

Feature Sensitive Mesh Segmentation

Yu-Kun Lai
Tsinghua University
Beijing, China *

Qian-Yi Zhou
Tsinghua University
Beijing, China †

Shi-Min Hu
Tsinghua University
Beijing, China ‡

Ralph R. Martin
Cardiff University
Cardiff, UK §

Abstract

Segmenting meshes into natural regions is useful for model understanding and many practical applications. In this paper, we present a novel, automatic algorithm for segmenting meshes into meaningful pieces. Our approach is a clustering-based top-down hierarchical segmentation algorithm. We extend recent work on feature sensitive isotropic remeshing to generate a mesh hierarchy especially suitable for segmentation of large models with regions at multiple scales. Using integral invariants for estimation of local characteristics, our method is robust and efficient. Moreover, statistical quantities can be incorporated, allowing our approach to segment regions with different geometric characteristics or textures.

1 Introduction

Triangular meshes are now widely used in computer graphics. They are easy to acquire and widely available. The demands for techniques of analysis, processing, storage, transmission and rendering of triangular meshes are ever increasing. However, due to their irregular connectivity and lack of high-level semantic structures, the automatic analysis of meshes is challenging. Cutting mesh models into meaningful pieces is one important step towards surface understanding, and has a wide range of applications.

Segmentation is a crucial step in reverse engineering of CAD models: it divides the mesh into regions, each of which is fitted using a single analytical surface [Várady et al. 1997]. In computer graphics, various applications use segmentation as a preprocessing step. Mesh simplification can be improved by constraining contraction to take place within segmented regions, leading to improved quality of simplified models [Zuckerberger et al. 2002]. Li et al. [Li et al. 2001] demonstrated the use of well chosen segmentation in improving the performance of collision detection. Segmentation is also useful in morphing [Shlafman et al. 2002; Zuckerberger et al. 2002] and skeleton-driven animation [Katz and Tal 2003].

Ideas from cognitive science give a useful basis for model segmentation. Hoffmann and Richards [Hoffmann and Richards 1984] proposed the so-called *minimal rule*: the human visual system perceives region boundaries along negative minima of principal curvature, or concave creases. Later, Hoffmann and Singh [Hoffmann and Singh 1997] pointed out that the depth of the concavity directly affects the salience of region boundaries. Thus, concave feature regions, as well as other features, are crucial for segmentation [Katz and Tal 2003; Liu and Zhang 2004]. In addition, we wish to take geometric texture information into account—texture segmentation is widely used in image processing and should also be useful here.

Recently, the *regularized isophotic metric* was proposed in [Pottmann et al. 2004]; this distance function depends both on position and normal information. Going further, using the idea of an image manifold from image processing [Kimmel et al. 2000], each point \mathbf{x} on the surface can be mapped to a corresponding point

$\mathbf{x}_f = (\mathbf{x}, w\mathbf{n})$ in \mathbb{R}^6 , where $\mathbf{n}(\mathbf{x})$ is the unit normal vector corresponding to \mathbf{x} and w is a user specified constant controlling feature sensitivity [Lai et al. 2006]. This maps the surface $\Phi \subset \mathbb{R}^3$ to the corresponding 2-manifold $\Phi_f \subset \mathbb{R}^6$. Then, the *feature sensitive distance* for a curve c may be defined to be the Euclidean length of the image curve c_f . A method for feature sensitive remeshing was proposed in [Lai et al. 2006] using isotropic remeshing of Φ_f . The characteristics of the meshes produced were studied in that paper, leading to the suggestion that they could be a useful tool for robust feature extraction.

Our segmentation approach here is a clustering-based segmentation algorithm, like other state-of-the-art mesh segmentation methods such as [Katz and Tal 2003]. It uses locally defined integral invariants [Manay et al. 2004] to estimate local properties of the surface, which is much more robust than simply computing dihedral angles or estimating discrete curvatures. Feature sensitive remeshing [Lai et al. 2006] is a useful tool for efficiently computing integral invariants for segmentation. Only normal information is used, avoiding direct estimation of higher-order differential quantities.

We use feature sensitive remeshing to produce a *hierarchy* of meshes, allowing us to efficiently construct a *hierarchical segmentation*. In this way, our method performs segmentation of models using the most appropriate level in the mesh hierarchy. It can thus handle larger models than previous k -means clustering based methods, and it can also segment coarse and fine details of complicated model using the same hierarchy. Moreover, by using statistical measures of integral invariants, we can achieve segmentation based on large-scale variation of local surface characteristics or variation in geometric texture. In this way, our method can separate regions without clear boundaries.

In Section 2, recent work on surface segmentation is discussed. We then outline our segmentation algorithm in Section 3. The two key steps, hierarchical feature sensitive remeshing, and hierarchical k -means clustering based segmentation, are detailed in Sections 4 and 5 respectively. Experimental results are presented in Section 6, while conclusions and discussions for future work are given in Section 7.

2 Related Work

Image segmentation is a key step in image analysis and understanding, and has received much attention. Its counterpart for 3D surfaces has been studied only much more recently.

Based on different aims, segmentation methods can generally be classified into two types: *patch-type* segmentation and *part-type* segmentation [Shamir 2004]. The former methods mostly aim to producing patches that satisfy certain geometric properties, often being similarity of geometric properties. For example, Sander et al. [Sander et al. 2003] segment a mesh model into a set of charts, each of which is almost planar; such segmentation is suitable for parameterization as done in multi-chart geometry images. Trying to produce planar patches is too restrictive, and later improvements by Juliu et al. [Julius et al. 2005] try to segment models into quasi-developable patches. Part-type segmentation on the other hand tries to segment models into meaningful pieces, usually based on significant features. Our method is of the latter type, and we will mainly

* laiyc@cg.cs.tsinghua.edu.cn

† zhouqy@cg.cs.tsinghua.edu.cn

‡ shimin@tsinghua.edu.cn

§ ralph@cs.cf.ac.uk

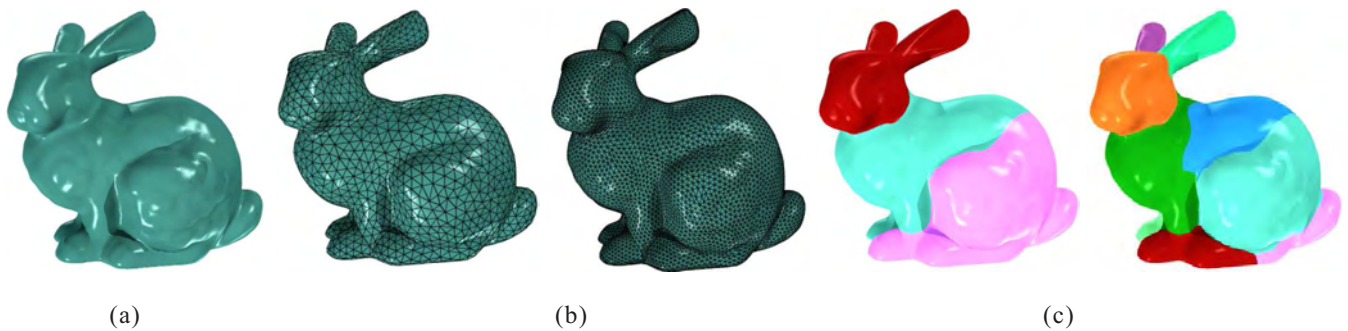


Figure 1: Steps of our segmentation algorithm

consider this type of segmentation.

