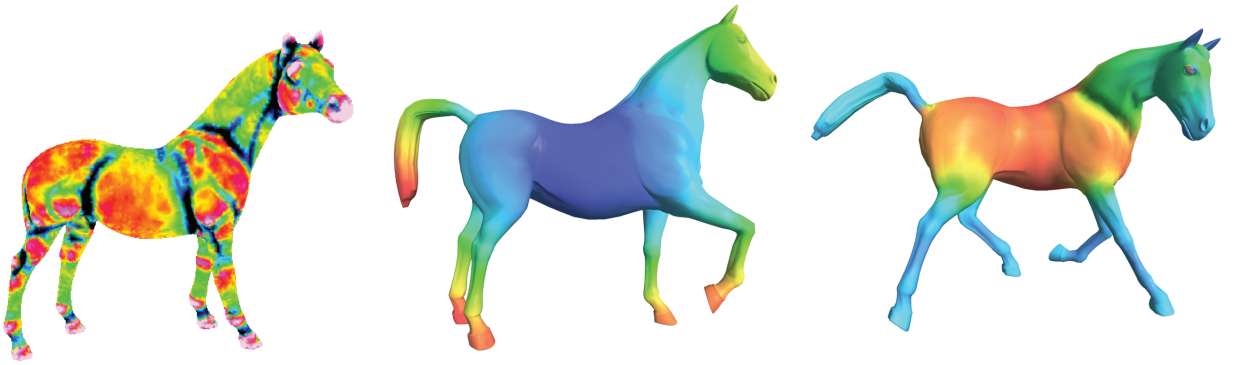

Figure 4: *Examples of mesh attributes used for partitioning. Left: minimum curvature, middle: average geodesic distance (AGD), right: shape diameter function (SDF).*

consistently partition meshes representing dynamically moving objects (see Section 7.1).

Lastly, when the object or mesh being segmented are dynamic or animated, several works have used *motion* characteristics of vertices for segmentation. This has been used in an animation sequence for compression [LLYL05,SSK05] or for ray-tracing acceleration [GFW*06].

4. Segmentation type and objectives

Although there are various segmentation objectives, we have found that there is a distinction between two 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 5). 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, MPS06], 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. Nevertheless, there are times when 'semantic' components are used by surface-type segmentation, e.g. in CAD oriented segmentations and reverse engineering [SAKJ01, IR05, VMC97], 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 meaningful parts, such as minimum curvature guiding 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 5: *Two different types of mesh segmentation: part-type segmentation (left, taken from [LLS*05]) and surface-type segmentation (right, taken from [SSGH01])*

4.1. 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]. Segmentation into general quadric surfaces is often sought in CAD for reverse engineering and

modelling [Pet02, LDB05]. 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, LDB05].

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 use planar patches, collision-detection, where bounding boxes are used on whole sub-mesh patches for efficiency [GWH01], animation with subdivision surfaces [DKT98], and ray-tracing accelerations of animated sequences [GFW*06].

4.2. 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. A recent comparative study on the results of some part-type segmentation technique can be found in [AKM*06].

In [MPS*03, MPS*04], part-type semantic segmentation is created based on analyzing the intersection curves of a ball centred 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 modelling 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, Pet02]. Such part matching can also be utilized for morphing [STK02]. Object part decomposition has also facilitated object skeleton creation [MPS*03, KT03, WML*06, MPS06, LKA06], 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].

5. Segmentation techniques

In this section, we review previous mesh segmentation algorithms. These algorithms find an approximation for the segmentation optimization problem. Hence, we classify them according to the approximation technique used to reach a solution. In fact, the basic segmentation problem can be viewed as the problem of assigning primitive mesh elements to sub meshes, which is similar to classic clustering problems of various elements into groups or clusters (sub-meshes). Consequently, the different algorithms used for mesh segmentation can be classified as variants of classic clustering algorithms.

