Mesh segmentation – A comparative study

M. Attene
IMATI-CNR
attene@ge.imati.cnr.it

S. Katz
TECHNION
sagikatz@tx.technion.ac.il

M. Mortara
IMATI-CNR
michela@ge.imati.cnr.it

G. Patané
IMATI-CNR
patane@ge.imati.cnr.it

M. Spagnuolo
IMATI-CNR
spagnuolo@ge.imati.cnr.it

A. Tal
TECHNION
ayellet@ee.technion.ac.il

Abstract

Mesh segmentation has become an important component in many applications in computer graphics. In the last several years, many algorithms have been proposed in this growing area, offering a diversity of methods and various evaluation criteria. This paper provides a comparative study of some of the latest algorithms and results, along several axes. We evaluate only algorithms whose code is available to us, and thus it is not a comprehensive study. Yet, it sheds some light on the vital properties of the methods and on the challenges that future algorithms should face.

1 Introduction

Mesh segmentation has become an important and challenging problem in computer graphics, with applications in areas as diverse as modeling [7], metamorphosis [10, 35], compression [13], simplification [6], 3D shape retrieval [36, 29], collision detection [18], texture mapping [17] and skeleton extraction [15, 4].

Mesh, and more generally shape, segmentation can be interpreted either in a purely geometric sense or in a more semantics-oriented manner. In the first case, the mesh is segmented into a number of patches that are uniform with respect to some property (e.g., curvature or distance to a fitting plane), while in the latter case the segmentation is aimed at identifying parts that correspond to relevant features of the shape. Methods that can be grouped under the first category have been presented for example in [9, 26, 6, 33], and may serve as a pre-processing for the recognition of meaningful features. Semantics-oriented approaches to shape segmentation have gained a great interest recently in the research community [5, 20, 28, 19, 34, 32, 15, 18, 16, 14, 25], because they can support parametrization or re-meshing schemes, metamorphosis, 3D shape retrieval, skeleton extraction as well as the *modeling by composition* paradigm that is based on natural shape decompositions.

It is rather difficult, however, to evaluate the performance of the different methods with respect to their ability to segment shapes into meaningful parts. This is due to the fact that the majority of the methods used in computer graphics are not devised for detecting specific features within a specific context, as for example is the case of form-feature recognition in product modeling and manufacturing [3]. Also, the shape classes handled in the generic computer graphics context are a broadly varying category: from virtual humans to scanned artefacts, from highly complex free-form shapes to very smooth and feature-less objects. Moreover, it is not easy to formally define the meaningful features of complex shapes in a non-engineering context and therefore the comparison of the different methods is mainly qualitative. Finally, shape segmentation methods are usually devised to solve a specific application problem, for example retrieval or parametrization, and therefore it is not easy to compare the efficacy of different methods for the shape segmentation itself.

These are the main motivations of the proposed paper, which has four goals. First, we review some of the latest algorithms, making their strengths and weaknesses evident with respect to their performance on different shape classes. Second, we identify several axes along which a segmentation algorithm can be evaluated. Third, we wish to provide an initial benchmark that consists of models from diverse domains, and hope that future papers on mesh segmentation will take these models into account when running their experiments. Last but not least, based on the results, we discuss current challenges in mesh segmentation.

In particular, the paper provides a comparative study of five algorithms, as described below. We evaluate only algorithms whose codes are available to us, and thus it is not a comprehensive study.

1. *Mesh decomposition using fuzzy clustering and cuts* [15]. The key idea of this algorithm is to first find the meaningful components using a clustering algorithm, while keeping the boundaries between the components fuzzy. Then, the algorithm focuses on the small fuzzy areas and finds the exact boundaries which go along the features of the object.

2. *Mesh segmentation using feature point and core extraction* [14]. This approach is based on three key ideas. First, *Multi-Dimensional Scaling (MDS)* is used to transform the mesh vertices into a pose insensitive representation. Second, *prominent feature points* are extracted using the MDS representation. Third, the core component of the mesh is found. The core along with the feature points provide sufficient information for meaningful segmentation.

3. *Tailor: multi-scale mesh analysis using blowing bubbles* [23]. This method provides a segmentation of a shape into clusters of vertices that have a uniform behavior from the point of view of the shape morphology, analyzed at different scales. The main idea is to analyze the shape by using a set of spheres of increasing radius, placed at the vertices of the mesh; the type and length of the sphere-mesh intersection curve are good descriptors of the shape and can be used to provide a multi-scale analysis of the surface.

4. *Plumber: mesh segmentation into tubular parts* [24]. Based on the *Tailor* shape analysis, the *Plumber* method decomposes the shape into tubular features and body components and extracts, simultaneously, the skeletal axis of the features; tubular features capture the elongated parts of the shape, protrusions or wells, and are well suited for articulated objects.

5. *Hierarchical mesh segmentation based on fitting primitives (HFP)* [1]. Based on a hierarchal face clustering algorithm, the mesh is segmented into patches that best fit a pre-defined set of primitives; in the current prototype, these primitives are planes, spheres, and cylinders. Initially each triangle represents a single cluster; at each iteration, all the pairs of adjacent clusters are considered, and the one that can be better approximated with one of the primitives forms a new single cluster. The approximation error is evaluated using the same metric for all the primitives, so that it makes sense to choose which is the most suitable primitive to approximate the set of triangles in a cluster.

The set of models examined in this work are medical models, CAD models, models of human figures in various postures, models of animals, and a miscellanea class of shapes.

The rest of the paper is structured as follows. Section 2 outlines the algorithms. Section 3 specifies several evaluation criteria that can be used when describing and analyzing segmentation algorithms. Section 4 presents the results and discusses them according to these criteria. Section 5 concludes and discusses future challenges.