Segmentation methods generally fall into two classes: *region based*, and *boundary based*. Segmentation methods rely on estimating local properties. Boundary based approaches use special values of these local properties as candidate locations for boundaries, and regions are deduced from the located boundaries. Region based methods, however, look for areas having similar properties, which define the regions, and the boundaries are deduced from them.

Various operators have been used for estimating properties. Some methods use discrete curvature estimators (e.g. [Mangan and Whitaker 1999; Page et al. 2003; Srinark and Kambhamettu 2003; Zhang et al. 2002]). Other work [Katz and Tal 2003; Liu and Zhang 2004; Shlafman et al. 2002] uses a combination of geodesic and angular distances for similarity measurement, angular distances being a function of dihedral angle between adjacent triangles. Gelfand and Guibas [Gelfand and Guibas 2004] proposed a rather different shape descriptor, using slippage analysis, which is more suitable for segmenting mechanical components than computer graphics models.

Some segmentation methods are boundary based. For example, Zhang et al. [Zhang et al. 2002] give an algorithm which explicitly locates boundaries using discrete curvatures. Generally, such approaches depend on accurate discovery of boundary loops.

However, many segmentation methods are region based. Mangan and Whitaker [Mangan and Whitaker 1999] extend the bobsleding watershed algorithm to triangular meshes. Page et al. [Page et al. 2003] use an alternative hill climbing algorithm for watershed segmentation. They compute a directional height map and use impeded climbing up negative principal curvature hills. Srinark et al. [Srinark and Kambhamettu 2003] classify local surface regions using curvature estimates, and then use region growing to segment the model, starting from certain seeds. Though fast, this approach does not seem to handle noise well, and can also result in over-segmentation. Gelfand and Guibas [Gelfand and Guibas 2004] use local slippage analysis and a multi-pass region growing approach based on slippage signatures to separate different regions. This approach also requires good quality models.

Other region-based approaches use iterative *clustering* as a tool (as we also do here). Use of global optimization allows such approaches to be more robust to variations in local properties caused by noise, for example. Shlafman et al. [Shlafman et al. 2002] use k -means clustering to provide a meaningful segmentation. However, the regions produced have jagged boundaries. This work was later improved in [Katz and Tal 2003], using hierarchical segmentation, fuzzy clustering and minimal boundary cuts to produce smoother boundaries. (In contrast, we use feature sensitive smoothing which, like the use of geometric snakes, tends to produce smoothed boundaries snapped to features). Spectral clustering was suggested by Liu et al. [Liu and Zhang 2004]; the authors claim it gives superior results for clean mesh models. However, these methods de-

pend on dihedral angles whose computation is sensitive to noise. Geodesic and angular distances between all pairs of triangles must be precomputed and stored for efficiency; this limits the size of input mesh for which this is practical, even if done in a hierarchical manner. Another drawback is that the results depend on mesh connectivity. Cohen-Steiner et al. [Cohen-Steiner et al. 2004] use a similar Lloyd’s-type clustering algorithm to that used in this work; however, their goal is surface approximation.

Unsupervised clustering techniques like the mean shift method can also be applied to mesh segmentation. Shamir et al. [Shamir et al. 2004] extend mean shift analysis to mesh models using local parameterization. Yamauchi et al. [Yamauchi et al. 2005] apply mean shift clustering to surface normals, and then use a method similar to that in [Sander et al. 2003] to compute the segmentation, based on the clustered normals. The number of clusters is computed during the segmentation process; however, the method presented in [Yamauchi et al. 2005] is likely to segment a model into more pieces than the desired number of meaningful parts.

Other approaches to segmentation also exist. Li et al. [Li et al. 2001] use edge contraction based skeletonization and space sweeping for mesh decomposition. Their method provides visually appealing results; however, it tends to capture large-scale shapes rather than features, making some decompositions impossible. Mitani and Suzuki [Mitani and Suzuki 2004] proposed a technique for making paper models from meshes. This can be considered as involving a special segmentation scheme that guarantees each region is developable. Wu and Levine [Wu and Levin 1997] proposed a method that simulates the distribution of electrical charges on the surface. Boundaries of regions are locations with minimal charge. The aim is to capture sharp concave features which are usually perceived as natural boundaries. Unlike most other methods, this approach does not depend on differential quantity estimation and is thus more stable. However, assumptions are needed about the nature of the input object, which should generally have an even distribution of concave feature regions for boundaries, and the mesh should also be closed. Katz et al. [2005] propose a segmentation algorithm based on feature points and core extraction. Pose-invariant results are reported in this paper. However, an expensive optimization method is used to find feature points, which limits the complexity of models that can be efficiently handled after simplification.

Other work has considered interactive mesh segmentation. As well as extracting features automatically, the approach proposed in [Lee et al. 2004] also gives the user tools to close and optimize boundary loops separating different parts of the object. Funkhouser et al. [2004] present a modeling system based on searching for and stitching parts from a database. An intuitive interactive segmentation tool is given to find optimal cuts guided by user-drawn strokes, formulated as a constrained least-cost path problem.

Our approach is automatic and region-based. As in the method in [Katz and Tal 2003], we can produce a hierarchy of segmenta-

tion, of particular use in certain applications. We use hierarchical feature sensitive isotropic remeshing to efficiently and robustly segment large models at several levels. Integral invariants and statistical quantities are used as local properties in a k -means clustering approach. These provide more robustness than dihedral angles, are insensitive to small fluctuations in surfaces, and are also capable of segmentation using certain types of geometric textures.

3 Algorithm Overview

Given an input model represented as a triangular mesh, and a few user specified parameters, our method produces a set of disjoint, constituent regions, whose union is identical to the input mesh.