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:

```
5.1 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 neighbours to Q
Merge small clusters into neighbouring ones
```

The main difference between various algorithms which 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 neighbouring 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. [LDB05] uses curvature as the criteria for growing constant curvature clusters. 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. In the initial stage of [KJS06] approximately-convex parts are extracted from the model by growing patches from seed triangles measuring convexity and compactness.

5.2. Multiple source region grow

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.

```
5.2 Multiple Source Region Grow Algorithm
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 neighbours s_i of s_k
        insert < s_i, C_k > to Q
Merge small clusters into neighbouring 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 grow the region inward. This also simplifies the test criteria which determines if an element can be added to an existing cluster because 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 (watersheds).

The watershed region growing algorithm is found in many variations, where the main difference between them is the definition of the feature energy or the height function inside which 'water rises'. For instance, in [ZH04] the AGD 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], and in [LDB05] on the analysis of the curvature tensor. In [SPP*02] the algorithm approximates the feature strength of each vertex based on 'normal-voting', i.e. the surface normal variation within a neighbourhood 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_{min} 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 AGD 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 5.4). There, the seed selection problem is alleviated because 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.

5.3. Hierarchical clustering

The search for local optimum of each region separately in region grow techniques 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' because it always chooses the best merging operation for all clusters and doesnot concentrates on growing one:

```
5.3 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 initializes each face with its own separate cluster. During clustering, each pair of clusters are assigned a cost for merging them to one cluster and the lowest cost pair is merged. 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 remains homeomorphic to a disk. In post-processing smooth boundaries between the charts are created calculating constrained shortest path (Figure 6).

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.

5.4. Iterative clustering

The two previous methods are often described as non-parametric, as the number of resulting clusters is unknown

Figure 6: *Raw segmentation results may require post-processing to smooth the boundary between patches (example taken from [SSGH01]).*

