Hierarchical Mesh Decomposition using Fuzzy Clustering and Cuts

Sagi Katz and Ayellet Tal
Department of Electrical Engineering
Technion – Israel Institute of Technology
sagikatz@techunix.technion.ac.il and ayellet@ee.technion.ac.il

Abstract

Cutting up a complex object into simpler sub-objects is a fundamental problem in various disciplines. In image processing, images are segmented while in computational geometry, solid polyhedra are decomposed. In recent years, in computer graphics, polygonal meshes are decomposed into sub-meshes. In this paper we propose a novel hierarchical mesh decomposition algorithm. Our algorithm computes a decomposition into the *meaningful components* of a given mesh, which generally refers to segmentation at regions of deep concavities. The algorithm also avoids over-segmentation and jaggy boundaries between the components. Finally, we demonstrate the utility of the algorithm in control-skeleton extraction.

Keywords: Mesh decomposition, mesh segmentation, control-skeleton extraction

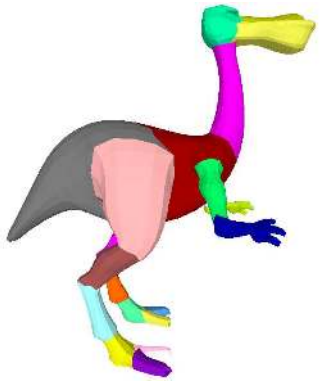


Figure 1: Decomposition of a dino-pet

1 Introduction

A hard problem might become easier if only the objects at hand could be cut up into smaller and easier to handle sub-objects. In computational geometry, solid convex decomposition has been exhaustively investigated [Chazelle and Palios 1994]. Similarly, in image processing, image segmentation has been considered a fundamental problem, which is a necessary pre-processing step for many higher-level computer vision algorithms [Sharon et al. 2000; Shi and Malik 2000]. The last few years have witnessed a growing interest in mesh decomposition for computer graphics applications [Chazelle et al. 1997; Gregory et al. 1999; Mangan and Whitaker 1999; Li et al. 2001; Shlafman et al. 2002].

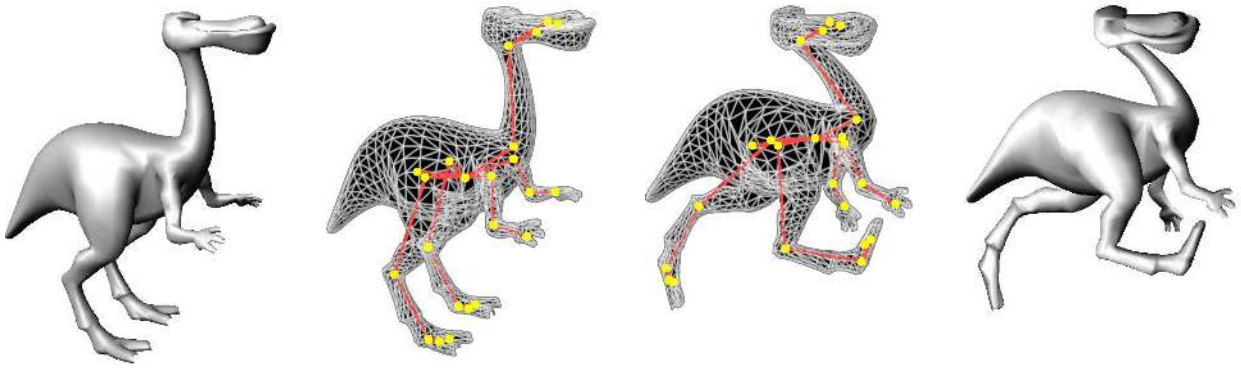
Mesh decomposition benefits many applications. In metamorphosis [Gregory et al. 1999; Zockler et al. 2000; Shlafman et al. 2002], mesh decomposition is used for establishing a correspondence. Compression [Karni and Gotsman 2000] and simplification [Garland et al. 2001; Zuckerberger et al. 2002] use decomposition for improving their compression rate. In 3D shape retrieval, a decomposition graph serves as a non-rigid invariant signature [Zuckerberger et al. 2002]. In collision detection, decomposition facilitates the computation of bounding-volume hierarchies [Li et al. 2001]. In texture mapping, parameterization is applied to each component [Levy et al. 2002]. Other potential applications include modification and modeling by parts.

Several approaches have been discussed in the past for decomposing meshes. In [Chazelle and Palios 1992; Chazelle et al. 1997] convex decomposition schemes are proposed, where a patch is called convex if it lies entirely on the boundary of its convex hull. Convex decompositions are important for applications such as collision detection. However, small concavities in the objects result in over-segmentation, which might pose a problem for other applications. In [Mangan and Whitaker 1999] a watershed decomposition is described. In this case, a post-processing step resolves over-segmentation. One problem with the algorithm is the dependency on the exact triangulation of the model. Furthermore, the meaningful components, even planar ones, might get undesirably partitioned. In [Garland et al. 2001], face clustering is proposed so that the clusters may be well approximated with planar elements. This algorithm is useful for simplification and radiosity, and less for applications seeking the meaningful components. In [Li et al. 2001], skeletonization and space sweep are used. Nice-looking results are achieved with this algorithm. However, smoothing effects might cause the disappearance of features for which it is impossible to get a decomposition. In [Shlafman et al. 2002] a K-means based clustering algorithm is proposed. The meaningful components of the objects are found. However, the boundaries between the patches are often jagged and not always correct.

In this paper we propose a new algorithm for decomposing meshes. Our work improves upon previous techniques in several aspects: our algorithm is hierarchical, it handles orientable meshes regardless of their connectivity, and avoids over-segmentation and jaggy boundaries. We elaborate below.

Previous algorithms produce "flat" decompositions. As a consequence, should the number of components be refined, the whole decomposition has to be calculated from scratch. Moreover, components which belong to a refined decomposition need not necessarily be contained in components of a coarser decomposition. A main deviation of our algorithm from previous ones is being *hierarchical*.