For some input models, preprocessing may be desirable. Though our algorithm can be applied to manifold models with or without holes, we do assume that any necessary topological correction has been carried out to remove unwanted gaps, tiny handles or other deficiencies (*topological noise*). For very large models (more than 10^6 triangles, using a current desktop PC), it may be desirable to apply mesh simplification [Garland and Heckbert 1997; Hoppe 1996] in order to obtain results in a reasonable time (a few minutes).

We now map the mesh from \mathbb{R}^3 to its counterpart in \mathbb{R}^6 , as described earlier. This is done vertex by vertex while keeping the connectivity unaltered. As part of this process, the normal at each vertex needs to be estimated. For models that are not too noisy, normals can be reliably estimated using 1-ring neighbors. For noisier models, however, improved results can be achieved by estimating normals using local planar or quadratic surface fitting to a neighborhood—see [Lai et al. 2006] for further details.

Next, the new mesh is subjected to hierarchical feature sensitive remeshing, as explained in detail in Section 4. This process generates a hierarchy of feature sensitive (FS) meshes, of increasing resolution with successive levels of detail, and with clear correspondences between adjacent levels. This is accomplished by first constructing the coarsest level of remeshing; for each finer level, every triangle in the coarser level is subdivided into 4 triangles and newly inserted vertices are repositioned in nearby locations to optimize isotropic sampling in \mathbb{R}^6 . The hierarchical output is conceptually similar to an Gaussian image pyramid. In the first step of segmentation, the coarsest level of remeshing is used, reducing the computational complexity and ensuring stability with respect to small scale fluctuations. As hierarchical segmentation proceeds, each area to be segmented at lower levels contains fewer triangles. When an area contains too few triangles on the current mesh so that the corresponding finer mesh contains fewer than a certain pre-specified number of triangles, we move to a finer mesh.

A k -means clustering algorithm is used to segment a given area (at the top level, the remeshed surface, or part of it at lower levels). This clusters triangles to form regions as detailed in Section 5. The number of clusters, k , can be specified by the user, or can be derived by optimization as in [Katz and Tal 2003].

A *metric* measuring distances between triangles is needed to perform clustering. We use a definition incorporating geodesic distance, integral invariants related to averaged normal curvature, and statistical measures of these invariants characterizing local properties such as geometric texture. Given a user specified number of regions to be generated at the current level, several initial regions are selected, which are then improved iteratively. This segmentation process is performed at several levels (if desired), giving a hierarchical segmentation.

The segmentation results on the FS mesh can be mapped to the original input mesh by projection, in a similar way to the approach used in [Katz and Tal 2003]. Alternatively, the segmented FS mesh itself, which has better properties, may be used in downstream applications.

Input: *a mesh model; certain user-specified parameters.*

Output: *a set of disjoint, contiguous regions representing meaningful parts of the input model.*

1. Preprocessing (optional);
2. Compute hierarchical feature sensitive remeshing.
3. Perform hierarchical segmentation with k -means clustering and a new distance metric.
4. Map the result back to the original model (optional).
5. Perform feature sensitive boundary smoothing.

Figure 2: Algorithm Overview

If the initial region boundaries are too jagged, they can be improved by *feature sensitive smoothing* as in [Lai et al. 2006] or by use of *geometric snakes* [Lee and Lee 2002].

Our algorithm is summarised in Figure 2.

4 Hierarchical Feature Sensitive Remeshing

Clustering-based segmentation algorithms need to compute distances between pairs of triangles on the mesh. In practice, distances between *most* pairs of triangles will be required. These pairwise distances need to be randomly accessed during k -means clustering and should be kept in main memory. They may be found using Dijkstra’s shortest-path algorithm in $O(N^2 \log N)$ time, where N is the number of triangles; $O(N^2)$ storage is required. This is expensive for large N , and previous methods (e.g. [Katz and Tal 2003]) have used a simplified mesh as a means to provide a segmentation for large models. Simplification ideally would be done in such a way as to ensure consistency of the segmentation with the original model.

Hierarchical segmentation was first introduced by Katz et al. [Katz and Tal 2003]. It is able to represent a decomposition of a model at different levels, mimicking the way people think. To achieve hierarchical segmentation, they simply segment each region at each level of detail into further regions, recursively. However, at the higher levels, the simplified triangulation count is low for each region, and inadequate detail may be present due to mesh simplification. This has an impact on the correctness and accuracy of the segmentation of the original mesh.

Our alternative approach is suited to large meshes: we use multi-resolution, hierarchical, feature sensitive remeshing to reduce the size of the computational problem, while avoiding the loss of significant detail in the reduced size meshes. In particular, the input model is remeshed into a hierarchy of models with different resolutions with clear correspondences between adjacent levels in the hierarchy. The coarsest remeshing is used for the initial segmentation. In earlier stages of hierarchical segmentation, *details* of models are usually of little use, and the global shape at a coarse resolution is important. Later, areas of the model are segmented using more detailed meshes. At finer levels of detail, only single already-segmented areas of the mesh need to be processed at a given time, not the whole mesh. By using hierarchical remeshing in conjunction with hierarchical segmentation, our method is capable of handling larger models, and segmenting them into more levels.

Thus, instead of using mesh simplification, we use *feature sensitive, isotropic, remeshing* [Lai et al. 2006]. Typically, we might make from one to three levels of such FS meshes, each with 1/4 the number of triangles of the previous level. An FS mesh in gen-

eral has almost equilateral, equally sized triangles in \mathbb{R}^6 . However, it also has the desirable property that triangles are elongated along sharp features. This makes it possible to efficiently represent models. Moreover, a topological disk on an FS mesh is a good approximation to a geodesic disk in \mathbb{R}^6 . Let α represent the mapping between Φ and Φ_f . It has been shown in [Lai et al. 2006] that the affine first derivative mapping $D\alpha^{-1}$ maps the unit circle \mathbf{k}_f in \mathbb{R}^6 , centered at some point \mathbf{x}_f on Φ_f , to an ellipse \mathbf{k} in \mathbb{R}^3 , in the corresponding tangent plane of Φ at the point \mathbf{x} . This mapping distorts the local shape. The *principal distortions*, corresponding to principal curvatures, are the extremal distortions. Thus, the distances of the vertices of the ellipse to its center are the corresponding principal distortions $1/\lambda_i$, $i = 1, 2$, which satisfy

$$\lambda_i^2 = 1 - w^2 K + 2w^2 H \kappa_i = 1 + w^2 \kappa_i^2, \quad (1)$$

where κ_i are the two principal curvatures, respectively [Lai et al. 2006]. Thus the principal distortions, which can be estimated as an integral quantity, are closely related to the local surface curvatures. We explain how we use them for segmentation in Section 5.

