CCIT Report 412 January 2003

Hierarchical Mesh Decomposition

Sagi Katz and Ayellet Tal
Department of Electrical Engineering
Technion

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 not only computes the meaningful components but also avoids over-segmentation and jaggy boundaries between components. We also demonstrate the utility of the algorithm in two applications: control-skeleton extraction and metamorphosis.

Keywords: Mesh decomposition, mesh segmentation, deformation, metamorphosis

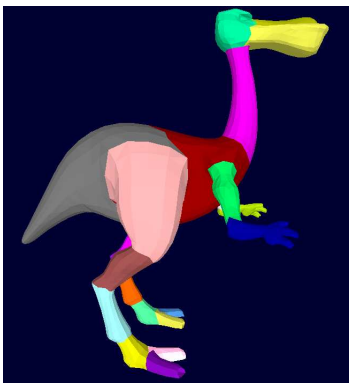


Figure 1: Decomposition of a dino-pet

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, and in particular tetrahelization, has been exhaustively investigated, e.g., [Chazelle 1984; Bajaj and Dey 1992; Ruppert and Seidel 1992; Aronov and Sharir 1994; 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 [Wu and Leahy 1993; Sharon et al. 2000; Shi and Malik 2000; Gdalyahu et al. 2001]. 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].

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 [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 and decomposition must be applied automatically to large databases [Zuckerberger et al. 2002]. In collision detection, decomposition facilitates the computation of bounding-volume hierarchies [Li et al. 2001]. We believe that the spectrum of applications which will benefit from mesh decomposition will grow even more in the future. Other potential applications include modification of objects, modeling by parts and texture mapping by parts.

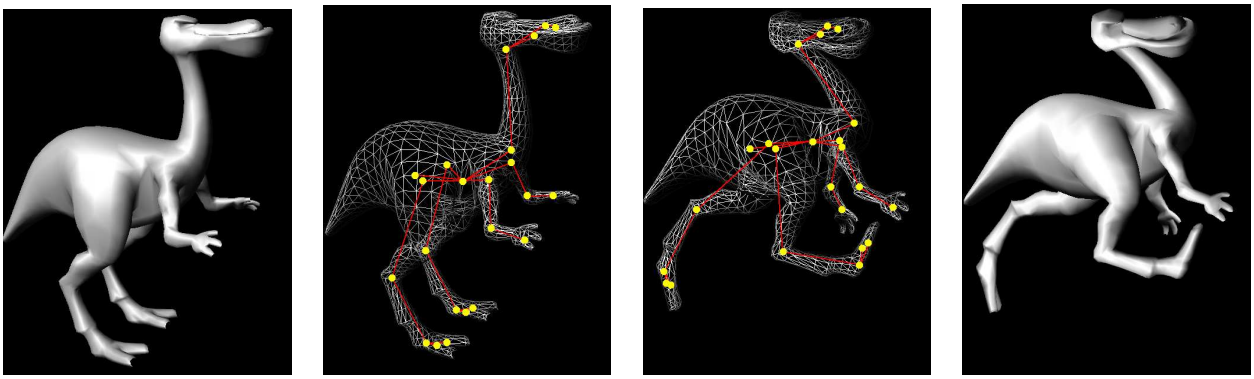
Several approaches have been discussed in the past for decomposing meshes. In [Chazelle and Palios 1992; Chazelle et al. 1997] a convex decomposition scheme is proposed, where a patch is called convex if it lies entirely on the boundary of its convex hull. Convex decompositions are important for applications such as collision detection. However, small concavities in the objects result with over-segmentation, which might pose a problem for other applications (i.e., metamorphosis). 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 [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. Moreover, the skeleton must be a tree, and thus loops and open meshes might pose a problem. 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 is related to that of [Shlafman et al. 2002], but it improves upon it in several aspects: our algorithm is hierarchical, handles arbitrary 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 boundaries between components are handled. Previously, the focus has 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, as illustrated in Figure 1



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

Figure 2: Deformation of a dino-pet

where the dino-pet is decomposed into its organs. (Each patch is colored differently.)