2 Algorithms

The following discusses the methods that will be compared in Section 3

2.1 Hierarchical mesh decomposition using fuzzy clustering and cuts [15]

The algorithm proposed in [15] proceeds from coarse to fine. Each node in the hierarchy tree is associated with a mesh of a particular patch and the root is associated with the whole input object. At each node, the algorithm determines a suitable number of patches k, and computes a k-way decomposition of this node.

A key idea of the algorithm is to first find the meaningful components, while keeping the boundaries between the components fuzzy. Then, the algorithm focuses on the small fuzzy areas and finds the exact boundaries which go along the features of the object.

To find fuzzy components, the condition that every face should belong to exactly one patch is relaxed, and fuzzy membership is allowed. In essence, this is equivalent to assigning each face a probability of belonging to each patch. The algorithm consists of four stages:

1. Assigning distances to all pairs of faces in the mesh, based on their geodesic distance as well as on their "angular distances".

2. After computing an initial decomposition, assigning each face a probability of belonging to each patch, using the distance values computed in the previous stage, as illustrated in Figure 1(a).

3. Computing a fuzzy decomposition by refining the probability values using an iterative clustering scheme, as shown in Figure 1(b).

4. Constructing the exact boundaries between the components using a minimum cut algorithm, thus transforming the fuzzy decomposition into the final one, as demonstrated in Figure 1(c) .

2.2 Mesh segmentation using feature point and core extraction [14]

The algorithm proposed in [14] produces hierarchical segmentations, with special emphasis on producing seg-

(a) probabilities (b) fuzzy decomp. (c) decomp.

Figure 1. Algorithm [15] outline.

mentation that are insensitive to pose and proportions. The approach is based on three key ideas: the transformation of the mesh vertices into a pose invariant representation, the robust extraction of *prominent feature points*, and the extraction of the core component of the mesh.

The algorithm proceeds from coarse to fine. For each node in the hierarchy tree, the algorithm consists of the following stages.

1. **Mesh coarsening:** Mesh coarsening is applied as a pre-processing step [8]. It assists not only in accelerating the algorithm when executed on large meshes, but also in decreasing the sensitivity of the algorithm to the presence of noise.

2. **Pose insensitive representation:** Multi-dimensional scaling is used to transform the mesh S into a canonical mesh S_{MDS}, as shown in Figure 2(a). Euclidean distances between points on S_{MDS} are similar to the geodesic distances between their corresponding points on S. This property makes the representation pose insensitive, because folded organs (i.e., arms) are "straightened" up by the transformation.

3. **Feature point detection:** A few points, the *prominent feature points*, are computed on S_{MDS}, and mapped back to their corresponding points on S, as illustrated in Figure 2(b). Intuitively, points on the tips of components, such as the tail, the legs and the head of an animal, are prominent feature points. The algorithm is based on the observation that feature points can be characterized by local as well as global conditions, in terms of their geodesic distances.

4. **Core component extraction:** The core component is extracted using a new *spherical mirroring* operation (Figure 2(c)).

5. **Mesh segmentation:** The algorithm computes the other segments, each representing at least one feature point, as illustrated in Figure 2(d).

6. **Cut refinement:** The boundaries between the segments, which were found in the previous stage, are refined. The goal is to find the boundaries that go along the "natural" seams of the mesh.

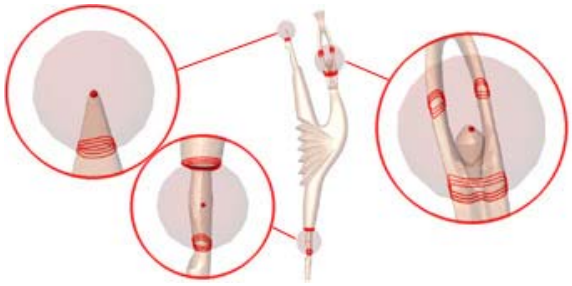
7. **Mesh refinement:** After the segmentation of the coarse-resolution mesh (Step 1) is computed, it is mapped to the input, fine-resolution mesh, and the cut is refined again, similarly to Step 6.

The hierarchical segmentation continues as long as the current segment S_i has feature points and the ratio between the number of vertices contained in the convex hulls of both S_i and $S_{i_{MDS}}$ and the total number of vertices is low (typically, 0.5). These conditions prevent situations in which objects without prominent components (i.e., almost convex objects), get further segmented.

2.3 Tailor: a multi-scale shape analysis using blowing bubbles [23]

The method proposed in [23] tries to merge global and local descriptors of shape features by using the paradigm of *blowing bubbles*. Given a 3D mesh $\mathcal{M}$ and a set of radii R_i, $i = 1, \ldots, n$ let $S_i = S(p, R_i)$ be the sphere of radius R_i and center p, and γ_i the boundary of the region of $\mathcal{M}$ containing p delimited by the intersection curves between the mesh and S_i. The study of the evolution of γ_i as function of R_i forms the core of the *Tailor* approach. The first morphological characterization of the surface in a 3D neighborhood of a vertex p at scale R_i is given by the number of connected components of γ_i. We consider the following cases:

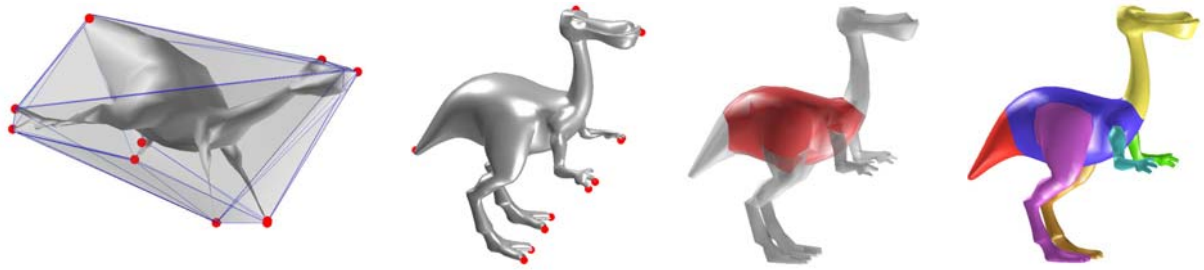
- *1 component*: the surface around p can be considered topologically equivalent to a disc (see Figure 3(a)),
- *2 components*: the surface around p is tubular-shaped (see Figure 3(b)),
- $n \geq 3$ *components*: in a neighborhood of p a branching of the surface occurs (see Figure 3(c)).