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:

```
5.4 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 two faces f_1 and f_2 is defined as a weighted combination of the

difference in the dihedral angle α between the faces, and **PhysDist**, the approximate geodesic distance. **PhysDist** (f_1, f_2) is defined as the sum of distance from the centroid of each face to the centre of their shared edge:

$$\text{Dist}(f_1, f_2) = (1-\delta)\cos^2(\alpha) + \delta\,\textbf{PhysDist}(f_1, f_2). \quad (2)$$

After the 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 R_i of the squared error between point on the patch and its planar proxy P_i. The point-difference is the distance between the point on the patch $x \in R_i$ and its orthogonal projection on the proxy $\pi_i(x) \in R_i$:

$$L^2(R_i, P_i) = \int\int_{x \in R_i} ||x - \pi_i(x)||^2 dx. \quad (3)$$

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 $n(x)$ of a point on the patch $x \in R_i$ and the proxy normal n_i:

$$L^{2,1}(R_i, P_i) = \int\int_{x \in R_i} ||n(x) - n_i||^2 dx. \quad (4)$$

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-squares method of [Pra87] is used where the sphere is represented implicitly as:

$$f(x, y, z) = A(x^2 + y^2 + z^2) + Bx + Cy + Dz + E. \quad (5)$$

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. As 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:

$$F(C, t) = (N_C \cdot n_t - \cos(\theta_C))^2. \quad (6)$$

Mesh charts are also defined in [SWG*03] for geometry image creation using iterative clustering. This algorithm also ensured connectivity by adding only neighbouring faces to existing charts. The cost of adding a face F' is a measure of geometric distance between the face and its neighbouring face F in the chart $(P_{F'} - P_F)$, and the difference between the face normal $N(F')$ and the chart normal N_C, when λ is usually 1:

$$\text{cost}(F, F') = (\lambda - (N_C \cdot N_{F'}))(P_{F'} - P_F). \quad (7)$$

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.

5.5. Spectral analysis

Spectral graph theory [Chu97, SM00] states the relationship between the combinatorial characteristics of a graph and the algebraic properties of its Laplacian. 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 submeshes and each one treated separately. However, these submeshes 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. More recently, these works have been extended in [LZ07] where the outer contour of the 2D spectral embedding of the mesh (or sub-mesh) is used to guide the segmentation. Following a distinction made between *structural segmentability* and *geometrical segmentatbility*, two different operators are used for the spectral projection. For structural segmentability the affinity matrix W_{ij} used is simply the graph adjacency:

$$W_{ij} = \begin{cases} 1 & \text{if} e_{ij} \in E \\ 0 & \text{otherwize,} \end{cases} \tag{8}$$

while for geometrical segmenatbility the affinity matrix used relates to concavities on the mesh and uses the minimum principal curvature. For each two vertices i, j if they are not connected then $W_{ij} = 0$, else $e_{ij} \in E$ and then:

$$W_{ij} = \begin{cases} (|\overrightarrow{\kappa_i}| + |\overrightarrow{\kappa_j}|) \cdot |\langle \overrightarrow{e}, \overrightarrow{\kappa} \rangle \cdot l \text{ if } \kappa_i < 0 \text{ or } \kappa_j < 0 \\ \epsilon \qquad\qquad \text{otherwize} \end{cases} \tag{9}$$

where, $\overrightarrow{\kappa_i}$ and $\overrightarrow{\kappa_j}$ are the minimal principal curvatures at vertices i and j, $\overrightarrow{e}$ is the direction of the edge e_{ij}, and l is the normalized length of the edge. Consequently, mesh vertices from continuous concave region will be close in the embedding space.

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.

6. Implicit methods

Some segmentation methods do not directly partition the set S of elements, but rather define the boundaries between subsets of S (or sub-meshes). Others infer a partitioning of the mesh based on a partitioning of a different structure or object such as the skeleton or an image representing the mesh. Using these approaches, the partitioning of S is created *implicitly*.

6.1. Constructing boundaries

Following the minima-rule from perception [HR84,HS97], minimum curvature feature-contours are extracted from the mesh in [LLS*05]. These contours are then closed to form loops around mesh parts for intelligent scissoring. Finally snakes are used to smooth the cuts which define a part-type segmentation of the object. Texture atlas generation in [LPRM02] uses sharp edges to define feature lines and grow charts inward from these lines. Similarly, feature-lines are first extracted and closed in [MS04] to segment the mesh. These segments are then simplified to create triangle strips that support the creation of real paper-craft toys.

To define cross-parametrization or mapping between two objects, a common partitioning is built implicitly by designating corresponding vertices and building compatible paths between them in [SAPH04, KS04].

6.2. Top down approach

Hierarchical clustering (Section 5.3) can be viewed as the method of bottom up construction of a tree. The process continues to merge clusters until one root is created representing the whole model. An opposite approach could be used to create a similar hierarchical structure. Starting from one root representing the whole object, a segmentation is created by partitioning it into two (or more) parts. This process continues in each part recursively until a certain tolerance is met for a part, or until the desired number of levels or parts is reached.