To demonstrate the usefulness of the algorithm, we present two applications. First, 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). It is also possible to apply the algorithm on large databases and use the skeletons for applications such as matching and retrieval. Second, we demonstrate the use of our algorithm for generating metamorphosis sequences. Decomposition is used as an aid in establishing a correspondence between meshes. A detailed correspondence for each component pair is then constructed. Since the boundaries between corresponding components are maintained, decomposition guarantees that features do not bleed into different features.

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 presents the applications. Finally, Section 6 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 arbitrary mesh (i.e., it need not be triangulated, closed or a 2-manifold).

Definition 2.1 ***k-way Decomposition:*** *$S_1, S_2, \ldots S_k$ is a 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:*** *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.*

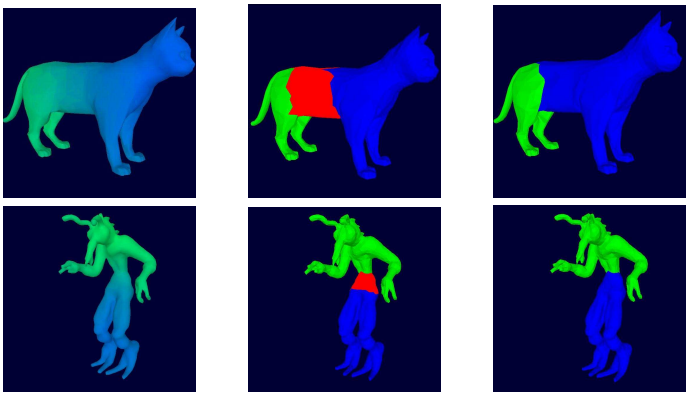
The algorithm works top-down. Each node in the hierarchy tree is associated with a mesh of a particular patch and the root is associated with the whole input object. At each node, the algorithm determines a suitable number of patches k, and computes a k-way decomposition of this node. If the input object 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 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 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), the greener the polygon, the higher its probability of belonging to the back (or upper) patch. Conversely, the bluer the polygon, the higher its 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.



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

Figure 3: Binary decomposition

Note that an alternative approach is find cuts in the whole model, as done in computer vision, rather than on constrained regions. One such cut is the *minimal cut*. The drawback of minimal cuts is that they tend to favor small sets of isolated nodes since the weight of the cut increases with the number of edges [Wu and Leahy 1993]. Our goal, however, is to find "meaningful components". Another option is to use normalized cuts [Shi and Malik 2000], which is an NP-complete problem. We experimented with approximation techniques adopted from image processing. 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 step-wise. Our approach solves these problems. One possible explanation is that since, unlike images, mesh objects are given wholly, obtaining the components is relatively easy. Furthermore, clearly finding good cuts in small regions is an easier problem than finding them in the whole object.

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 Distance function

The probability that a face belongs to a certain patch depends on its distance from other faces in this patch. The underlying assumption 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 edges 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 edges and $\eta = 1$ is used for concave edges.

The distance between f_i and f_j is then defined as follows. 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 the faces.

$$Dist(f_i, f_j) = \delta \cdot \frac{Geod(f_i, f_j)}{avg(Geod)} + (1-\delta) \cdot \frac{Ang\_Dist(\alpha_{ij})}{avg(Ang\_Dist)}. \quad (1)$$

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.