(a) (b) (c)

Figure 3. Different cases of sphere to surface intersection: (a) one, (b) two, and (c) three or more intersection curves.

The number of intersections between the spheres and the shape boundary gives a first qualitative characterization of



(a) MDS transform and its convex hull (b) Feature points (c) Core (d) Segment. & Cut refin.

Figure 2. Algorithm [14] outline.

the shape in a 3D neighborhood of each vertex. For example, for a thin limb, that intersection will start simply connected and will rapidly split into two components. For a point on the tip of a limb, that intersection will usually simply remain connected, but the ratio of its length to the radius of the bubble will be decreasing. For a point on a blend, that ratio will exceed 2π.

More precisely, if γ has only one connected component, then the curvature of the surface around v at scale R is approximated by the non-negative ratio $G_r(v) := l_\gamma / r$ [11], where l_γ is the length of γ. Furthermore, v is classified as *planar* if $G_r(v) \approx \alpha$, *sharp* if $G_r(v) < \alpha$, and *blend* if $G_r(v) > \alpha$, where α is a given threshold.

Let us now suppose that γ has two connected components, and in this case the vertices are labeled as *limb*. The vertex v at scale R is classified as *cylindrical* when the ratio between the maximal and minimal length of γ_1 and γ_2 does not exceed a given threshold ϵ, that is $l_{\gamma_1} \leq \epsilon l_{\gamma_2}$; otherwise, it is labeled as *conical*. If γ has 3 or more connected components, v is a *branching* and we do not consider other geometric descriptors.

The set of radii is automatically set by uniformly sampling the interval between the minimum edge length and the diagonal of the bounding box of $\mathcal{M}$. These parameters, as well as those ones used for the classification of the vertices (i.e., $\alpha := 2\pi$, $\epsilon := 2$), can be selected by the user if an a-priori information on the input shape is available or if he/she is searching for some specific configurations (e.g., vertices whose sharpest angle is less than a given value). The choice of α and ϵ can obviously take into account a specific application context, as for example specified in [22].

After this labeling step, a number of other properties are evaluated in order to refine the classification into specific features types, such as sharp protrusions or wells, mounts or dips, blends or branching parts, as summarized in Table 1. The main issue in this case is to distinguish properly if the point is on a feature which is on a convex or concave area of the shape, based on the orientation of the surface.

Table 1. Morphological feature characterization.

Feature	Color	$\#\cap$	Geo-metric	Status
TIP	red	1	sharp	convex
PIT	blue	1	sharp	concave
MOUNT	orange	1	rounded	convex
DIP	cyan	1	rounded	concave
BLEND	pink	1	blend	–
LIMB	yellow	2	cylindrical	full
WELL	violet	2	cylindrical	empty
JOINT	brown	2	conical	full
FUNNEL	gray	2	conical	empty
SPLIT	green	≥ 3	–	–

Small radii can be used to determine detail features, while bigger ones are used to analyze the global characteristics of the surface. From these considerations, it follows that the choice of R_i is related to the scale of the features which have to be extracted. The use of a set of increasing radii is suitable for performing a multi-scale analysis of the shape over neighborhoods of variable size, by taking into account also the morphology of the shape in that neighborhood (see Figure 4).

2.4 Plumber: shape segmentation into tubular parts [24]

Plumber specializes the *Tailor* approach to the segmentation of a shape into generic body components and tubular features. At the first step, seed vertices are located and clustered to form candidate seed regions which are then used to compute the first reliable tube section, called the *medial loop*. This loop is ensured to be around each candidate tube and works as a generator of the tubular feature. Then, the medial loop is moved in both directions on the shape, by us-

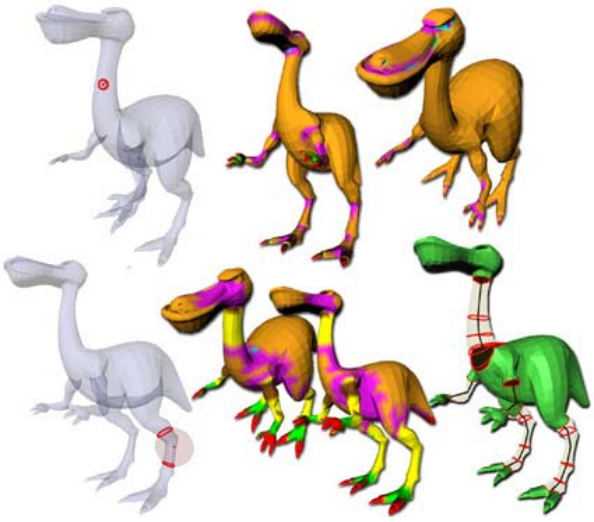


Figure 4. *Tailor* **characterization at two different scales: colors are those described in Table 1. With the smallest scale, we identify the eyes, mouth, and blend regions; the next scale locates seed tubular regions, depicted in yellow, used by** *Plumber* **to detect the tubular features along with their tubular axes.**

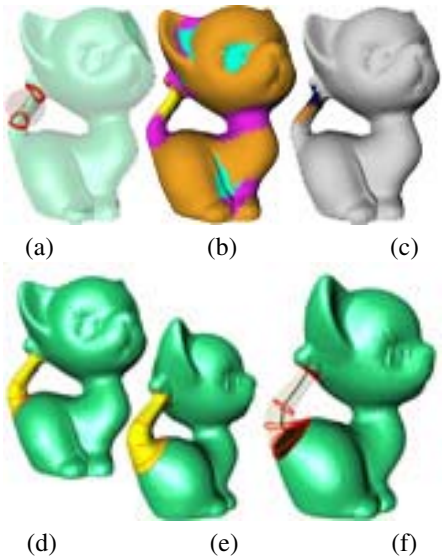


Figure 5. (a) Selection of a level of detail R, (b) classification of vertices and identification of a seed limb region, (c) medial loop, (d-e) extraction and (f) abstraction of the tubular feature as a skeletal line and a set of contours.

ing spheres placed not on the surface but at the barycenter of the medial loop iteratively and until the tube is completely swept, according to some stopping criteria. The tube detection works in a multi-scale setting, starting with the extraction of small tubes first.