```
6.2 Top Down Partitioning Algorithm
Create a root set  S_r  including all elements
Insert S_r to a priority queue Q
Loop until Q is empty
    Get the top set S from Q
    If S can be split
      Split S into {S_i}
      Insert all S_i to Q
```

Each partitioning in the top-down approach is often achieved implicitly by finding the best boundary between parts. This idea has been used in [KT03, PSG*06 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 can also create a single level segmentation by using multi-way cut.

In [LKA06] such a scheme is used to create both a skeleton and a partitioning of 3D models simultaneously. The tolerance threshold for the recursive partitioning measures the quality of the approximated skeleton and a concavity

measure of the object. The partitioning is created by searching for the best path that cuts the object to create an ACD.

Note that graph cuts have been used extensively for image segmentation and feature extraction [BJ01, LSTS04]. For the purpose of mesh partitioning, graph cut is often used as a post-processing step no matter what initial segmentation algorithm was used, to smooth the borders between two or more segments on the mesh.

6.3. Inferring

As stated earlier, part-type segmentation and shape skeleton extraction are strongly linked problems. Several methods first extract a skeleton and then impose a partitioning of the object based on a partitioning of the skeleton. For instance, in [LTTH01] 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. In [RGS04] the object is approximated using bead-like primitives by first extracting a voxelized skeleton and then partitioning it. As mentioned above, in [LKA06] an iterative approach is used that simultaneously generates a shape decomposition and a corresponding set of skeletons. If the quality threshold of the extracted skeleton is not met, it is refined hierarchically to produce an ACD 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 partitioning of the image imposes a mesh segmentation in 3D.

7. Discussion

Table 2 summarizes the different automatic segmentation solutions in terms of segmentation type, technique, and the attributes used. It is clear that there is no real connection between any specific technique and the segmentation type. Similarly, in most cases the same attributes can be used by different techniques to define the segmentation. There is, however, a link between the attributes used and the goal of segmentation as discussed in Section 4. Planarity, normal and dihedral angles and curvatures are used when surface-type partitioning is sought, while higher level primitives and skeletons are used for part-type segmentations. Geodesic distances as well as topological attributes are used by both.

One of the key issues in the segmentation of meshes is the tradeoff between segmentation quality and the number of parts. On the one hand, most optimization criteria are better fulfilled by small parts, but on the other, one does not want the final segmentation of the object to include too many or too small parts. Non-parametric techniques such as region growing or hierarchical clustering (when it is stopped by a quality criteria) tend to over-segment and usually include merging as post processing. On the other hand, using parametric techniques such as iterative clustering or spectral analysis the user must determine the number of parts a priori, or a meta-algorithm must be used to search for the number of parts.

Almost all techniques use some post-processing to smooth the boundaries between the segments as these tend to depend more on the triangulation than on the actual segmentation or attribute used (Figure 6).

7.1. Pose invariance

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, such as a human or animal models, it can maintain various poses. In such cases it is important that the object's segmentations 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 higher dimensional distances (geodesic distances from each point to all other points) into a lower dimension Euclidean space, where Euclidean distances approximate well the higher dimensional distances. Often this maps different poses of the same object to similar poses. Later, feature points are extracted, namely points that reside on tips of prominent components of a given model. Each prominent component (or segment) 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]. As stated earlier, the SDF remains largely consistent through pose changes of the same object. Thus, it can guide pose-invariant segmentations. The segmentation uses iso-contours of this function on the mesh along with graph cut refinement. Another extension defined is the compatible segmentation of multiple but different meshes. Such segmentation enables correspondence between objects and object parts, which is important to motion transfer, shape matching or editing. The SDF function 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.