The distance definition is extended to non-adjacent faces in a Hausdorff manner, as described in Equation 2. The distance between faces which belong to different connected components is defined to be ∞.

$$Dist(f_i, f_j) = \min_{f_k \neq f_i, f_j} (Dist(f_i, f_k) + Dist(f_j, f_k)) \quad (2)$$

3.2 Assigning probabilities to faces

Assume that we are given two fuzzy patches A and B, each represented by one of its faces, REP_A and REP_B, respectively. 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:

$$\begin{aligned} P_B(f_i) &= \frac{a_{f_i}}{a_{f_i} + b_{f_i}} = \quad (3) \\ &= \frac{Dist(f_i, REP_A)}{Dist(f_i, REP_A) + Dist(f_i, REP_B)}. \end{aligned}$$

It can be easily verified that the function complies with the following desirable constraints.

1) $\forall a_{f_i} < b_{f_i}, \quad P_B(f_i) < 0.5$

2) $\forall a_{f_i} > b_{f_i}, \quad P_B(f_i) > 0.5$

3) $\forall a_{f_i} = b_{f_i}, \quad P_B(f_i) = 0.5$

4) $P_B(f_i) = 1 - P_A(f_i)$

The first and the second constraints suggest 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. The third constraint suggests that a face which is equally distant from A and B is as likely to belong to one as to the other. The last constraint guarantees symmetry. Finally, as expected, $P_B(REP_B) = 1$, $P_B(REP_A) = 0$ 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 [Duda and Hart 1973] clustering scheme [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

$$F = \sum_p \sum_f probability(f \in patch(p)) \cdot Dist(f, p). \quad (4)$$

During an initialization phase, a subset of k *representatives* V_k, is chosen. In the binary case, the initial pair of representatives REP_A and REP_B is chosen such that the distance between them is the largest possible. Then, the algorithm iterates on the following steps.

1. Compute the probabilities of faces to belong to each patch, as described in Equation 3.
2. Re-compute the set of representatives $V_{k'}$, minimizing the function in Equation 4.
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 for 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 face is 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.

$$A = \{f_i | P_B(f_i) < 0.5 - \epsilon\}$$
$$B = \{f_i | P_B(f_i) > 0.5 + \epsilon\}$$
$$C = \{f_i | 0.5 - \epsilon \leq P_B(f_i) \leq 0.5 + \epsilon\}$$

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 averages distances as described below. Empirically, this definition improves the results and expedites convergence.

$$a_{f_i} = avg_{f_j \in A}(Dist(f_i, f_j))$$
$$b_{f_i} = avg_{f_j \in B}(Dist(f_i, f_j))$$

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 graph $G = (V, E)$, 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. We also consider the set of 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) $V = V_{A'} \cup V_{B'}$

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

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

(4) $weight(Cut(V_{A'}, V_{B'})) = \sum_{u \in V_{A'}, v \in V_{B'}} \omega(u, v)$ 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 6).

$$V' = V_C \cup V_{CA} \cup V_{CB} \cup \{S, T\}$$
$$E' = E_C \cup \{(S, v), \forall v \in V_{CA}\} \cup \{(T, v), \forall v \in V_{CB}\} \cup$$
$$\cup \{e_{ij} | i \in V_C, j \in V_{CA}\} \cup \{e_{ij} | i \in V_C, j \in V_{CB}\}.$$

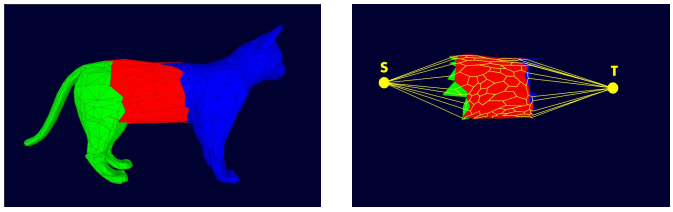


Figure 6: The flow network graph, where C is the red region

Next, the capacities of the edges 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 edges 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):

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

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

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.