Another deviation of the current algorithm is the way the boundaries between components are handled. Previously, the focus has



(a) object (b) skeleton (c) deformed skeleton (d) deformed object

Figure 2: Deformation of a dino-pet

been on generating either meaningful components or components which comply with certain geometric properties. The boundaries between the components, however, were a by-product of the process. As a result, the boundaries were often too jagged [Chazelle et al. 1997; Mangan and Whitaker 1999; Shlafman et al. 2002] or too straight [Li et al. 2001] in a way that did not always fit the model. The current algorithm aims at avoiding jagginess, by specifically handling the boundaries.

Finally, the algorithm avoids over-segmentation and decomposes the objects into *meaningful components*. A meaningful component refers to a component which can be perceptually distinguished from the remaining object. The boundaries between the meaningful components pass at regions of deep concavities [Biederman 1987], as illustrated in Figure 1, where the dino-pet is decomposed into its organs. (Each component is colored differently.)

To demonstrate the usefulness of the algorithm, we show that decomposition gives rise to an automatic, general (i.e., meshes need neither be closed nor 2-manifolds), fast, and simple algorithm for extracting control-skeletons [Gagvani et al. 1998; Teichmann and Teller 1998; Bloomenthal and Lim 1999; Wade and Parent 2002]. Since skeleton extraction is done automatically, skeletal animations can be created by novice users (Figure 2).

The rest of this paper is structured as follows. Section 2 describes the problem and outlines our hierarchical decomposition algorithm. Section 3 discusses the details of the algorithm for the binary case, whereas Section 4 describes the extension to the k-way case. Section 5 shows some results. Section 6 presents the control-skeleton extraction application. Finally, Section 7 concludes and discusses future directions.

2 Overview

This section begins with a few notations and then provides an outline of our algorithm. Let S be an orientable mesh. It need neither be triangulated nor closed or a 2-manifold. (Non-manifold meshes might yield less expected results.)

Definition 2.1 ***k-way Decomposition:*** *$S_1, S_2, \ldots S_k$ is a k-way decomposition of S iff (i) $\forall i, 1 \leq i \leq k$, $S_i \subseteq S$, (ii) $\forall i$, S_i is connected, (iii) $\forall i \neq j, 1 \leq i, j \leq k$, S_i and S_j are face-wise disjoint and (iv) $\cup_{i=1}^{k} S_i = S$.*

Definition 2.2 ***Binary Decomposition:*** *S_1, S_2 is a binary decomposition of S if it is a $k-$way decomposition with $k = 2$.*

Definition 2.3 ***Patch:*** *Given $S_1, S_2, \ldots S_k$, a k-way decomposition of S, each S_i is called a patch of S.*

The algorithm 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. If the input object consists of multiple connected components, the algorithm is applied to each component separately. The examples in this paper contain a single connected component, which is the more challenging case.

A key idea of our 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, we relax the condition that every face should belong to exactly one patch, and allow fuzzy membership. 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.
2. After computing an initial decomposition, assigning each face a probability of belonging to each patch.
3. Computing a fuzzy decomposition by refining the probability values using an iterative clustering scheme.
4. Constructing the exact boundaries between the components, thus transforming the fuzzy decomposition into the final one.

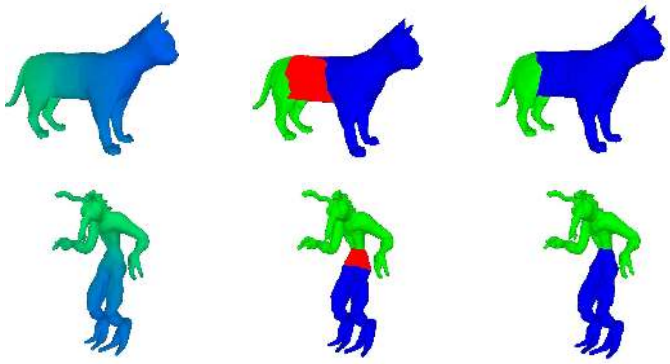
For instance, we wish to partition the objects in Figure 3 into two components. After computing distances, each polygon is assigned a probability of belonging to the patches. In Figure 3(a), a green polygon has a high probability of belonging to the back (or upper) patch. Conversely, a blue polygon has a high probability of belonging to the front (or lower) patch. The fuzzy decomposition is shown in Figure 3(b), where the fuzzy region is drawn in red. Figure 3(c) illustrates the final binary decomposition, after the exact boundaries are found.

3 Algorithm – the binary case

This section describes each stage of the algorithm for the binary case (i.e., each node in the hierarchy is decomposed into two submeshes). An extension to the $k-$way case is presented in the next section.

3.1 Computing distances

The probability that a face belongs to a certain patch depends on its distance from other faces in this patch. The underlying assumption



(a) probabilities (b) fuzzy decomposition (c) decomposition

Figure 3: Binary decomposition

is that distant faces, both in terms of their geodesic distance and of their angular distance, are less likely to belong to the same patch than faces which are close together.

Given f_i and f_j, two adjacent faces and α_{ij}, the angle between their normals, we define their angular distance to be

$$Ang\_Dist(\alpha_{ij}) = \eta(1 - \cos\alpha_{ij}).$$

When $\eta = 1$, convex and concave dihedral angles are treated equally. Since a concave feature makes a better candidate for a boundary, we check for convexity prior to computing distances. A small positive η is used for convex angles and $\eta = 1$ is used for concave angles.

Let $avg(Geod)$ be the average geodesic distance between the centers of mass of all the adjacent faces in the object, and $avg(Ang\_Dist)$ be the average angular distance between these faces. Consider the dual graph of the mesh, where every face of the mesh is a vertex in this graph and two vertices are joined by an arc if and only if their corresponding faces are adjacent. The weight of the arc connecting the dual vertices of f_i and f_j is then defined as follows:

$$\begin{aligned} &Weight(dual(f_i), dual(f_j)) = \\ &= \delta \cdot \frac{Geod(f_i, f_j)}{avg(Geod)} + (1-\delta) \cdot \frac{Ang\_Dist(\alpha_{ij})}{avg(Ang\_Dist)}. \end{aligned} \quad (1)$$

The first term is affected by the geodesic distance whereas the second term is affected by the angular distance. Note that the latter is zero when the faces are coplanar. The denominator reduces effects that may appear for similar objects having different sampling rates.