Assuming that the shape is represented by a triangle mesh $\mathcal{M}$ and that we are using a set of levels of detail $\{R_i\}_i$, the steps of the segmentation are the following. For each vertex $v \in \mathcal{M}$ and scale R_i, we first apply the *Tailor* analysis and consider all vertices that are labeled as *limb*, that is, vertices where the intersection curve γ_i has two connected components (see the previous section). The vertex classification is used for the identification, at each scale, of the *seed limb-regions*, which are defined as the maximal edge-connected regions of limb-vertices with respect to a depth-first search (see Figure 5(a-c)). These regions have the shape of generalized cones or cylinders. Then, we compute the *medial loop* of each seed limb-region; a medial loop represents the generator of the feature and is used for its expansion until some stop criteria are satisfied. Once all the tubular features at a given scale are identified, the process is iterated on $\mathcal{M}$ by considering the next level of detail. In Figure 5(d), the medial loop is the boundary of the dark region, while the growing phase is shown in 5(e).

The radius, or scale, of the sphere influences two steps of the tube recognition process: once for the morphological analysis, to locate the *limb* vertices and candidate tube regions, and once for the tube growing phase. The stop condition of the tube sweeping phase is decided by a threshold on the variation of the intersection length, by the ending of the tubular feature itself, or by the splitting of the tube at a branching site. If the tubular feature ends, the tube is called *cap* and it will have only one boundary, as it is shaped as a generalized cone.

The extraction of tubes adopts a fine-to coarse strategy, marking triangles as visited while the tube grows so that they are not taken into account at the following steps. At the end of the whole process, tubes are labeled with respect to the scale at which they were found. The connected components of the shape which are not classified as tubular features define the body parts of the input surface.

The described segmentation method is robust to noise, to uneven vertex distributions, and to irregular connectivity.

2.5 Hierarchical Segmentation based on Fitting Primitives (HFP) [1]

In [1], a hierarchical segmentation algorithm for triangle meshes that is based on fitting primitives belonging to an arbitrary set, is described.

The algorithm has the same structure of the hierarchical face clustering proposed in [9], in which only fitting planes

are searched, but it has been designed to provide more flexibility, and can incorporate support for several fitting primitives [31, 30] and error metrics. This method is completely automatic and generates a binary tree of clusters, each of which fitted by one of the primitives employed.

Initially, each triangle represents a single cluster; at every iteration, all the pairs of adjacent clusters are considered, and the one that can be better approximated by one of the primitives forms a new single cluster. The approximation error is evaluated using the same metric for all the primitives, so that it makes sense to choose which is the most suitable primitive to approximate the set of triangles in a cluster. Based on this framework, a prototype system has been implemented which uses planes, spheres and cylinders fitted using a standard L^2 metric. Such a system proved to be extremely efficient and robust to noise. If the model is known to be made of a well defined set of primitives, as typical, for example, for mechanical objects, the algorithm may accept a plug-in for each of them in which the computation of both the fitting parameters and the error are implemented. Moreover, being a greedy method, the level of accuracy is somehow reflected by the cluster hierarchy which, once computed, may be interactively navigated by the user through a slider which sets the desired number of clusters or a threshold error, as shown in Figure 6.

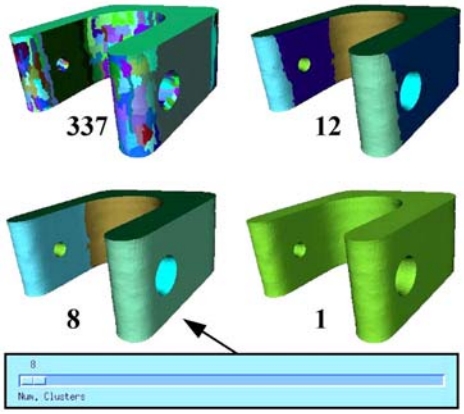


Figure 6. Various clustering resolutions for the same model (the number of clusters is indicated.

3 Evaluation Criteria

There are various ways to evaluate the quality of a segmentation. We hereby define several possible criteria along which segmentation algorithms can be evaluated.

1. *Type of segmentation:* It is common [27] to divide segmentation algorithms into algorithms that segment into meaningful components (i.e., following the minima rule [12]) and those that segment into disk-like components, or more generally, into purely geometric shapes.

2. *Extracting the "correct" segments:* Defining the "correct" components of a given model is impossible. For instance, which of the segmentations in Figure 8 is the correct one? The right segmentation depends both on the application and on the viewer's perspective and knowledge of the world (i.e., the hat of the model of Santa in Figure 8). Judging the "correctness" of the segments can be only done by looking at the images themselves.

3. *Boundaries:* Defining the "correctness" of the boundaries between segments is also infeasible. For instance, in the figure of the Color Plate, should the neck belong to the segment of the head or to the segment of the body?