FS remeshing can be computed by extending an isotropic remeshing algorithm (e.g. [Alliez et al. 2003; Surazhsky et al. 2003; Witkin and Heckbert 1994]) which works in \mathbb{R}^3 to the feature sensitive metric in \mathbb{R}^6 . We use the method in [Lai et al. 2006]. It uses the iterative method in [Witkin and Heckbert 1994] to optimize the sampling, and to further improve the results, geodesic distances computed by the method in [Surazhsky et al. 2005], rather than Euclidean distances, are used in an energy function which is a sum of spring energies designed to cause vertices to repel one another.

As geodesic distances are used, an FS mesh contains almost equilateral triangles with almost identical size in terms of the feature sensitive metric on the input model. Given such a mesh, if each triangle is split into four smaller ones, by inserting a vertex at some appropriate point near the mid-point of each edge, the resulting refined model is still nearly isotropic in this sense. The refined model can be computed as follows. Insert a vertex at the mid-point of each edge of the coarse level mesh, and project these newly inserted vertices onto the input model in \mathbb{R}^3 . (The projection can be done in \mathbb{R}^3 or \mathbb{R}^6 . However, it appears to be more robust to do the projection in \mathbb{R}^3). The connectivity and positions of the vertices from the coarser level mesh are kept unaltered. This ensures that vertices do not move globally, and the correspondence between the coarse and finer levels is simply given by the above one-to-four mapping. The new vertices are repositioned using spring energy optimization. The neighborhood used for computing spring energy functions can be easily found from the topological neighbors based on subdivision. The optimization carried out is similar to the one used for remeshing at the coarsest level. Armijo rule [Kelley 1999] step-size control can be incorporated to make the result more stable. The initial positions are usually quite close to the required solution, and it takes just a few iterations to achieve acceptable remeshing results.

As noted earlier, it is usually sufficient to remesh models at from one to three levels, depending on detail in the input model and degree of segmentation required. Fig. 3 shows the Armadillo model (originally with 345,944 triangles) remeshed with 11,756 (left) and 47,024 triangles (right).

5 Hierarchical Segmentation

Use of remeshed models makes segmentation tractable, and estimation of geometric properties efficient. The triangles in the model as clustered into k meaningful regions using k -means clustering, which assigns triangles to clusters according to distances from an iteratively updated *representative* triangle for each cluster. Section 5.1 gives our basic approach to clustering-based hierarchical

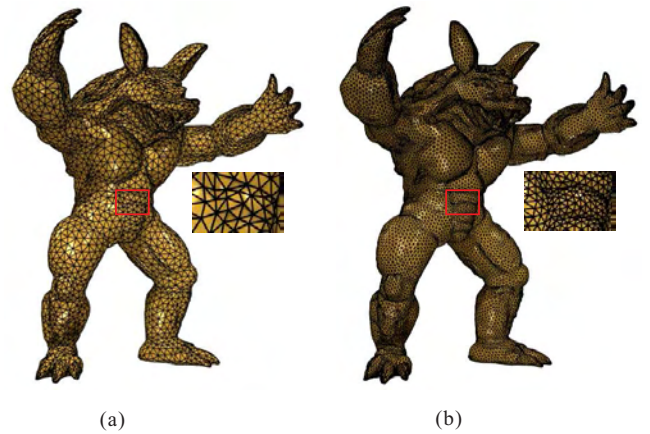


Figure 3: FS meshes of the Armadillo model at coarser and finer resolutions

segmentation, Section 5.2 discusses the key issue of distance computation, and Section 5.3 briefly considers how to ensure that each region has a smooth boundary.

5.1 Hierarchical Segmentation Approach

Our hierarchical segmentation approach is similar to those in [Katz and Tal 2003] and [Shlafman et al. 2002]. The main differences lie in the distance computation (see Section 5.2) and the use of multiresolution remeshing. The number of clusters, k , can be specified by the user, or can be derived automatically by optimization as in [Katz and Tal 2003].

The algorithm proceeds from coarse to fine segmentation of the input mesh. Segmentation is performed on the appropriate FS mesh and mapped back to the original mesh if desired. Initially, the entire lowest resolution FS level mesh is segmented into regions. Subsequently, if further segmentation is required, the segmentation process is applied to the individual regions identified at the previous level of segmentation. A finer FS mesh is used if the region has a manageable size in this mesh, i.e. contains fewer than a certain pre-specified maximum number (say 10,000) of triangles. Otherwise, the coarser FS mesh is still used.

In order to segment a target (the whole object or a region) into smaller regions, k -means clustering is used as follows:

1. **Precompute distances between triangles.** The distance between each pair of triangles is computed using a metric which combines geodesic, curvature-related and geometric texture-based information.
2. **Pick seeds.** Seed triangles may be chosen by the user. Otherwise a seed triangle is randomly selected for the first cluster, and seeds are found for the other clusters one by one, by choosing the triangle that has the largest average distance to all other seeds found so far.
3. **Assign triangles to the nearest cluster.** Each triangle is assigned to the cluster to whose representative is closest.
4. **Update the representative of each cluster.** The representative r is updated to be the one minimizing

$$\sum_{f \in \text{Region}(r)} D(f, r);$$

here $D(f, r)$ is the distance between triangles f and r .

Steps 3 to 4 are iterated; in practice, just a few iterations suffice to converge to adequate results.

Note that the precomputed pairwise distances can be used again for further levels of segmentation if the same resolution of FS mesh is used, leading to efficiency.

5.2 Distance Computation

Distance computation is the key step in any k -means clustering algorithm. It affects the clustering outcome and a suitable metric must be carefully chosen dependent on the problem. For clustering-based segmentation, geodesic distance and angular distance have been used [Katz and Tal 2003; Shlafman et al. 2002; Liu and Zhang 2004].

Geodesic distance favors segmentation of equal sized regions, whereas angular distance tries to force region boundaries to lie where surface direction changes quickly, e.g. at sharp edges. This works well for clean models, producing regions typically separated by (ideally deep) valleys. However, for real, noisy, scanned models, angular distances are less useful. Noisy meshes can contain triangles with relatively large dihedral angles between them. In the extreme, a small spike on a relatively smooth surface may be segmented as a region if its triangles are far from all its neighbors in angular distance. Moreover, the angular distance approach usually applies a nonlinear mapping (e.g. the cosine function) to the dihedral angles, in order to reduce this distance in flatter regions. The overall effect for a specific portion of surface depends on whether it is represented by a few large triangles, or a larger number of smaller triangles.

Like previous authors, we first define the distance between an *adjacent* pair of triangles. The distance between *any* pair of triangles on the mesh can then be computed by following the shortest path on the dual graph of the mesh, using Dijkstra’s algorithm.