Given any pair of faces on the mesh f_l and f_m, their distance $Dist(f_l, f_m)$ is defined to be the shortest path between their dual vertices on the dual graph. The distances between all pairs of faces are calculated once, in a pre-processing step, using an *all-pair shortest paths* algorithm [Cormen et al. 2001], where the distance between faces which belong to different connected components is defined to be ∞.

3.2 Initialization and assigning probabilities

During an initialization phase, k faces which are considered the *representatives* of the k initial patches, are chosen. In the binary case, the initial pair of representatives REP_A and REP_B (representing patches A and B, respectively) is chosen such that the distance between them is the largest possible.

Our goal is to assign each face f_i its probability $P_B(f_i)$ of belonging to patch B. Let $a_{f_i} = Dist(f_i, REP_A)$ and $b_{f_i} = Dist(f_i, REP_B)$. We define $P_B(f_i)$ (and equivalently $P_A(f_i)$) as follows:

$$\begin{aligned} P_B(f_i) &= \frac{a_{f_i}}{a_{f_i} + b_{f_i}} = \\ &= \frac{Dist(f_i, REP_A)}{Dist(f_i, REP_A) + Dist(f_i, REP_B)}. \end{aligned} \quad (2)$$

It can be easily verified that if a face is closer to patch A than to B, the probability of belonging to A is larger than the probability of belonging to B, and vice-versa. Moreover, a face which is equally distant from A and from B is as likely to belong to one as to the other. Finally, $P_B(f_i) = 1 - P_A(f_i)$, $P_B(REP_B) = 1$, $P_B(REP_A) = 0$ and for all other faces $0 < P_B(f) < 1$.

3.3 Generating a fuzzy decomposition

One way to obtain a decomposition is to apply a K-means clustering scheme [Duda and Hart 1973] as done in [Shlafman et al. 2002]. Our goal, however, is to construct a *fuzzy* decomposition, thus we use fuzzy clustering.

Let p be a face representing a patch and let f be a face. The goal of our algorithm is to cluster the faces into patches by minimizing the following function

$$F = \sum_p \sum_f probability(f \in patch(p)) \cdot Dist(f, p). \quad (3)$$

During an initialization phase, a subset of k *representatives* V_k, is chosen, as described above. Then, the algorithm iterates on the following steps.

1. Compute the probabilities of faces to belong to each patch, as described in Equation 2.
2. Re-compute the set of representatives $V_{k'}$, minimizing the function in Equation 3.
3. If V_k is different from $V_{k'}$, set $V_k \leftarrow V_{k'}$ and go back to 1.

Choosing the set of new representatives (i.e., Step 2) is done by using the following formulas:

$$REP_A = \min_f \sum_{f_i} (1 - P_B(f_i)) \cdot Dist(f, f_i)$$

$$REP_B = \min_f \sum_{f_i} P_B(f_i) \cdot Dist(f, f_i).$$

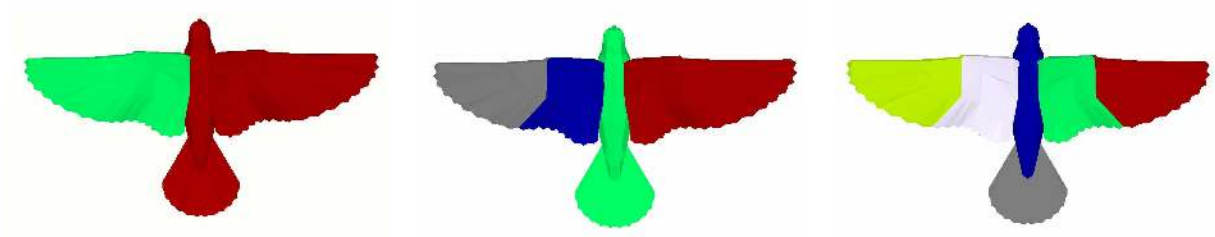
Next, if the probability of a face of belonging to a patch exceeds a certain value, it is assigned to the patch. There are, however, faces which are almost as likely to belong to one patch as to the other. In this case, the faces are considered fuzzy. In the binary case, the mesh is decomposed into three patches A, B and C, where C contains all the faces which are (almost) as likely to belong to A as to B. This is done by partitioning the faces as follows and is illustrated in Figure 3(b) where C is the red region.

$$A = \{f_i | P_B(f_i) < 0.5 - \epsilon\}$$

$$B = \{f_i | P_B(f_i) > 0.5 + \epsilon\}$$

$$C = \{f_i | 0.5 - \epsilon \le P_B(f_i) \le 0.5 + \epsilon\}$$

A practical problem which arises in this step is the dependence of the probability values on the specific representative of the patch. One way to overcome this problem is to re-define a_{f_i} and b_{f_i} using the average distances: $a_{f_i} = avg_{f_j \in A}(Dist(f_i, f_j))$ and $b_{f_i} = avg_{f_j \in B}(Dist(f_i, f_j))$. Empirically, this definition improves the results and expedites convergence.



(a) first level (b) second level (c) third level

Figure 4: Hierarchical binary decompositions of a dove

3.4 Generating the final decomposition

In the previous stage the meaningful components were found but not the exact boundary between them. The goal of the current stage is to construct this boundary within region C. Once the boundary is determined, the faces of C are assigned to either patch A or patch B.