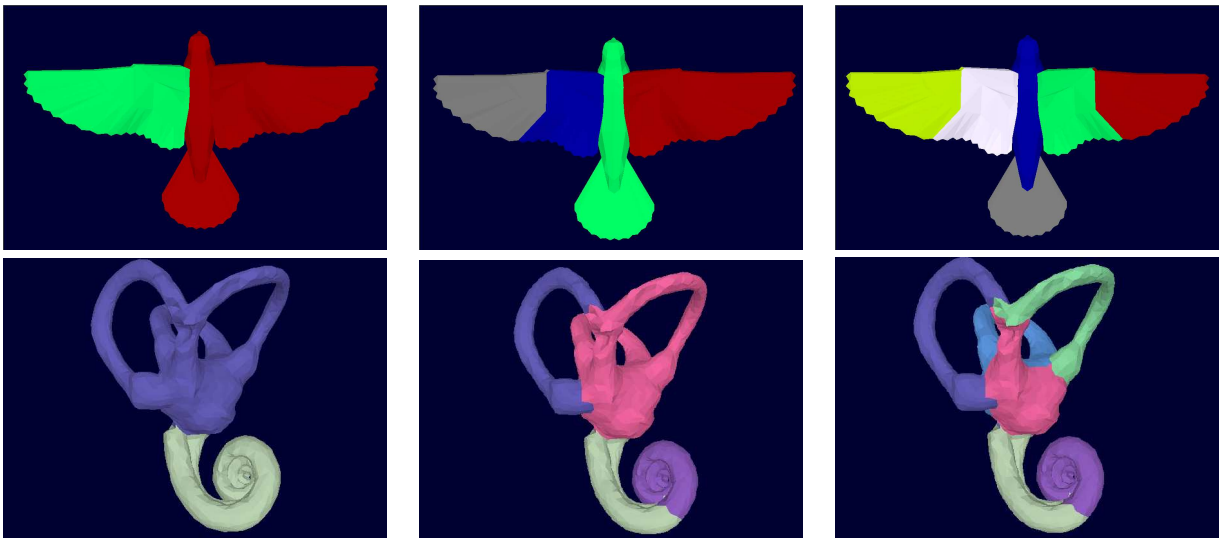
In the binary case, each patch is decomposed into two patches until a stopping condition is met. The initial representatives are chosen as the two faces which are farthest apart. In the $k-$way case, the representatives are chosen iteratively. The first representative is assigned to be the face having the minimum sum of distances from all other faces. 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:

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

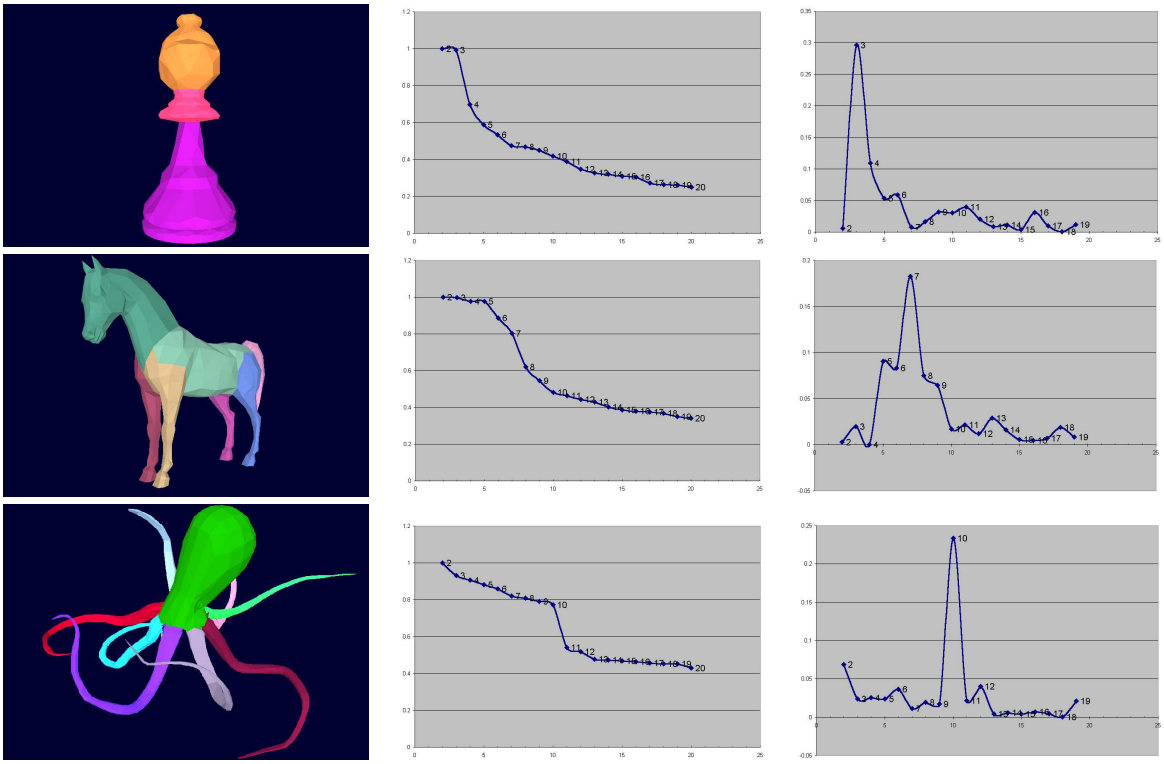
Obviously, this function decreases as we add more representatives. Empirical experiments show that after assigning representatives to all the major parts of an object, adding one more representative will cause a large decrease of G. This observation aids in determining the number of components k. We compute the first derivative of G and choose its maximum. See Figure 5.