We define the distance function differently to previous authors, in terms of integral and statistical information about local features. In addition to using geodesic distance D_{geod} like previous work, we add two further terms, namely the *curvature distance* D_{curv} , derived from the mapping distortion of \mathbb{R}^6 geodesic disks, and the *texture distance* D_{texture} that measures changes in geometric texture or other statistical surface properties. The distance between an adjacent pair of triangles f_i and f_j is overall defined as

$$\begin{aligned} D(f_i, f_j) &= c_1 \cdot D_{\text{geod}}(f_i, f_j) / \bar{D}_{\text{geod}} \\ &+ c_2 \cdot D_{\text{curv}}(f_i, f_j) / \bar{D}_{\text{curv}} \\ &+ (1 - c_1 - c_2) \cdot D_{\text{texture}}(f_i, f_j) / \bar{D}_{\text{texture}} \end{aligned} \quad (2)$$

where the averages \bar{D}_* are over all pairs of adjacent triangles. We typically use $0.1 \leq c_1 \leq 0.2$ and $0.7 \leq c_2 \leq 0.9$.

The geodesic distance $D_{\text{geod}}(f_i, f_j)$ between two adjacent triangles is defined as the sum of the distances from the barycenters of two triangles to the center of the edge that is shared by the two triangles.

To efficiently compute the other two distance terms, we compute the mapping distortion at each vertex of the FS mesh. For a vertex v , we use an r -ring of neighbours (typically $r = 1$ to 3) as an approximation to a \mathbb{R}^6 geodesic disk. The freedom to choose r differently provides flexibility to selectively ignore small scale features, allowing a tradeoff between accuracy and robustness. The principal distortions λ_{min} and λ_{max} , and corresponding directions \mathbf{d}_{min} and \mathbf{d}_{max} , can be approximated using the geodesic disk. We find the shortest and longest distance from v to any point on the boundary of this neighborhood using a fast geodesic distance computation [Surazhsky et al. 2005]. Let these distances be Δ_{min} and Δ_{max} respectively, with corresponding vectors (projected on the tangent plane of v) \mathbf{d}_{min} and \mathbf{d}_{max} . We can estimate the mapping distortion as $\lambda_{\text{min}} = \Delta_{\text{min}}/l$ and $\lambda_{\text{max}} = \Delta_{\text{max}}/l$, where l is the radius of the

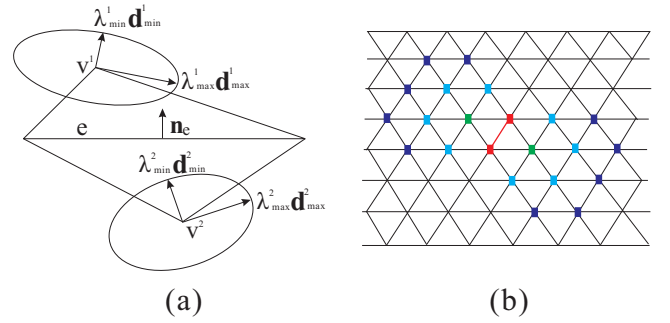


Figure 4: Distortion estimation and sampling pattern for geometric statistics

geodesic disk in the feature sensitive metric. We use \mathbf{d}_{min} and its orthogonal direction in the tangent plane of v as approximate principal directions

We now compute D_{curv} in Equation 2. For an adjacent triangle pair f_i and f_j , we consider the mapping distortion λ_e in the direction orthogonal to their common edge e , as follows. Taking the two vertices v^1, v^2 opposite to e in these triangles, we use their principal distortions to estimate the mapping distortion λ_e in the direction \mathbf{n}_e orthogonal to e :

$$\lambda_e = \frac{1}{2} \sum_{i=1,2} (\lambda_{\text{min}}^i \mathbf{d}_{\text{min}}^i + \lambda_{\text{max}}^i \mathbf{d}_{\text{max}}^i) \cdot \mathbf{n}_e. \quad (3)$$

(see Fig. 4(a)). D_{curv} can then be defined as

$$D_{\text{curv}}(f_i, f_j) = \eta G\left(\frac{1}{\lambda_e} - 1\right). \quad (4)$$

where η is a coefficient controlling the relative importance of convex and concave regions. Cognitive theory emphasises the importance of segmentation at concave regions, so η should be a small number (e.g. $0.1 \leq \eta \leq 0.2$) for convex regions and 1.0 for concave ones. G is a sigmoidal nonlinear function used to reduce the effect of low responses. Its use is very similar to application of the cosine function to dihedral angles when computing angular distances. We use

$$G(x) = 1 - \cos\left(\frac{\pi}{c} \min(x, c)\right) \quad (5)$$

where c is a threshold; $c = 0.5$ works well in practice. All the examples presented in this paper use this setting for c .

Most previous methods for mesh segmentation do not take into account differences in geometric textures when performing segmentation. While computation of local similarity is possible, it tends to be expensive, and sensitive to noise. We therefore compute statistical properties of integral invariants (e.g. λ_{min}) for use as descriptors of local surface properties. Another suitable integral quantity, the radius ratio ρ , is given by

$$\rho = \sqrt{\text{Area in } \mathbb{R}^3 / \text{Area in } \mathbb{R}^6}. \quad (6)$$

Small ρ corresponds to at least one of the principal curvatures being large. It can also be seen from Equation 1 that λ_{min} has a close relationship to the local average curvature. To use these descriptors for local shapes, given an edge e between two adjacent triangles f_i and f_j , we sample a specific neighborhood on either side of the edge (see Fig. 4(b)). The average and standard deviation of λ_{min} and ρ are computed and placed in a vector V (the standard deviation is multiplied by a weight to control relative importance of averages and standard deviations), and D_{texture} is thus defined as:

$$D_{\text{texture}} = \|V_i - V_j\|^2. \quad (7)$$

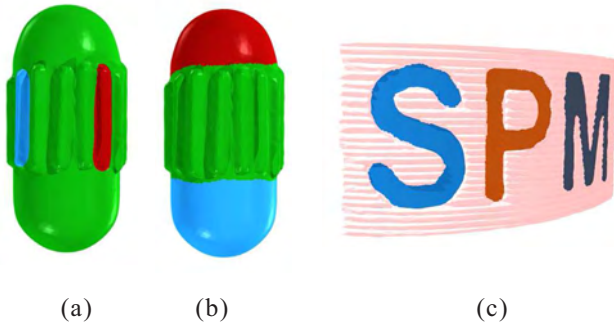


Figure 5: Segmentation by texture

By balancing the weights of constituent parts of the distance function, our method can produce varying results, as desired by the user. Fig. 5(b) shows an example where D_{texture} has been emphasised to segment a textured object into three regions; in this case we used $c_1 = c_2 = 0.1$. The result in Fig. 5(a) was produced with a weight for D_{texture} of 0.