We formulate our problem as a graph partitioning problem. Consider the dual graph of the mesh $G = (V, E)$ and the set of the dual vertices of patches A and B, V_A and V_B respectively. Our goal is to partition V into two subsets of vertices $V_{A'}$ and $V_{B'}$, such that the disassociation between $V_{A'}$ and $V_{B'}$ is minimized. We are essentially looking for a constrained minimum cut in G, requiring that:

$$(1)\quad V = V_{A'} \cup V_{B'}$$

$$(2)\quad V_{A'} \cap V_{B'} = \phi$$

$$(3)\quad V_A \subseteq V_{A'}, \qquad V_B \subseteq V_{B'}$$

$$(4)\quad weight(Cut(V_{A'}, V_{B'})) = \sum_{u \in V_{A'}, v \in V_{B'}} \omega(u, v) \quad \text{is minimal.}$$

We denote the dual graph of C by $G_C = (V_C, E_C)$ and the set of all vertices in V_A whose corresponding faces in A share an edge with faces in C by V_{CA} (resp., V_{CB}). We now construct an undirected flow network graph $G' = (V', E')$ adding two new vertices S and T, as follows (Figure 5).

$$\begin{aligned} V' &= V_C \cup V_{CA} \cup V_{CB} \cup \{S, T\} \\ E' &= E_C \cup \{(S, v), \forall v \in V_{CA}\} \cup \{(T, v), \forall v \in V_{CB}\} \cup \\ &\cup \{e_{ij} \in E | i \in V_C, j \in \{V_{CA} \cup V_{CB}\}\} \end{aligned} \quad (4)$$

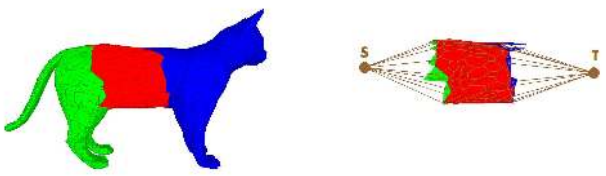


Figure 5: The flow network graph, where C is the red region

Next, the capacities of the arcs need to be defined. There are many ways to define capacities within this framework. The key principle is that the minimum cut tends to pass through arcs with small capacities. We experimented with various capacity functions, some which take only dihedral angles into account and others which also take arc length into account. We found the following function to produce good results. For two vertices v_i and v_j, let α_{ij} be the angle between the normals of their dual faces. The capacity $Cap(i, j)$ is defined as follows (where the average factor handles precision problems):

$$Cap(i,j) = \begin{cases} \frac{1}{1+\frac{Ang\_Dist(\alpha_{ij})}{avg(Ang\_Dist)}} & \text{if } \{i, j \neq S, T\} \\ \infty & \text{else} \end{cases} \quad (5)$$

A boundary between the components can now be found by applying a maximum flow (minimum cut) algorithm from S to T (e.g., [Cormen et al. 2001] [Goldberg and Tarjan 1988]). By the definition of $Cap(i, j)$, the cut tends to pass through edges having highly concave dihedral angles.

3.5 Stopping conditions

Each node in the hierarchy is recursively decomposed until at least one of the following conditions is met: (a) the distance between the representatives is smaller than a given threshold; (b) the difference between the maximal dihedral angle and the minimal dihedral angle is smaller than a threshold, so that patches having a fairly constant curvature will not be decomposed; (c) the ratio between the average distance in the patch and that of the overall object does not exceed a threshold. Since the average distance captures both the size (i.e., the geodesic distance) and the angular information, further decomposition is unnecessary when both are small relative to the original object.

Figure 4 demonstrates results of a hierarchical binary decomposition. Note how the different organs are progressively extracted.

4 Algorithm – the k-way case

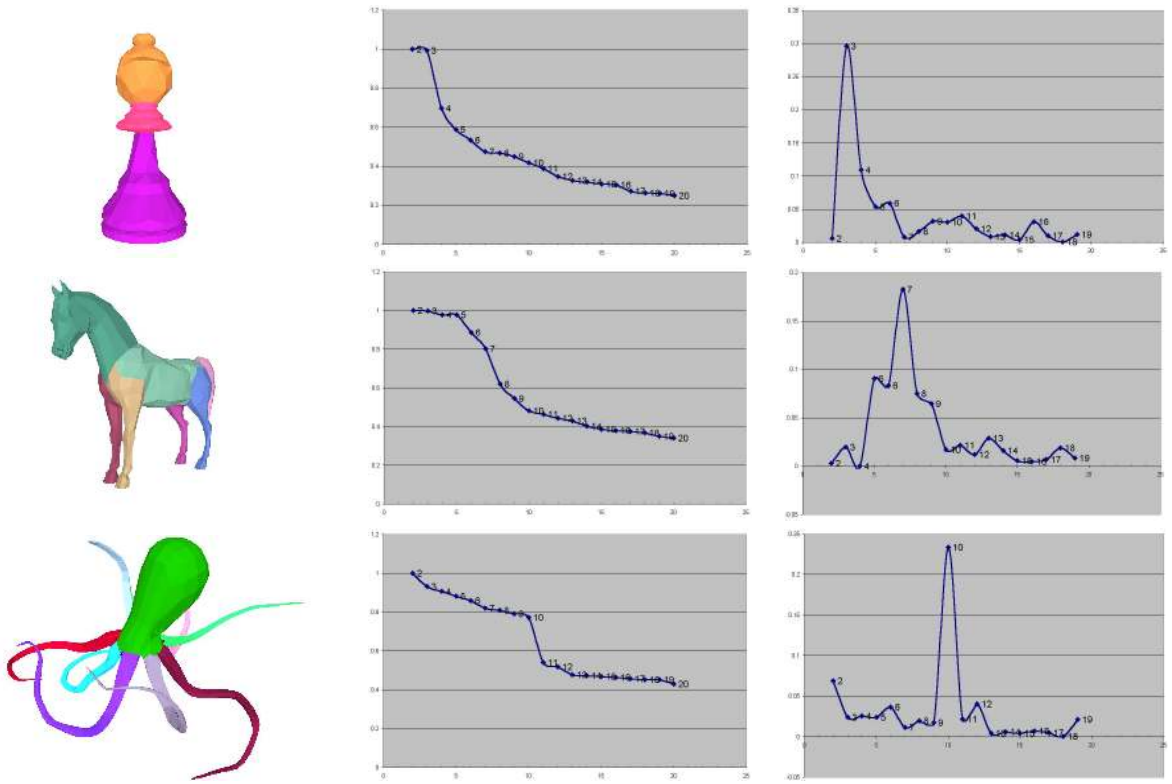
A $k-$way decomposition is a generalization of the binary case. There are, however, three issues which require explanation. The first issue is the determination of the number of patches a node in the hierarchy should be decomposed into. The second issue is the assignment of probabilities. The third issue is the extraction of the fuzzy area. We discuss these issues below.