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



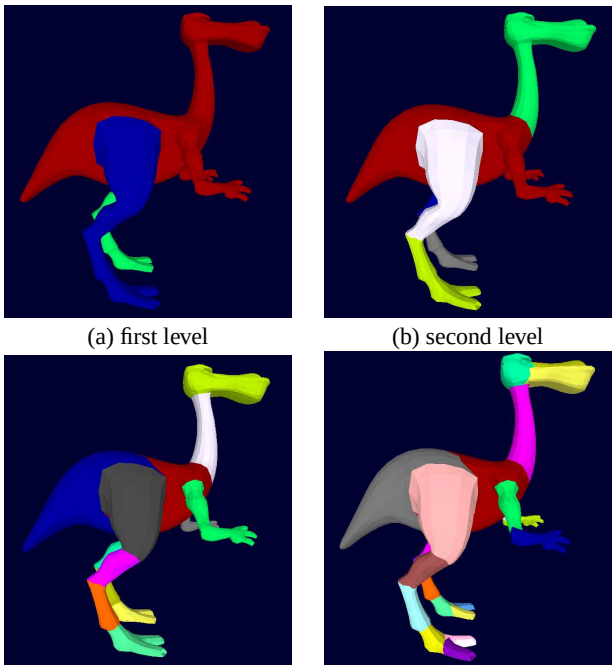
(a) first level (b) second level (c) third level

Figure 4: Hierarchical binary decompositions of a dove and an inner part of a human ear