Fig. 5(c) shows another example that segments the flat letters *SPM* from a surface covered with a grooved geometric texture. Because a large neighborhood has been used for geometric texture statistics, the locations of the boundaries are not very accurate and there also exist some rounding effects at corners due to the use of smoothing. A separate boundary optimization process is likely to be necessary in practice to further improve the results.

5.3 Patch Boundary Smoothing

Though our results tend to produce more robust boundaries near sharp features than previous methods, by utilising integral quantities rather than dihedral angles or discrete differential quantities, in some cases, the segmentation results are still jagged. We use *feature sensitive smoothing* [Lai et al. 2006] to improve this. The basic idea is to optimize discretized spline-in-tension energy in the feature sensitive metric. Segmentation boundaries tend to pass through feature regions, and such smoothing has the ability to snap the boundary to the features. It bears some similarity to geometric snakes [Lee and Lee 2002], but is much simpler to implement and avoids local parameterization which includes unavoidable mapping distortions.

Region boundary smoothing needs to be done carefully. As there are *branching* vertices on boundaries where three or more regions meet, we cannot move each boundary loop independently. Such vertices are detected, and the boundary is split into segments where any such vertices exist. Each segment is smoothed separately while keeping the locations of the branching vertices fixed (the positions of the latter could also be optimized too for further improvement).

The remeshing results can be mapped back to the original input model if desired using a projection method similar to the one in [Katz and Tal 2003].

Boundary smoothing should be done after projection.

6 Experimental Results

The weight w used for mapping into \mathbb{R}^6 should be chosen according to the scale of the input mesh [Lai et al. 2006] In the experiments reported later, we scaled the models to fit into a bounding box of size 1. The choice of w is not critical, and we typically used $0.05 \leq w \leq 0.1$.

Fig. 1 illustrates the main steps of our algorithm using the ‘bunny’ model with 25,000 triangles, remeshed using two levels of FS mesh with 17,336 and 4,334 triangles, respectively. Two levels of segmentation are shown, using the coarser and finer FS mesh respectively.

Fig. 6 shows segmentation of various objects with our method. The results are better for models with sharp features, especially concave features separating different parts (e.g. Figs. 6(a-d)). The FS meshes contain elongated triangles which tend to follow features, so that the computed region boundaries do not need smoothing. For models with fewer features, or whose features do not form closed loops, slightly more jagged boundaries result, and the segmentation is not as robust. For example, in Fig. 6(f), the front leg is cut higher than others, which in some sense is also reasonable, because the cut boundary takes into account certain creases on the model. Our method is robust in the presence of small fluctuations, as illustrated by Fig. 6(j). By varying the neighborhood size when computing integral invariants, the scale of features being considered can be altered. This is a useful property when small scale, sharper features coincide with large scale, smoother features. The various scanned models segmented in this Figure show the insensitivity of our method to noise.

Fig. 5 shows an example of segmentation by texture. The size of the statistical neighborhood used affects the texture captured. Our method can efficiently separate certain kinds of geometric textures, but due to the use of *simple* statistical measures, limitations exist; we wish to extend it to handle more complex kinds of geometric texture.

Fig. 7 shows two examples of segmenting models into a larger number of levels. The original ‘eagle’ model contains 33,072 triangles, and we used a simplified version of the ‘Lucy’ model containing 237,278 triangles. Segmentation was done to three levels. Using hierarchical FS meshes, detail can be retained while keeping the computation time and main memory manageable. Using the finest FS mesh, even the hand of ‘Lucy’ and the foot of the ‘eagle’ can be reliably segmented.

We tested our method on a Pentium IV 2.4GHz computer with 1GB RAM. Remeshing a model with 200K triangles to about 10K triangles takes under a minute. Segmentation time is directly related to the size of the input FS mesh. For models with 4K triangles, local property estimation takes 0.1s, pairwise distance computation 15.4s, and clustering 1.3s. For models with 11K triangles, the times are 0.2s, 134.8s, and 7.9s respectively. The time is dominated by computing pairwise distances. Note that smaller models require significantly less time to compute. By using *hierarchical* FS meshes, the time for pairwise distance computation is greatly reduced.

Compared to previous work, our approach is capable of handling complicated models with high efficiency, due to the use of feature sensitive hierarchical remeshing. Generally, our method is relatively insensitive to choices of the parameters in the algorithm. However, by using an appropriate neighborhood size in the computation of integral invariants, we believe more robust results might be achieved. Note that the ability to separate certain kinds of geometric *texture* means our method is particularly useful for certain types of application.

7 Conclusions

In this paper, a top-down hierarchical mesh surface segmentation algorithm was presented. As it is based on isotropic remeshing, it is insensitive to the input triangulation of the mesh. By using integral and statistical properties, problems due to noise are avoided, and textures can also be captured and used to drive the segmentation. Hierarchical FS remeshing not only provides an efficient tool for computation, but can handle larger models with more accuracy