7.2. Interactive methods

Lastly, fully automatic segmentation still remains a hard problem especially because it concerns semantics of shape and form. Several manual or user guided segmentation and

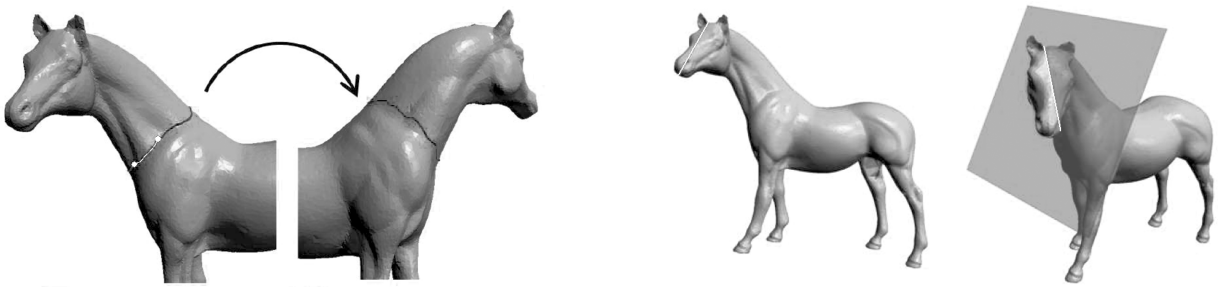

Figure 7: *Manual segmentation examples. Left: in semi-implicit methods the user designates some portion of the cut and the system completes it. Right: in an explicit method an explicit cutting tool such as a plane is used to partition the mesh.*

partitioning techniques have been proposed in literature [GSL*99, ZSH00, FKS*04, LLS*04, JLCW06]. Using interactive methods, segmentation is often created by using cuts that define the boundaries between the segments. Cuts can be either defined explicitly using user interface tools, semi-implicitly, by designating some vertices and calculating shortest path between them, or implicitly by defining two sides of a graph on the mesh and using the graph-cut algorithm (Figure 7). Recently, interactive methods have been concerned more with user interface design issues to enable natural gestures which imitate the physical notion of cutting [SBSCO06,JLCW06].

8. Concluding remarks

We have formulated boundary mesh segmentation as an optimization problem and presented the main approaches used in literature for segmentation. We have also listed the mesh attributes used by the different techniques to define the criteria for optimization. In general, the key factor for choosing both the algorithm and the criteria is the application in mind. We have identified a distinct difference between the 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 it is reflected in the segmentation goal. Surface attributes such as planarity, element angles and curvature are more appropriate for surface-type segmentation, while higher level primitive fitting or skeletons are suitable for object partitioning.

As stated in the preliminaries, all algorithms presented here are approximations that solve an optimization problem and reach some local minima. A comparison to the global optimum of the various schemes is interesting, although out of the scope of this review. Similarly, it would be interesting to research which of the different objective functions are easier and which are harder to approximate.

Although there are already numerous techniques for mesh segmentation, it seems that directions to address this problem are only beginning. Just as a measure one can compare the large number of image segmentation publications to the relatively small number of mesh segmentation publications. It seems that more advanced issues such as examining invariance under different types of transformations and defining compatible partitioning are still largely open problems and would require further research.

Acknowledgements

The author would like to thank the reviewers for valuable comments and suggestions, and Hugues 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] AIM AT SHAPE: 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/, February 2007.

[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), IEEE Computer Society Press, pp. 14–25.

[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.

[BD92] BAJAJ C. L., DEY T. K.: Convex decomposition of polyhedra and robustness. *SIAM Journal on Computing 21*, 2 (1992), 339–364.

[BMM*03] BIASOTTI S., MARINI S., MORTARA M., PATANE G., SPAGNUOLO M., FALCIDIENO B.: *Proceedings ofthe 11th International Conference On Discrete Geometry for Computer Imagery DGCI 2003*, Naples, Italy, November 19–21, 2003 Lecture Notes in Computer Science Springer, Berlin/Heidelberg Volume 2886/2003, pp. 194–203.

[Bie87] BIEDERMAN I.: Recognition-by-components: A theory of human image understanding. *Psychological 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.

[Cha81] CHAZELLE B. M.: Convex decompositions of polyhedra. In *STOC '81: Proceedings of the thirteenth annual ACM symposium on Theory of computing* ACM Press, New York, NY, USA, 1981, pp. 70–79.

[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.

[CP94] CHAZELLE B., PALIOS L.: Decomposition algorithms in geometry. In *Algebraic Geometry and its Applications*,BAJAJ C., (Ed.). Springer-Verlag, Berlin, 1994, ch. 27, pp. 419–447.

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

[CSM03] COHEN-STEINER D., MORVAN J.-M.: Restricted delaunay triangulations and normal cycle. In *SCG '03: Proceedings of the nineteenth annual symposium on Computational geometry* (New York, NY, USA, 2003), ACM Press, pp. 312–321.

[CSM07] CORNEA N., SILVER D., MIN P.: Curve skeleton, properties and algorithms. *IEEE Transactions on Visualization and Computer Graphics accepted for publication* (2007).