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

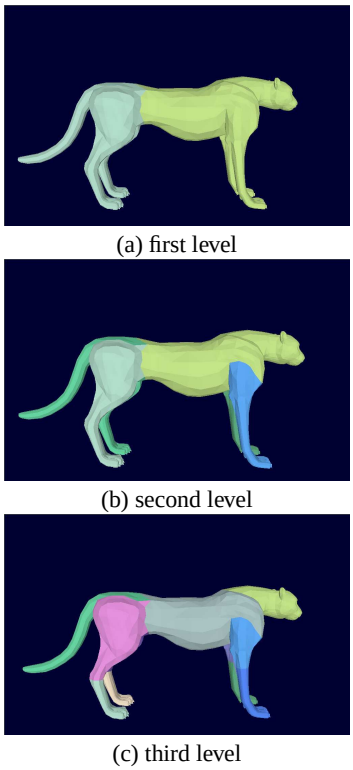
Figure 5: Determining optimal k



(a) first level (b) second level

(c)third level (d) fourth level

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

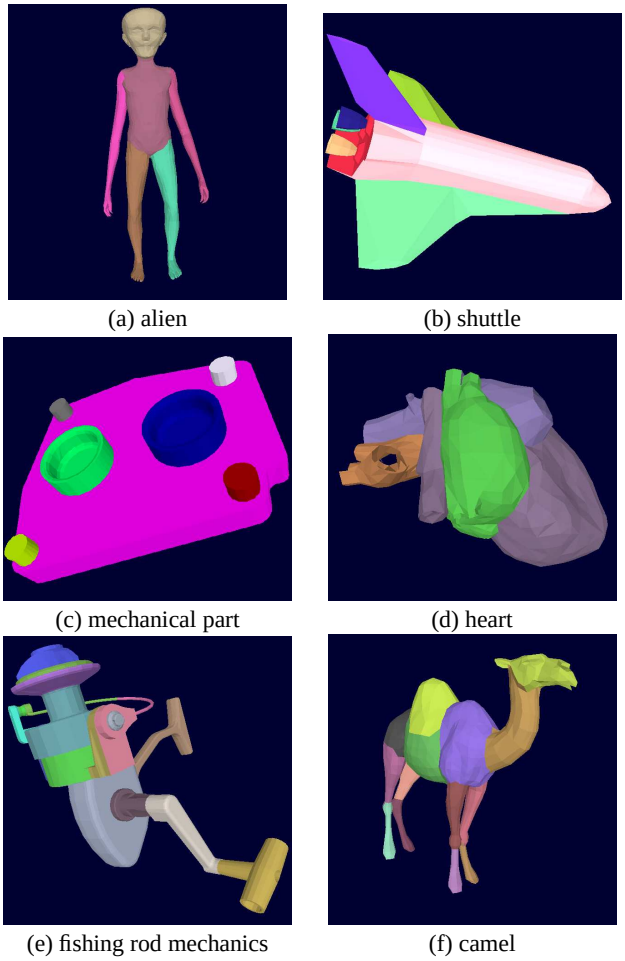


(a) first level

(b) second level

(c) third level

Figure 8: Hierarchical k-way decomposition of a cheetah



(a) alien (b) shuttle

(c) mechanical part (d) heart

(e) fishing rod mechanics (f) camel

Figure 9: Decompositions of various objects

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.