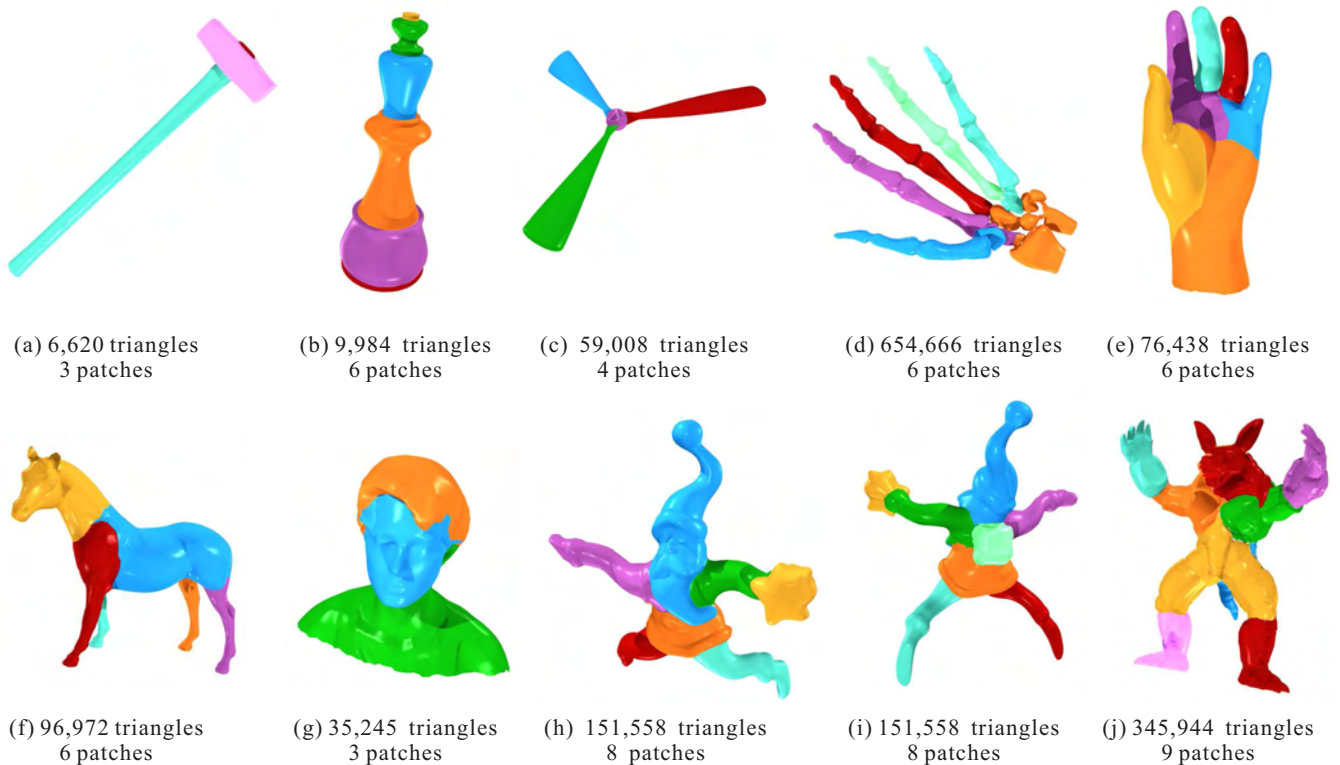


Figure 6: Various segmentation results.

than earlier methods. Use of finer models leads to better following of features, and better region boundaries. Improved results are obtained when more levels of segmentation are used.

We hope to address remaining limitations in the future. The algorithm is dependent on the initial seed points used, and better approaches for their placement needs to be explored. Choosing the number of regions automatically and reliably also needs further work. Our statistical approach needs extension to handle more complicated textures.

Acknowledgements

Models in this paper are courtesy of Cyberware, Stanford University, Georgia Institute of Technology and the Polhemus Corporation. This work was partially supported by the Natural Science Foundation of China (Projects 60225016, 60333010, 60321002) and the National Basic Research Project of China (Project 2002CB312101).

References

- ALLIEZ, P., DE VERDIÈRE, É. C., DEVILLERS, O., AND ISENBURG, M. 2003. Isotropic surface remeshing. In *Proceedings of Shape Modeling International*, 49–58.
- COHEN-STEINER, D., ALLIEZ, P., AND DESBRUN, M. 2004. Variational shape approximation. In *Proceedings of SIGGRAPH*, 905–914.
- FUNKHOUSER, T., KAZHDAN, M., SHILANE, P., MIN, P., KIEFER, W., TAL, A., RUSINKIEWICZ, S., AND DOBKIN, D. 2004. Modeling by example. In *Proceedings of SIGGRAPH*, 652–663.
- GARLAND, M., AND HECKBERT, P. 1997. Surface simplification using quadric error metrics. In *Proceedings of SIGGRAPH*, 209–216.
- GELFAND, N., AND GUIBAS, L. J. 2004. Shape segmentation using local slippage analysis. In *Proceedings of Eurographics Symposium on Geometry Processing*, 219–228.
- HOFFMANN, D. D., AND RICHARDS, W. A. 1984. Parts of recognition. *Cognition* 18.
- HOFFMANN, D. D., AND SINGH, M. 1997. Saliency of visual parts. *Cognition* 63, 29–78.
- HOPPE, H. 1996. Progressive meshes. In *Proceedings of SIGGRAPH*, 27–36.
- JULIUS, D., KRAEVOY, V., AND SHEFFER, A. 2005. D-charts: Quasi-developable mesh segmentation. *Computer Graphics Forum* 24, 3, 581–590.
- KATZ, S., AND TAL, A. 2003. Hierarchical mesh decomposition using fuzzy clustering and cuts. In *Proceedings of SIGGRAPH*, ACM Press, vol. 22(3), 954–961.
- KATZ, S., LEIFMAN, G., AND TAL, A. 2005. Mesh segmentation using feature point and core extraction. *The Visual Computer* 21, 8–10, 865–875.
- KELLEY, C. T. 1999. *Iterative Methods for Optimization*. SIAM.
- KIMMEL, R., MALLADI, R., AND SOCHEN, N. 2000. Images as embedded maps and minimal surfaces: movies, color, texture and volumetric images. *Intl. J. Computer Vision* 39, 111–129.