Unlike the binary case, in the $k-$way case, the representatives are chosen iteratively. The first representative is assigned to be the face having the minimum sum of distances from all other faces. This is done in order to represent the main "body" of the object. Then, representatives are added, each in turn, so as to maximize their minimum distance from previously assigned representatives.

The remaining question is how many representatives to add. We look at the following function which is the minimum distance of the k^{th} representative from previously assigned representatives:

$$G(k) = min_{i<k}(Dist(REP_k, REP_i)).$$

Obviously, this function decreases as we add more representatives. Empirical experiments show that after assigning representatives to



(a) object (b) function G (c) first derivative of G

Figure 6: Determining the number of patches

all the major parts of an object, adding one more representative will cause a large decrease of G. This observation aids in determining the number of components k. We choose k to be the value that maximizes the first derivative of G. See Figure 6.

The second issue is the assignment of probabilities. For a representative, the probability of belonging to its own patch is defined to be 1. Otherwise, for a face f_i, the probability $P_{p_j}(f_i)$ of belonging to patch p_j is defined as:

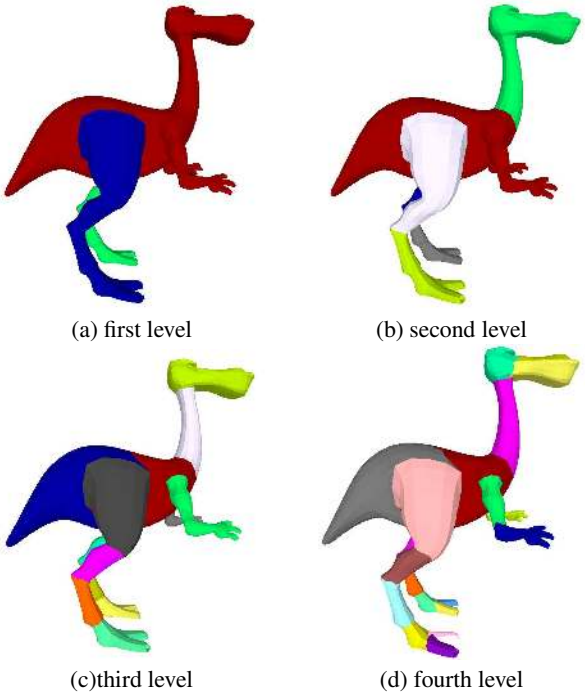
$$P_{p_j}(f_i) = \frac{\frac{1}{Dist(f_i, REP(p_j))}}{\sum_l \frac{1}{Dist(f_i, REP(p_l))}}.$$

It can be easily verified that this function is an extension of the binary case. Moreover, $P_{p_j}(f_i)$ complies with the following constraints: (1) $0 \leq P(f_i, p_j) \leq 1$, (2) the sum of the probabilities is 1, and (3) as the distance of a face from a representative increases, the probability to belong to this patch decreases.

The third issue is the extraction of the fuzzy areas once the components have been found. We consider each pair of neighboring components and proceed similarly to the binary case.

Figures 7–8 demonstrate several hierarchical k-way decompositions.

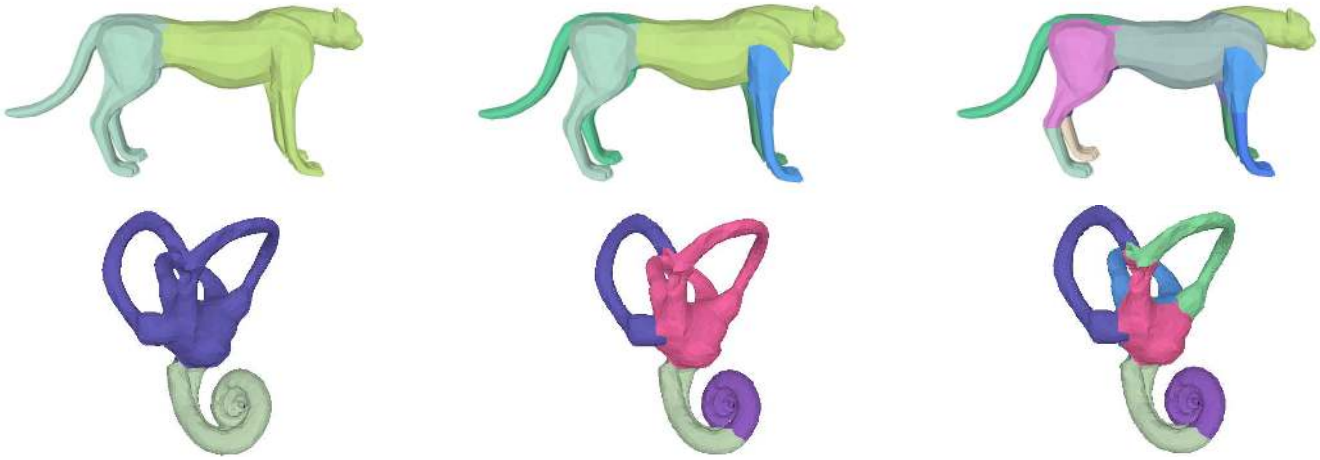
The overall computational complexity of the algorithms is $O(V^2 \log V + IV^2)$ where V is the number of vertices and I is the number of iterations in the $K-$means algorithm. The first phase, distances computation, is done once, in a pre-processing step, using Dijkstra's algorithm in $O(V^2 \log V)$. The next phases involve an iterative algorithm where faces are assigned to patches and are performed in $O(IV^2)$. (In the actual implementation I is bounded by a constant.) Finally, the minimum cut can be found in $O(V^2 \log V)$ [Goldberg and Tarjan 1988]. In our case, the minimum cuts are computed within the *fuzzy* regions. Thus, this step



(a) first level (b) second level

(c)third level (d) fourth level

Figure 7: Hierarchical k-way decomposition of a dino-pet



(a) first level (b) second level (c) third level

Figure 8: Hierarchical k-way decompositions of a cheetah and an inner part of a human ear

costs $O(C^2 \log C)$ where C is the size of the *fuzzy* region.