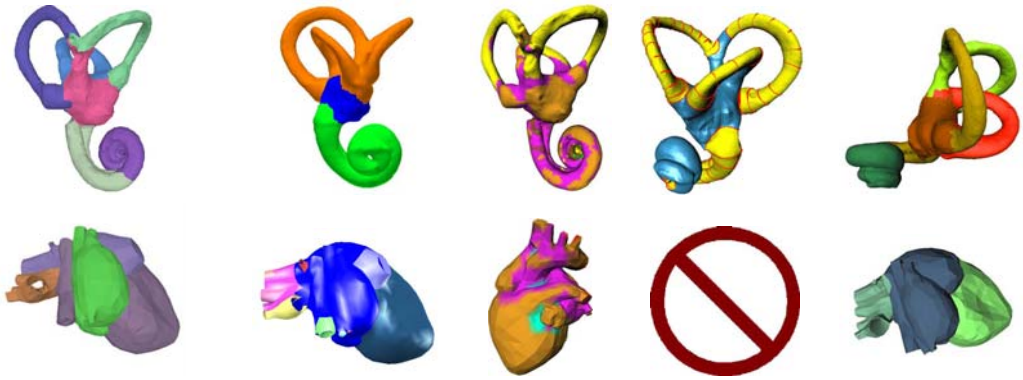
   However, given a set of segments as produced by a segmentation algorithm, desirable geometric properties of the boundaries can be defined. Such properties can include the smoothness of the boundary, the length of the boundary, and its location along concave features.

4. *Hierarchical / multi-scale segmentation:* It is rarely the case that a single segmentation of a model would exactly fit the segmentation the user "has in mind". Therefore, most segmentation algorithms are either hierarchical or multi-scaled.

5. *Sensitivity to pose:* For some applications, such as skeleton extraction, metamorphosis and retrieval, it is important that models of similar objects in different poses, will be segmented compatibly. The figure of the Color Plate demonstrates this situation.

6. *Sensitivity to noise and tesselation:*

   Mesh coarsening is applied as a pre-processing step [8] and it reduces the sensitivity of the algorithm to the presence of noise. The *Plumber* segmentation method is robust to noise and independent of the vertex sampling and connectivity regularity. In fact, the computation of the intersection curves among $\mathcal{M}$ and the selected set of spheres uses the connectivity structure only for the computation of γ, while the classification of a vertex p as belonging to a tube at scale r (i.e., $\|p-c\|_2 \leq r$, with c center of the current sphere) relies only on the set of vertices (ref. Figure 3 in [22]). As described in [2], the hierarchical clustering may be actually used to remove the noise by snapping the vertex positions onto the corresponding fitting primitive(s). Also, the algorithm proved to be robust to uneven sampling (ref. Figure 9 in [2]).



Katz & Tal 03 | Katz et al. 05 | *Tailor* | *Plumber* | HFP

Figure 7. Segmentations of medical data.

7. *Asymptotic complexity:* It is common to mention the running times of algorithms on various models. Even though the running time is an important factor, it depends on the implementation and on the exact platform the algorithm is running on, which makes it difficult to compare. We therefore discuss here the asymptotic complexity, which gives an indication of the worst-case running time.

8. *Control parameters:* The number and type of control parameters gives some indication as to the interaction needed to produce high–quality segmentations.

4 Results

Figures 7–8 show some results. In particular, Figure 7 shows segmentations of medical data – an inner part of the ear and a heart. Figure 9 shows segmentations of CAD data. Figure 10 illustrates segmentations of a variety of animals. The figure of the Color Plate demonstrates the segmentations of human figures in various poses. Finally, Figure 8 shows the result of the segmentation on a number of miscellanea models. If one of the algorithms does not produce results on a specific example, we do not insert any picture.

For each of the models, the results are displayed side by side, so that the reader can evaluate directly the qualitative performance of the method with respect to a specific application. Hereafter, the results are compared along the axes proposed in Section 3.

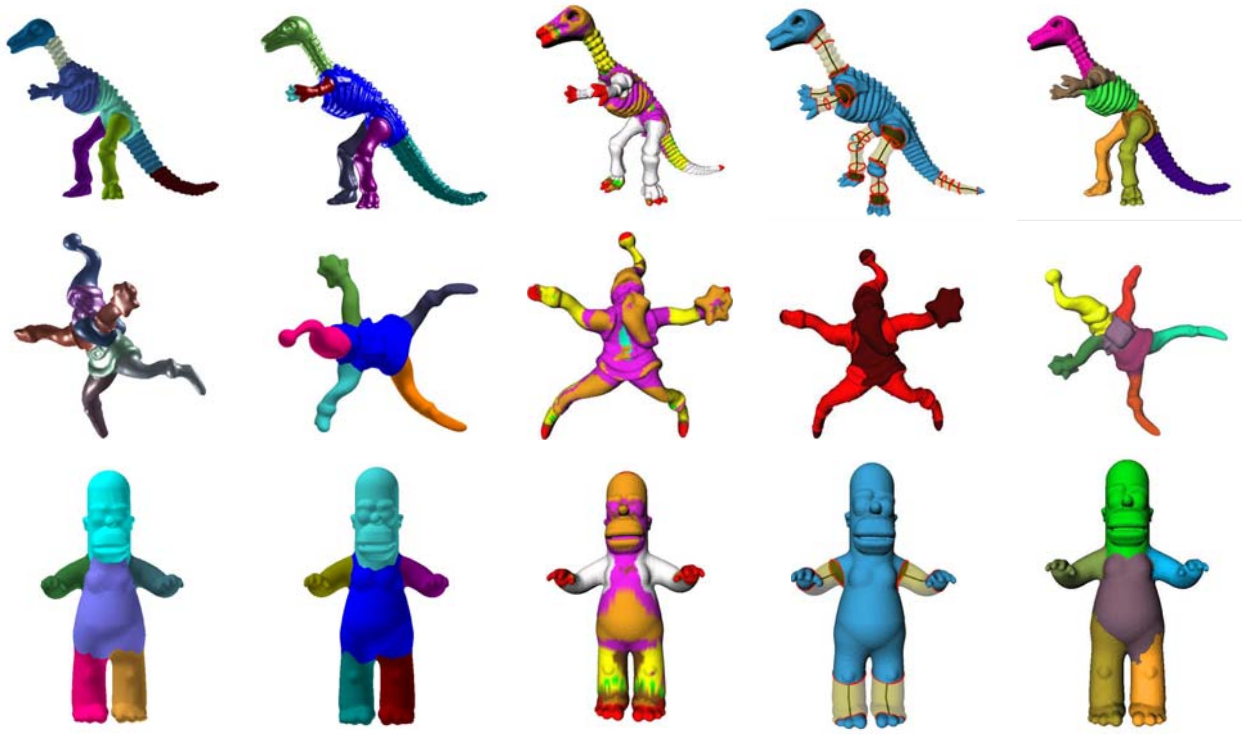
1. *Type of segmentation:* As best illustrated in Figure 9, two of the algorithms ([15, 14]) are designed to accommodate the minima rule, while the other algorithms ([23, 24, 1]) are designed to accommodate with certain geometric properties. CAD applications might find the latter category more appropriate, while applications that use "natural" data (i.e., articulated object applications such as skeleton extraction) might find the former class more appropriate.