[Del99] DELINGETTE H.: General object reconstruction based on simplex meshes. *International Journal of Computer Vision 32*, 2 (September 1999), 111–146.

[DHS00] DUDA R. O., HART P. E., STORK D. G.: *Pattern Classification (2nd ed.)*. Wiley Interscience, ACM, New York, NY 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*, 3 (2004), 652–663.

[GFW*06] GUNTER J., FRIEDRICH H., WALD I., SEIDEL H.-P., SLUSALLEK P.: Ray tracing animated scenes using motion decomposition. *Computer Graphics Forum (Proc. EG 2006)* 3 (2006), 517–525.

[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* ACM Press, New York, NY, USA, 2004, 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.

[IR05] IP C. Y., REGLI W. C.: Manufacturing classification of cad models using curvature and svms. In *International Conference on Shape Modeling and Applications 2005 (SMI' 05)* (2005), pp. 363–367.

[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.

[JLCW06] JI Z., LIU L., CHEN Z., WANG G.: Easy mesh cutting. *Computer Graphics Forum (Proc. EG 2006)* 3 (2006), 283–292.

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

[KJS06] KRAEVOY V., JULIUS D., SHEFFER A.: *Shuffler: Modeling with Interchangeable Parts*. Tech. Rep. TR-2006-09, Department of Computer Science, University of British Columbia, April 2006.

[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, February 2007.

[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 (Proceedings of Pacific Graphics 2005) 21*, 8–10 (2005), 865–875.

[KS98] KIMMEL R., SETHIAN J.: Fast marching methods on triangulated domains. *Proceedings of the National Academy of Sciences 95* (1998), 8341–8435.

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

[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.

[LA06] LIEN J.-M., AMATO N. M.: *Approximate convex decomposition of polyhedra*. Tech. Rep. TR06-002, Parasol Lab, Dept. of Computer Science, Texas A&M University, January 2006.

[LDB05] Lavoué G., DUPONT F., BASKURT A.: New cad mesh segmentation method A, based on curvature tensor analysis. *Computer Aided Design 37*, 10 (2005), 975–987.

[LKA06] LIEN J.-M., KEYSER J., AMATO N. M.: Simultaneous shape decomposition and skeletonization. In *Proceedings of ACM Solid and Physical Modeling Symposium (SPM'06)* (2006), pp. 219–228.

[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.

[LLYL05] LEE T.-Y., LIN P.-H., YAN S.-U., LIN C.-H.: Mesh decomposition using motion information from animation

sequence. In *Proceedings of Computer Animation and Virtual Worlds* (2005), pp. 519–529.

[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 Transactions on Graphics 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.

[LZ07] LIU R., ZHANG R.: Mesh segmentation via spectral embedding and contour analysis. *Computer Graphics Forum, Eurographics Proceedings to appear* (2007).

[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.

[MPS06] MORTARA M., PATANÈ G., SPAGNUOLO M.: From geometric to semantic human body models. *Computers & Graphics 30*, 2 (2006), 185–196.

[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.

[Pet02] PETITJEAN S.: A survey of methods for recovering quadrics in triangle meshes. *ACM Computing Surveys 34*, 2 (2002), 211–262.

[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.

[PSG*06] PODOLAK J., SHILANE P., GOLOVINSKIY A., RUSINKIEWICZ S., FUNKHOUSER T.: A planar-reflective symmetry transform for 3D shapes. *ACM Transactions on Graphics (Proc. SIGGRAPH) 25*, 3 (July 2006).

[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. *Journal of Computing and Infomations Science in Engineering 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 of Pacific Graphics 2006) 22* (2006), 835–844.

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

[Sha04] SHAMIR A.: A formalization of boundary mesh segmentation. In *Proceedings of the second International Symposium on 3DPVT (3D Data Processing, Visualization, and Transmission)* (2004), pp. 82–89.

[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.

[SSK05] SATTLER M., SARLETTE R., KLEIN R.: Simpler and efficient compression of animation sequence. In *Proceedings of 2005 ACM SIGGRAPH/EG symposium on computer animation* (2005), pp. 209–217.

[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 of Pacific Graphics 2006) 22*, 9–11 (2006), 825–834.

[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.

[VMC97] VARADY T., MARTIN R., COX J.: Reverse engineering of geometric models–an introduction. *Computer-Aided Design 29*, 4 (1997), 255–268.

[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 modelling: 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* ACM Press, New York, NY, USA, 2004, 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.