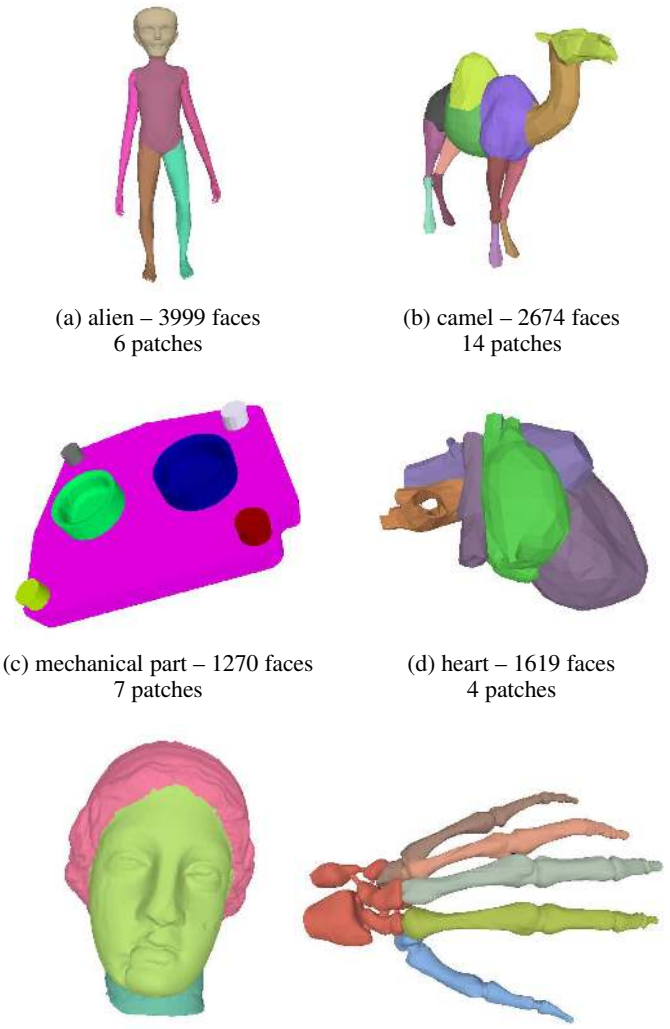
5 Results

In order to handle large models, we utilize the consistency in the way similar objects are decomposed. The procedure for decomposing large models consists of four stages. First, the model is simplified [Garland and Heckbert 1997]. Second, a decomposition is computed for the simplified model using our algorithm. Third, the boundaries found for the simplified model are used to define the *fuzzy* regions in the original model, by "projecting" the faces adjacent to the boundaries to the original model. This projection is performed by assigning each face in the original model to the patch containing the closest face in the simplified model, where voxelization is used to further accelerate this step. Fourth, the minimum cuts are computed on these fuzzy regions (in the original model), which are very small. In order to avoid erroneous projections, this process is performed progressively, using simplified models at various simplification levels.

Figure 9 shows several objects at different levels of hierarchy as decomposed by our algorithm, where for Figures 9(e)-(f), a simplification was applied, as described above. The running times for the objects described in this paper, on a $P4$, 1500 MHz, 512Mb RAM PC, vary between 1 second for the mechanical part and 57 seconds for the dino-pet (3999 faces, 4 levels of hierarchy). The running times of decomposing Venus and the skeleton hand (Figure 9(e)-(f)), using simplification as described above, are 244 seconds and 1654 seconds respectively (including loading, simplification and storing).

Figure 10 demonstrates results of three algorithms: [Li et al. 2001], [Shlafman et al. 2002] and the algorithm described here. First, notice the boundaries between the back legs and the body. Only in Figure 10(c), the "natural" ones were extracted. Second, the boundaries in Figure 10(a) tend to be very straight due to space sweeping utilized in the algorithm, whereas in Figure 10(b) the boundaries are jaggy. In Figure 10(c), the boundaries pass along the object's features. Finally, since our algorithm is hierarchical, only the major organs were found, but not the smaller ones such as the fingers and the toes. The latter would be extracted at a finer level of the hierarchy, as shown in Figure 7.

An alternative approach to ours is to use graph partitioning methods on the whole model, rather than on constrained regions. Possible cuts are *minimal cuts*. Their drawback is that they tend to favor



(a) alien – 3999 faces
6 patches

(b) camel – 2674 faces
14 patches

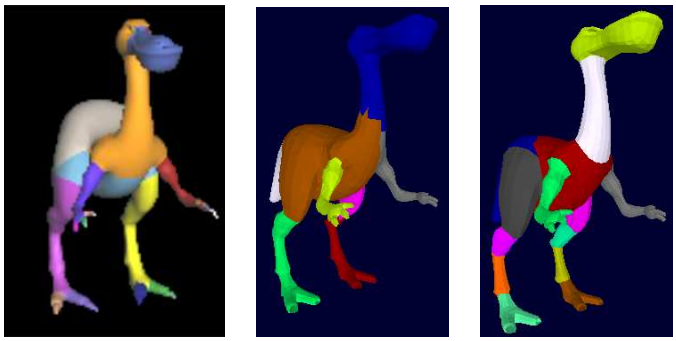
(c) mechanical part – 1270 faces
7 patches

(d) heart – 1619 faces
4 patches

(e) Venus – 67,170 faces
3 patches

(f) skeleton hand – 654,666 faces
6 patches

Figure 9: Decompositions of various objects



(a) [Li et al] (b) [Shlafman et al] (c) Our algorithm

Figure 10: Comparison to [Li et al] and [Shlafman et al]

small sets of isolated nodes since the weight of the cut increases with the number of edges [Wu and Leahy 1993].