2. *Extracting the "correct" segments:* For CAD models, the segmentation of the surface into patches of simple geometry is usually considered a pre-processing for the more complex recognition of form-features; in this context it is possible, or easier, to define precise geometric and morphological rules to detect certain configurations, even if the problem is not fully solved [3]. Methods like *Plumber* are also based on an a-priori knowledge about the features we want to extract, that are in this case defined as generalized sweep-like features. *Plumber*, indeed, performs better on features with elongation axis larger than section axis: in the tiger model, for instance, only the tail is recognized correctly as a tubular feature while the body (see Figure 10) is not identified as a tubular feature because its section is almost equivalent to its length. The inner ear (see Figure 7) is well segmented and the sections of the tube follow the sweep direction of the feature.

   For articulated objects that are used in applications such as skeleton extraction, metamorphosis and retrieval, it is expected that the meshes be segmented at their joints. In this case, deep concavities as well as the size of the components, indicate the locations of segment boundaries. See Figures 10- 8.

3. *Boundaries:* As discussed in [15], a segmentation can be partitioned into two sub-problems: the extraction of the segments (see the previous item) and the smooth refinement of the cuts. There are, however, methods, such as *Plumber* that guarantee by definition a smooth boundary.

   For other methods, that do not inherently produce smooth boundaries, a post-processing stage that refines



Katz & Tal 03 | Katz et al. 05 | *Tailor* | *Plumber* | HFP

Figure 8. Segmentations of miscellanea models.

the boundaries can be added. This was done, for instance in [15] and [14], where a minimum cut algorithm was applied to the initial segmentation.

It is important to mention, however, that not all applications require smooth boundaries.

4. *Hierarchical / multi-scale segmentation:* Among the algorithms studied in this paper, [15], [14] and [1] produce hierarchical segmentations (i.e., a refined segmentation is a sub-segmentation of a coarse one), while [23] and [24] produce multi-scale segmentations.

   Multi-scale segmentations can be exploited to get a global segmentation. For example, the segmentation into patches of uniform behavior provided by *Tailor* highlights well detail-features rather than bigger shape components, but the persistence of the labeling across different scales give less sparse clusters [23].

   Hierarchical segmentations are important when consistency across the refined levels is required. This would be the case for graphics applications that use articulated objects. Examples of full hierarchical segmentations can be found in the papers describing the algorithms ([15, 14, 1]).

5. *Sensitivity to pose:* In the figure of the Color Plate, three similar models – a human model running, sitting and walking – were segmented. It can be seen that some of the algorithms are more sensitive to pose, mainly due to the role that curvatures play in the segmentation, while others are less sensitive.

   In [14] it is proposed that the model be transformed into a pose-insensitive representation (using MDS), in order to avoid sensitivity to pose. This method can be applied as a pre-processing stage in other segmentation algorithms.

6. *Asymptotic complexity:* The overall complexity of [15] is $O(V^2 \log V + IV^2)$ where V is the number of vertices and I is the number of iterations in the $K-$means algorithm. In the actual implementation, I is bounded by a constant. The complexity is dominated by distances computation that uses Dijkstra's algorithm.

   The overall complexity of [14] is $O(f^2 \times no\_iterations + m^2 \log m)$, where f is the number of faces in the coarse model (typically up to 1000 faces) and m is the number of faces in the search region of the fine (original) model.

   In [1], the computational complexity is $O(f^2)$ in the

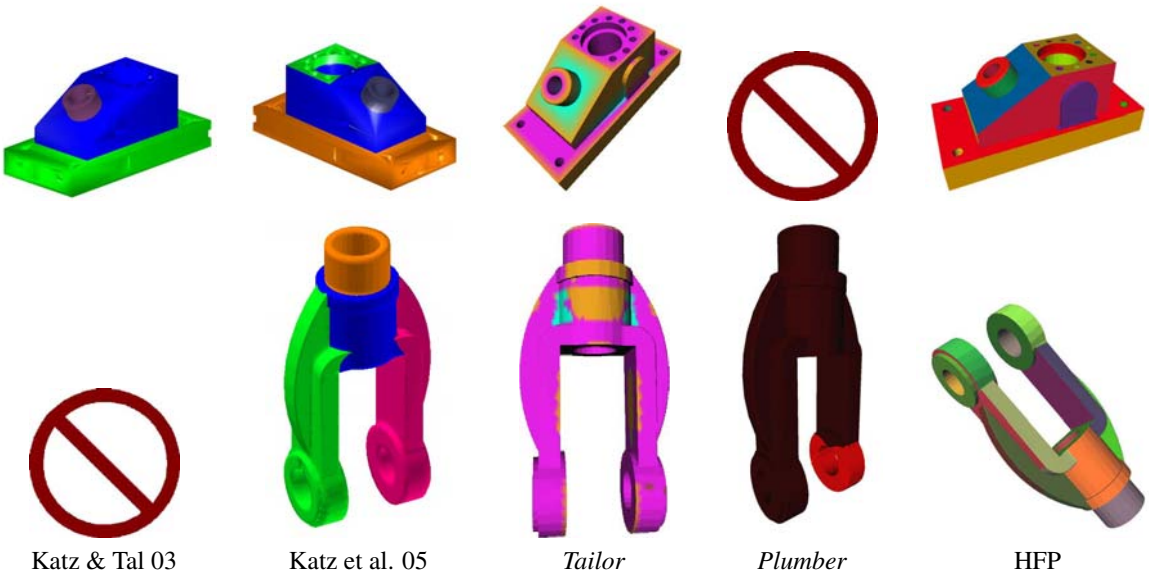


Figure 9. Segmentations of CAD data.