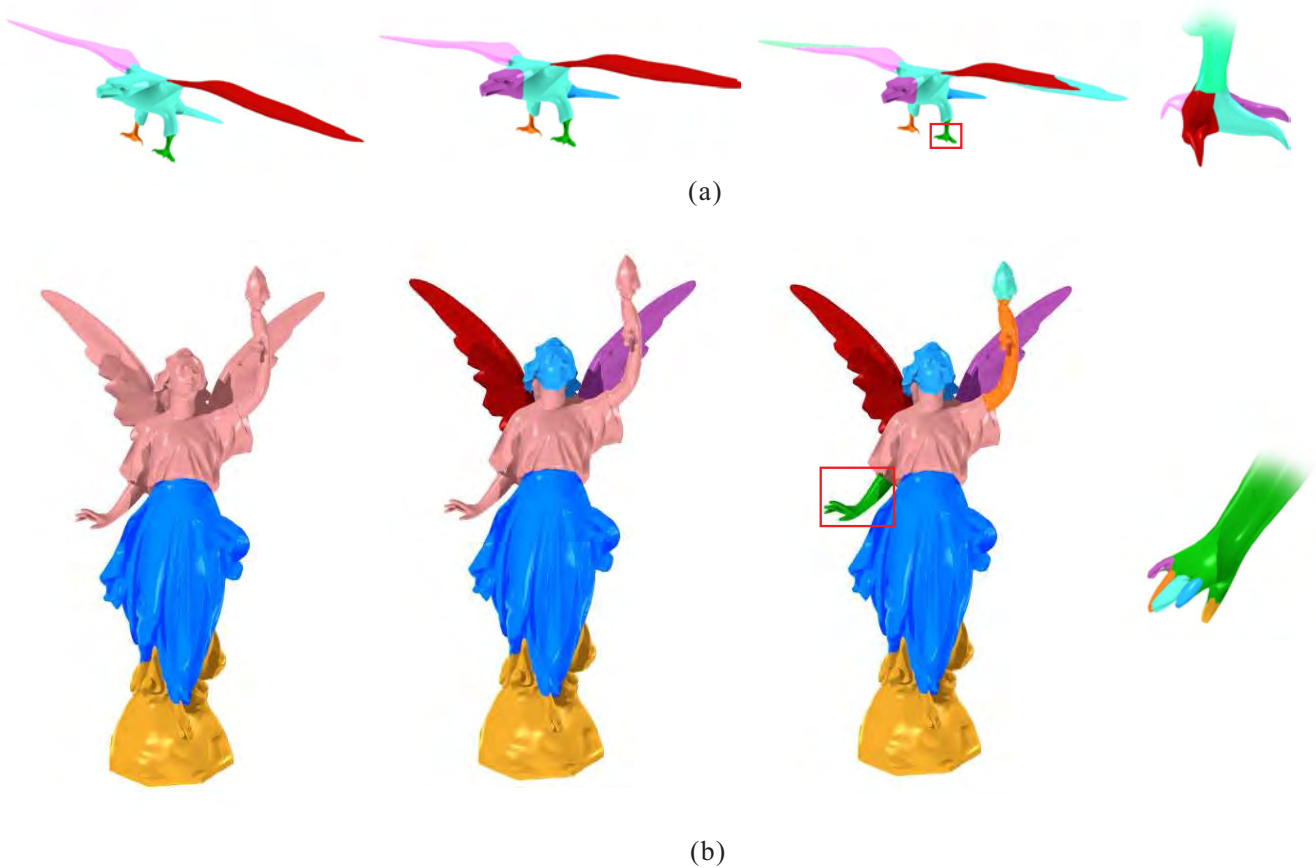


Figure 7: Hierarchical segmentation of larger models.

- LAI, Y.-K., ZHOU, Q.-Y., HU, S.-M., WALLNER, J., AND POTTMANN, H. 2006. Robust feature classification and editing. *IEEE Transactions on Visualization and Computer Graphics*. to appear.
- LEE, Y., AND LEE, S. 2002. Geometric snakes for triangular meshes. *Computer Graphics Forum* 21, 3, 229–238.
- LEE, Y., LEE, S., SHAMIR, A., COHEN-OR, D., AND SEIDEL, H.-P. 2004. Intelligent mesh scissoring using 3d snakes. In *Proceedings of Pacific Graphics*, 279–287.
- LI, X., WOON, T. W., TAN, T. S., AND HUANG, Z. 2001. Decomposing polygon meshes for interactive applications. In *Proceedings of ACM Symp. on Interactive 3D Graphics*, 35–42.
- LIU, R., AND ZHANG, H. 2004. Segmentation of 3D meshes through spectral clustering. In *Proceedings of 12th Pacific Graphics*, 298–305.
- MANAY, S., HONG, B.-W., YEZZI, A. J., AND SOATTO, S. 2004. Integral invariant signatures. In *Proceedings of ECCV*, Springer, 87–99.
- MANGAN, A. P., AND WHITAKER, R. T. 1999. Partitioning 3D surface meshes using watershed segmentation. *IEEE Trans. on Visualization and Computer Graphics* 5, 4, 308–321.
- MITANI, J., AND SUZUKI, H. 2004. Making papercraft toys from meshes using strip-based approximate unfolding. In *Proceedings of SIGGRAPH*, 259–263.
- PAGE, D. L., KOSCHAN, A. F., AND ABIDI, M. A. 2003. Perception-based 3D triangle mesh segmentation using fast marching watershed. In *Proceedings of the IEEE Conf. on Computer Vision and Pattern Recognition*, 27–32.
- POTTMANN, H., STEINER, T., HOFER, M., HAIDER, C., AND HANBURY, A. 2004. The isophotic metric and its application to feature sensitive morphology on surfaces. In *Proceedings of ECCV 2004, Part IV*, Springer, 560–572.
- SANDER, P. V., WOOD, Z. J., GORTLER, S. J., SNYDER, J., AND HOPPE, H. 2003. Multi-chart geometry images. In *Proceedings of Eurographics Symposium on Geometry Processing*, 146–155.
- SHAMIR, A., SHAPIRA, L., COHEN-OR, D., AND GOLDENTHAL, R. 2004. Geodesic mean shift. In *Proceedings of the 5th Korea Israel conference on Geometric Modeling and Computer Graphics*, 51–56.
- SHAMIR, A. 2004. A formulation of boundary mesh segmentation. In *Proceedings of Second International Symposium on 3D Data Processing, Visualization and Transmission*, 82–89.
- SHLAFMAN, S., TAL, A., AND KATZ, S. 2002. Metamorphosis of polyhedral surfaces using decomposition. *Computer Graphics Forum* 21, 3, 219–228.
- SRINARK, T., AND KAMBHAMETTU, C. 2003. A novel method for 3D surface mesh segmentation. In *Proceedings of the 6th Intl. Conf. on Computers, Graphics and Imaging*, 212–217.

- SURAZHISKY, V., ALLIEZ, P., AND GOTSMAN, C. 2003. Isotropic remeshing of surfaces: a local parameterization approach. In *Proceedings of 12th Intl. Meshing Roundtable*, 215–224.
- SURAZHISKY, V., SURAZHISKY, T., KIRSANOV, D., GORTLER, S., AND HOPPE, H. 2005. Fast exact and approximate geodesics on meshes. In *Proceedings of SIGGRAPH*, 553–560.
- VÁRADY, T., MARTIN, R. R., AND COX, J. 1997. Reverse engineering of geometric models - an introduction. *Computer Aided Design* 29, 4, 255–268.
- WITKIN, A., AND HECKBERT, P. 1994. Using particles to sample and control implicit surfaces. In *Proceedings of SIGGRAPH*, 269–277.
- WU, K., AND LEVIN, M. D. 1997. 3D part segmentation using simulated electrical charge distributions. *IEEE Transaction on Pattern Analysis and Machine Intelligence* 19, 11, 1223–1235.
- YAMAUCHI, H., LEE, S., LEE, Y., AND OHTAKE, Y. 2005. Feature sensitive mesh segmentation with mean shift. In *Proceedings of Shape Modeling International*, 236–243.
- ZHANG, Y., PAIK, J., KOSCHAN, A., ABIDI, M. A., AND GORSICH, D. 2002. A simple and efficient algorithm for part decomposition of 3-D triangulated models based on curvature analysis. In *Proceedings of Intl. Conf. on Image Processing*, III, 273–276.
- ZUCKERBERGER, E., TAL, A., AND SHLAFMAN, S. 2002. Polyhedral surface decomposition with applications. *Computers & Graphics* 26, 5, 733–743.