Another option is to use normalized cuts [Shi and Malik 2000], which is an NP-complete problem. We implemented an extension of [Shi and Malik 2000] to meshes, where the weight is defined to take into account both the geodesic distance and the angular distance. The results varied. For some objects, good decompositions were produced while for other objects, the meaningful components were not found or the boundaries between them were jagged or step-wise. This can be explained by the fact that the cut approximation might cause artifacts. In addition, balancing the size of the parts need not always fit the model.

6 Control Skeleton Extraction

Control skeletons are beneficial for various applications, including matching, retrieval, metamorphosis and computer animation. Previous algorithms are based on medial surface extraction [Gagvani et al. 1998; Teichmann and Teller 1998; Bloomenthal and Lim 1999; Wade and Parent 2002], *level set diagrams* [Lazarus and Verroust 1999] or *Reeb Graphs* [Shinagawa et al. 1991]. Mesh decomposition gives rise to a novel control-skeleton extraction algorithm, where the joints are calculated directly from the hierarchical structure of the decomposition. The algorithm is general (i.e., the models need not be closed or 2-manifolds), fully automatic, simple and fast. It is thus beneficial for applications requiring automation as well as for novice users of applications where user-intervention is acceptable or desirable. Figure 11 demonstrates the use of skeletons for animating otherwise static objects.

The algorithm starts by decomposing the given model. It is essential that features which depend on the position of another feature become its descendants in the hierarchy. For instance, the elbow joint of a humanoid object should be a descendant of the shoulder joint, so that a shoulder motion will cause an elbow motion. A simple way to achieve this is to guarantee that the decomposition of every node in the hierarchy consists of a central patch connected to all other patches.

To force this star-shaped decomposition structure, the decomposition algorithm is slightly modified. We consider the central patch to be the one which contains the first assigned representative. After decomposing a given patch, the algorithm verifies that the decomposition is star-shaped. If not, a patch which is not adjacent to the central patch is merged with a neighboring patch with which it has the smallest average angle between their normals along the cut.

Once the hierarchical k-way decomposition is computed, the decomposition tree is traversed and a tree of joints is generated. At each level of the hierarchy, joints between the central patch and its adjacent patches are created. Each joint is positioned at the center of mass of the boundary between the patches. Each node in the tree of joints is associated with a list of faces. Initially, the root node is associated with the whole model. As the tree is traversed from coarse to fine, the relevant faces are transfered from a parent node to its children.

In order to animate articulated objects, it is necessary to bind the joints and pose of the object. Each vertex v_i of the mesh is assigned a weight w_{ij} indicating the extent to which it belongs to joint j. The simplest approach is to let w_{ij} be the percentage of faces that belong to joint j and adjacent to vertex v_i. This guarantees that an internal vertex is bound only to the patch it resides on, that each vertex corresponds to at least one joint and that $\sum_j w_{ij} = 1$. More advanced methods take also the cut's angles into account. Finally, to pose the object, a *skeleton-subspace deformation* method is used [Lewis et al. 2000].

Objects are deformed by adjusting the joints' angles of their skeletons. To compute the modified vertex position we use the following equation [Weber 2000]: $y_i = \sum_{j=0}^{J-1} (w_{ij} x_{ij} M_j)$, where J is the number of joints, x_{ij} is the original vector position of v_i relative to the coordinate system of joint j, and M_j is the transformation matrix of joint j. Thus, a vertex which belongs to a single joint has a constant position relative to this joint, while a vertex which belongs to more than one joint is positioned between the locations it would have, had it belonged entirely to each of the joints.

7 Conclusion

We have presented an algorithm for hierarchically decomposing meshes. The algorithm avoids jaggy boundaries as well as oversegmentation. The key idea of the algorithm is to first find the meaningful components of the mesh and only then focus on generating the exact boundaries between the components. To find the components, both geodesic distances and convexity are considered. Computing the boundaries is done by formulating the problem as a constrained network flow problem. We demonstrated the applicability of the algorithm for control skeleton extraction.

Several enhancements can be added to our algorithm. For instance, different distance functions and different capacity functions can be experimented with. Furthermore, non-geometric features, such as color and texture, can be embedded in the algorithm. We believe that the spectrum of applications which will benefit from mesh decomposition will further grow in the future. We are currently looking at compression and texture mapping.

Acknowledgments

This research was supported in part by the Israeli Ministry of Science, Culture & Sports, Grant 01-01-01509. We are grateful to Daniel Aliaga, Tom Funkhouser, Idan Hadari, Itai David and the anonymous referees for their help. We thank Huang Zhiyong, Marcelo Kallmann and Ronen Basri for letting us use their images.