worst case, where f is the number of mesh faces. The worst case, however, would imply a completely unbalanced binary tree. In practical situations the tree is mostly balanced, thus the average case complexity becomes $O(f \log f)$, as reflected by the computing time observed during experimentation. Once the hierarchy is computed, it can be browsed at interactive speed through a user-controlled slider (see Figure 6). In particular, the selection of a specific level of the hierarchy (i.e., a segmentation with a user-defined number of clusters) can be performed once the hierarchy is computed at interactive speed for meshes made of up to 100k faces on a standard PC.

In *Tailor* [23], characterizing a vertex requires the analysis of a spherical neighborhood. In the worst case, the number of vertices within such a neighborhood is $O(V)$, thus characterizing all the vertices may cost up to $O(V^2)$ operations. In most practical situations, however, the size of interesting neighborhoods is much smaller than the size of the mesh, and thus the algorithm can produce results within less than a minute for $V < 20000$ on a standard PC. In *Plumber* [24], the complexity is dominated by the *Tailor* characterization, and the same considerations hold.

7. *Control parameters:* In [15], three parameters are determined by the user: the first parameter controls the importance of the geodesic distance vs. the angular distance, the second parameter controls the size of the search region for the minimum cut algorithm, and the third parameter controls the depth of the hierarchy, if needed.

   In [14], there are two parameters that concern the size of the search region for the minimum cut algorithm – one for the coarse model and the other for the refined model. There are two other parameters that the user can set, though we have not set them in the presented segmentations – the size of the coarse model and the importance of geodesic distance vs. the angular distance.

   In [1], the computation of the cluster hierarchy is fully automatic. Once it is computed, the user may select a specific clustering level out of the hierarchy in three ways: by manually browsing the tree through an interactive slider, by specifying an exact number of clusters, or by specifying the maximum approximation error allowed.

   In *Tailor* [23] and *Plumber* [24], the user is required to specify the radii representing the scales at which the characterization is performed. Also, in *Plumber* an additional parameter may be specified to change the default stopping condition for the tube generation phase; specifically, by default a tube terminates if the ratio between the length of two consecutive sections exceeds two. Increasing such a default value would make more conical features to be identified as tubes, while decreasing it would make *Plumber* identify only proper cylinders.

Figure 10. Segmentations of animals.

5 Discussion

This paper has reviewed five of the latest algorithms on mesh segmentation. These algorithms were executed on a common dataset that contains a variety of models ranging from medical data to CAD data to various human and animal models. The resulting segmentations were presented side by side.

Several criteria for evaluating segmentation have been suggested. They include the extraction of correct segments, the boundaries between segments, the type of multi-scale segmentation, the sensitivity to pose and the asymptotic complexity. Based on the experimental results, the algorithms were compared along these axes. We hope that future research works on mesh segmentation will incorporate a discussion of these properties while analyzing their algorithms. Moreover, some of the models used here can serve as an initial benchmark that consists of models from diverse domains. Hopefully, this benchmark will keep growing.

It is evident from the results presented in this paper that there is no perfect segmentation algorithm. Each algorithm has benefits and drawbacks. The search for better algorithms will undoubtedly continue.

The hidden goal of segmentation algorithms is the attempt to imitate human visual perception. Since segmentation can neither be formalized nor measured mathematically, an empirical basis for research should be provided. This can be done by collecting hand-segmentations representing the ground-truth of various models, and comparing each algorithm's results to it, similarly to the way proposed in computer vision [21].

Another major challenge for the future is the ability to base segmentation on semantics. The algorithms today produce segmentations that are semantically reasonable, yet they are not aimed at recognizing a specific part and its role (i.e., a leg).

One way to to tackle this difficult problem is to examine the behavior of existing and future segmentation methods within particular, context-specific applications. Some work has been done for *Plumber* in the context of human body models [22], but several other applications contexts are worth to be considered in the future (biomedical data, CAD data, 4-legged animals, to name a few).

Acknowledgments.

Special thanks are given to the Shape Modeling Group at IMATI-CNR. This work has been supported by the EC-IST FP6 Network of Excel-

lence "AIM@SHAPE"; models are courtesy of the AIM@SHAPE repository (*http://shapes.aimatshape.net*). *Tailor* and *Plumber* are available at *http://www.ge.imati.cnr.it/ima/smg/resources.html*.

References

[1] M. Attene, B. Falcidieno, and M. Spagnuolo. Hierarchical segmentation based on fitting primitives. *The Visual Computer*, (to appear), 2006.

[2] M. Attene, B. Falcidieno, and M. Spagnuolo. Hierarchical mesh segmentation based on fitting primitives. *The Visual Computer*, 2006, to appear.

[3] D. Bespalov, A. Shokoufandeh, W. C. Regli, and W. Sun. Local feature extraction using scale-space decomposition. In *ASME Design Engineering Technical Conferences, Computers and Information in Engineering Conference Conference (DETC 2004-57702)*. ASME Pres, Sep 2004.

[4] S. Biasotti, S. Marini, M. Mortara, and G. Patané. An overview on properties and efficacy of topological skeletons in shape modelling. In M. Kim, editor, *SMI '03: Proceedings of Shape Modeling International 2003*, pages 245–254, Los Alamitos, May 2003. IEEE Computer Society.