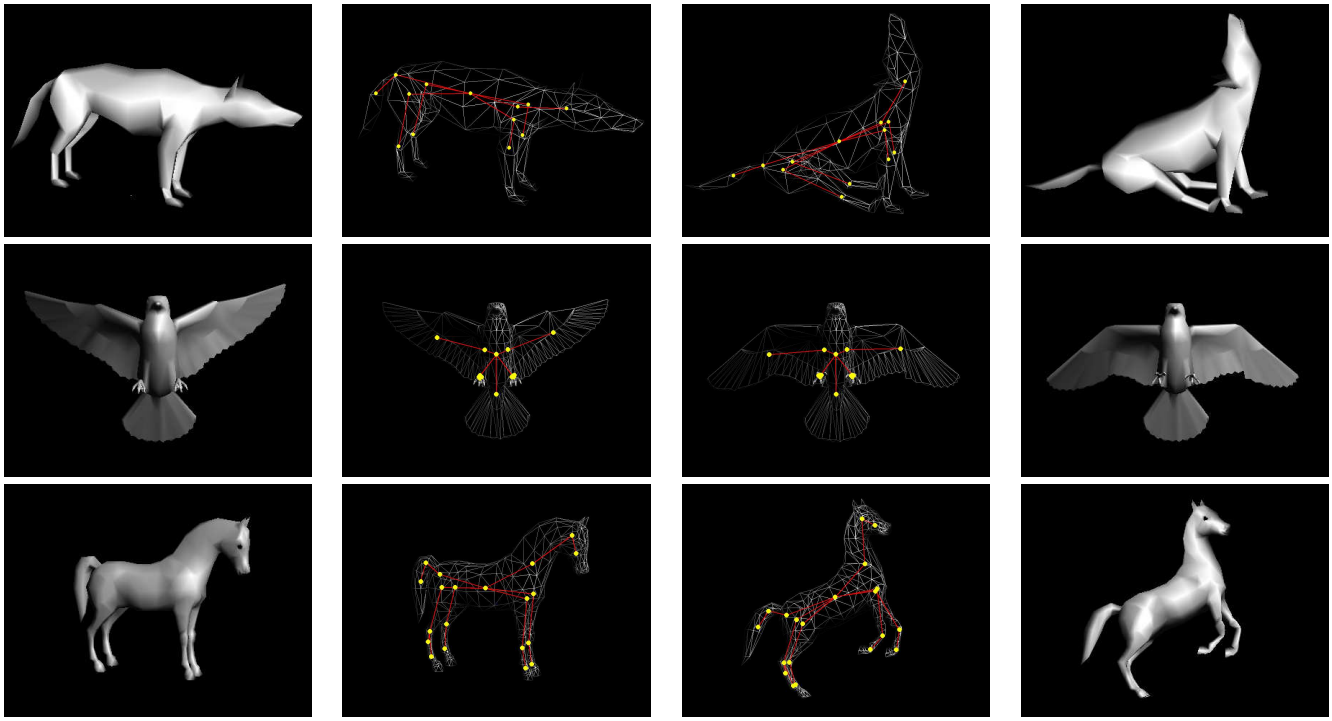
Figures 7–8 demonstrate a few levels of hierarchical k-way decompositions. Figure 9 shows decompositions of various levels.

5 Applications

We demonstrate our algorithm with two applications. First, we propose a new algorithm for control-skeleton extraction. Second, we illustrate the use of our algorithm for metamorphosis.

5.1 Control Skeleton extraction

Control skeletons are beneficial for various applications, including matching, metamorphosis and computer animation. As opposed



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

Figure 10: Deformations of various objects

to previous algorithms which are based on medial surface extraction [Gagvani et al. 1998; Teichmann and Teller 1998; Bloomenthal and Lim 1999; Wade and Parent 2002] and sometimes also on voxelization [Wade and Parent 2002], mesh decomposition gives rise to a novel control-skeleton extraction algorithm. The key idea is to calculate the joints directly from the hierarchical structure of mesh decomposition, without any user intervention. 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. The latter is demonstrated in Figure 10, where skeletons were used 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 it 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. At first, the root node is associated with the whole model. As the tree is traversed downward, 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 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 a non-border 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 n. d.]: $y_i = \sum_{j=0}^{J-1} (w_{ij} x_{ij} M_j)$. Here, 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.

5.2 Metamorphosis

A general approach for finding a correspondence between meshes is to construct their common embedding. The drawbacks of this approach are the difficulty in obtaining detailed correspondences as well as the requirement that the models be of genus zero. One way to overcome these shortcomings is to decompose the models prior to constructing common embeddings of their patch pairs [Gregory et al. 1999; Zockler et al. 2000; Shlafman et al. 2002]. When models have the same meaningful components (i.e., 4-legged animals), this approach gives rise to an automatic algorithm for finding a coarse correspondence. Since the boundaries between correspond-