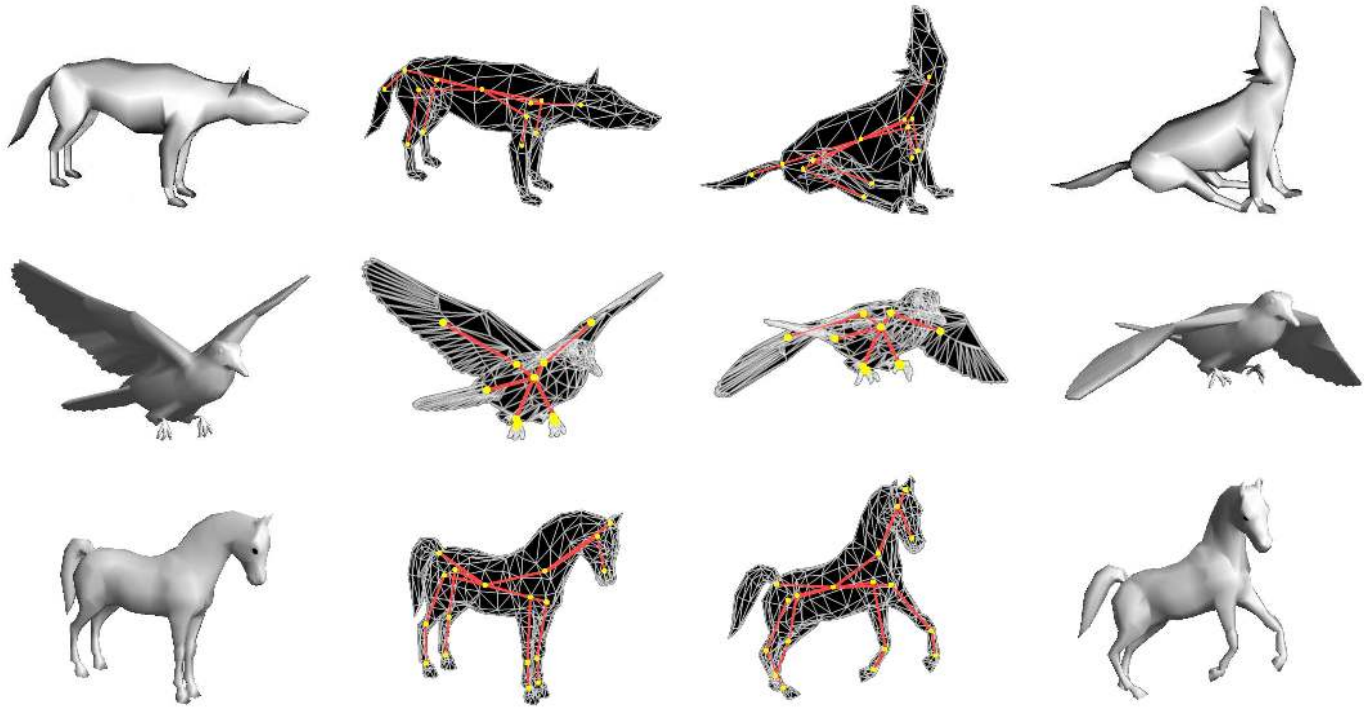
References

BIEDERMAN, I. 1987. Recognition-by-components: A theory of human image understanding. *Psychological Review 94*, 115–147.

BLOOMENTHAL, J., AND LIM, C. 1999. Skeletal methods of shape manipulation. In *International Conference on Shape Modeling and Applications*, 44–49.

CHAZELLE, B., AND PALIOS, L. 1992. Decomposing the boundary of a nonconvex polyhedron. In *SWAT*, 364–375.

CHAZELLE, B., AND PALIOS, L. 1994. Decomposition algorithms in geometry. In *Algebraic Geometry and its Applications*, Springer-Verlag, C. C. Bajaj, Ed., Ed., 419–447.



(a) object (b) skeleton (c) deformed skeleton (d) deformed object

Figure 11: Deformations of various objects

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

CORMEN, T., LEISERSON, C., RIVEST, R., AND STEIN, C. 2001. *Introduction to Algorithms*. McGraw-Hill.

DUDA, R., AND HART, P. 1973. *Pattern Classification and Scene Analysis*. New-York, Wiley.

GAGVANI, N., KENCHAMMANA-HOSEKOTE, D., AND SILVER, D. 1998. Volume animation using the skeleton tree. In *IEEE Symposium on Volume Visualization*, 47–53.

GARLAND, M., AND HECKBERT, P. 1997. Surface simplification using quadric error metrics. In *Proceedings of SIGGRAPH 1997*, 209–216.

GARLAND, M., WILLMOTT, A., AND HECKBERT, P. 2001. Hierarchical face clustering on polygonal surfaces. In *Proceedings of ACM Symposium on Interactive 3D Graphics*, 49–58.

GOLDBERG, A., AND TARJAN, R. 1988. A new approach to the maximum flow problem. *Journal of the ACM 35*, 4, 921–940.

GREGORY, A., STATE, A., LIN, M., MANOCHA, D., AND LIVINGSTON, M. 1999. Interactive surface decomposition for polyhedral morphing. *The Visual Computer 15*, 453–470.

KARNI, Z., AND GOTSMAN, C. 2000. Spectral compression of mesh geometry. In *Proceedings of SIGGRAPH 2000*, ACM SIGGRAPH, 279–286.

LAZARUS, F., AND VERROUST, A. 1999. Level set diagrams of polyhedral objects. In *ACM Symposium on Solid Modeling and Applications*, 130–140.

LEVY, B., PETITJEAN, S., RAY, N., AND MAILLOT, J. 2002. Least squares conformal maps for automatic texture atlas generation. In *Proceedings of SIGGRAPH 2002*, ACM SIGGRAPH, 362–371.

LEWIS, J., CORDNER, M., AND FONG, N. 2000. Pose space deformations: A unified approach to shape. In *Proceedings of SIGGRAPH 2000*, ACM SIGGRAPH, 165–172.

LI, X., TOON, T., TAN, T., AND HUANG, Z. 2001. Decomposing polygon meshes for interactive applications. In *Proceedings of the 2001 symposium on Interactive 3D graphics*, 35–42.

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

SHARON, E., BRANDT, A., AND BASRI, R. 2000. Fast multiscale image segmentation. In *Proceedings IEEE Conference on Computer Vision and Pattern Recognition*, 70–77.

SHI, J., AND MALIK, J. 2000. Normalized cuts and image segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence 22*, 8, 888–905.

SHINAGAWA, Y., KUNII, T., AND KERGOSIEN, Y. 1991. Surface coding based on morse theory. *IEEE Computer Graphics and Applications 11*, 5, 66–78.

SHLAFMAN, S., TAL, A., AND KATZ, S. 2002. Metamorphosis of polyhedral surfaces using decomposition. In *Eurographics 2002*, 219–228.

TEICHMANN, M., AND TELLER, S. 1998. Assisted articulation of closed polygonal models. In *Conference abstracts and applications: SIGGRAPH*, ACM SIGGRAPH, 14–21.

WADE, L., AND PARENT, R. 2002. Automated generation of control skeletons for use in animation. *The Visual Computer 18*, 2, 97–110.

WEBER, J. 2000. *Run–Time Skin Deformation*. Intel Architecture Labs, www.intel.com.

WU, Z., AND LEAHY, R. 1993. An optinal graph theoretic approach to data clustering: Theory and its application to image segmentation. *PAMI 11*, 1101–1113.

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

ZUCKERBERGER, E., TAL, A., AND SHLAFMAN, S. 2002. Polyhedral surface decomposition with applications. *Computers & Graphics 26*, 5, 733–743.