[5] B. Chazelle, D. Dobkin, N. Shourhura, and A. Tal. Strategies for polyhedral surface decomposition: An experimental study. *Computational Geometry: Theory and Applications*, 7(4-5):327–342, 1997.

[6] D. Cohen-Steiner, P. Alliez, and M. Desbrun. Variational shape approximation. *ACM Transactions on Graphics (SIGGRAPH)*, 23(3):905–914, 2004.

[7] T. Funkhouser, M. Kazhdan, P. Shilane, P. Min, W. Kiefer, A. Tal, S. Rusinkiewicz, and D. Dobkin. Modeling by example. *ACM Trans. Graph. (SIGGRAPH)*, 23(3):652–663, 2004.

[8] M. Garland and P. Heckbert. Surface simplification using quadric error metrics. In *Proceedings of SIGGRAPH 1997*, pages 209–216, 1997.

[9] M. Garland, A. Willmott, and P. Heckbert. Hierarchical face clustering on polygonal surfaces. In *Proceedings of ACM Symposium on Interactive 3D Graphics*, pages 49–58, 2001.

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

[11] V. Guillemin and A. Pollack. *Differential Topology*. Englewood Cliffs, New Jersey, 1974.

[12] D. Hoffman and W. Richards. Parts of recognition. In S. Pinker, editor, *Visual Cognition*, pages 65–96. MIT Press, London, 1985.

[13] Z. Karni and C. Gotsman. Spectral compression of mesh geometry. In *Proceedings of SIGGRAPH 2000*, pages 279–286. ACM SIGGRAPH, 2000.

[14] S. Katz, G. Leifman, and A. Tal. Mesh segmentation using feature point and core extraction. *The Visual Computer*, 21(8-10):865–875, 2005.

[15] S. Katz and A. Tal. Hierarchical mesh decomposition using fuzzy clustering and cuts. *ACM Trans. Graph. (SIGGRAPH)*, 22(3):954–961, 2003.

[16] Y. Lee, S. Lee, A. Shamir, D. Cohen-Or, and H.-P. Seidel. Intelligent mesh scissoring using 3d snakes. In *Pacific Conference on Computer Graphics and Applications*, pages 279–287, 2004.

[17] B. Levy, S. Petitjean, N. Ray, and J. Maillot. Least squares conformal maps for automatic texture atlas generation. In *Proceedings of SIGGRAPH 2002*, pages 362–371. ACM SIGGRAPH, 2002.

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

[19] R. Liu and H. Zhang. Segmentation of 3d meshes through spectral clustering. In *Pacific Conference on Computer Graphics and Applications*, pages 298–305, 2004.

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

[21] D. Martin, C. Fowlkes, D. Tal, and J. Malik. A database of human segmented natural images and its application to evaluating segmentation algorithms and measuring ecological statistics. In *Proc. 8th Int'l Conf. Computer Vision*, volume 2, pages 416–423, July 2001.

[22] M. Mortara, G. Patanè, and M. Spagnuolo. From geometric to semantic human body models. *Computer & Graphics, Special Issue on Shape Reasoning and Understanding*, (to appear).

[23] M. Mortara, G. Patanè, M. Spagnuolo, B. Falcidieno, and J. Rossignac. Blowing bubbles for the multi-scale analysis and decomposition of triangle meshes. *Algorithmica, Special Issues on Shape Algorithms*, 38(2):227–248, 2004.

[24] M. Mortara, G. Patanè, M. Spagnuolo, B. Falcidieno, and J. Rossignac. Plumber: A multi-scale decomposition of 3d shapes into tubular primitives and bodies. *Proc. of Solid Modeling and Applications*, pages 139–158, 2004.

[25] G. Patanè, M. Spagnuolo, and B. Falcidieno. Para-graph: Graph-based parameterization of triangle meshes with arbitrary genus. *Computer Graphics Forum*, 23(4):783–797, 2004.

[26] P. Sander, J. Snyder, S. Gortler, and H. Hoppe. Texture mapping progressive meshes. In *SIGGRAPH '01*, pages 409–416, 2001.

[27] A. Shamir. A formalization of boundary mesh segmentation. In *Proceedings of the second International Symposium on 3DPVT*, 2004.

[28] S. Shlafman, A. Tal, and S. Katz. Metamorphosis of polyhedral surfaces using decomposition. *Eurographics*, pages 219–228, September 2002.

[29] A. Tal and E. Zuckerberger. Hierarchical mesh segmentation based on fitting primitives. In *International Conference on Computer Graphics Theory and Applications*, 2006, to appear.

[30] T. Várady, R. R. Martin, and J. Cox. Reverse engineering of geometric models - an introduction. *Computer-aided Design*, 29(4):255–268, 1997.

[31] J. Wu and L. Kobbelt. Structure recovery via hybrid variational surface approximation. *Computer Graphics Forum*, 24(3):277–284, 2005.

[32] E. Zhang, K. Mischaikow, and G. Turk. Feature-based surface parameterization and texture mapping. *ACM Trans. Graph.*, 24(1):1–27, 2005.

[33] K. Zhou, J. Synder, B. Guo, and H.-Y. Shum. Isocharts: Stretch-driven mesh parameterization using spectral analysis. In *Eurographics/ACM SIGGRAPH symposium on Geometry processing*, pages 45–54, 2004.

[34] Y. Zhou and Z. Huang. Decomposing polygon meshes by means of critical points. In *MMM*, pages 187–195, 2004.

[35] M. Zockler, D. Stalling, and H.-C. Hege. Fast and intuitive generation of geometric shape transitions. *The Visual Computer*, 16(5):241–253, 2000.

[36] E. Zuckerberger, A. Tal, and S. Shlafman. Polyhedral surface decomposition with applications. *Computers & Graphics*, 26(5):733–743, 2002.