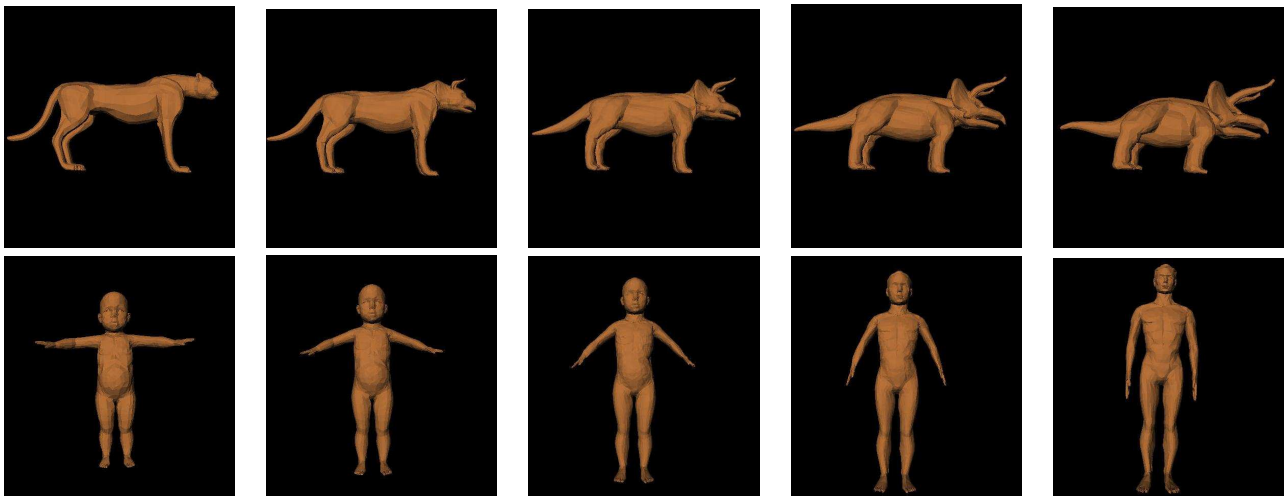


Figure 11: Metamorphosis sequences

ing components are maintained throughout the morph sequence, decomposition guarantees that features (i.e., organs) do not bleed into different features. We implemented this approach using our decomposition algorithm, and illustrate some results in Figure 11.

6 Conclusion

We have presented an algorithm for decomposing meshes into meaningful components. The algorithm is hierarchical and it avoids jaggy boundaries as well as over-segmentation. The main idea of the algorithm is to first find the meaningful components of the mesh and only then focus on generating the exact boundaries between components. The latter is done by formulating the problem as a constrained network flow problem. Furthermore, we successfully demonstrated the algorithm in two applications – control skeleton extraction and metamorphosis.

As future work, several enhancements can be easily 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. It is also possible to apply simplification prior to decomposition in the higher levels of the hierarchy, in order to accelerate the algorithm. We are also looking at two applications: compression and texture mapping. We believe that decomposition will have more applications in computer graphics in the future.

Acknowledgments

This research was supported in part by the Israeli Ministry of Science, Culture & Sports, Grant 01-01-01509. We would like to thank Dan Hadari, Itai David, Shymon Shlafman and Dan Aliaga for their great help.

References

ARONOV, B., AND SHARIR, M. 1994. Castles in the air revisited. *Disc. Comput. Geom 12*, 119–150.

BAJAJ, C., AND DEY, T. 1992. Convex decompositions of polyhedra and robustness. *SIAM J. Comput. 21*, 339–364.

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.

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.

CHAZELLE, B. 1984. Convex partitions of polyhedra: a lower bound and worst-case optimal algorithm. *SIAM J. Comput. 13*, 488–507.

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.

GDALYAHU, Y., WEINSHALL, D., AND WERMAN, M. 2001. Self-organization in vision: stochastic clustering for image segmentation, perceptual grouping, and image database organization. *IEEE Transactions on Pattern Analysis and Machine Intelligence 23*, 10, 1053–1074.

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.

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.

RUPPERT, J., AND SEIDEL, R. 1992. On the difficulty of triangulating three-dimensional non-convex polyhedra. *Disc. Comput. Geom 7*, 227–253.

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.

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. *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.