Visual Comput (2005)
DOI 10.1007/s00371-005-0344-9

ORIGINAL ARTICLE

Sagi Katz
George Leifman
Ayellet Tal

Mesh segmentation using feature point and core extraction*

Published online: 1 September 2005

S. Katz (✉) · G. Leifman · A. Tal
Department of Electrical Engineering
Technion – Israel Institute of Technology
{sagikatz@tx, gleifman@tx,
ayellet@ee}.technion.ac.il

* This work was partially supported by European FP6 NoE grant 506766 (AIM@SHAPE), by the Israeli Ministry of Science, grant 01-01-01509 and by the Ollendorff foundation.

Abstract Mesh segmentation has become a necessary ingredient in many applications in computer graphics. This paper proposes a novel hierarchical mesh segmentation algorithm, which is based on new methods for prominent feature point and core extraction. The algorithm has several benefits. First, it is invariant both to the pose of the model and to different proportions between the model's components. Second, it produces correct hierarchical segmentations of meshes, both in the coarse levels of the hierarchy and in the fine levels, where tiny segments are extracted. Finally, the boundaries between the segments go along the natural seams of the models.

Keywords Mesh segmentation · Mesh decomposition · Feature point extraction

1 Introduction

Cutting up a mesh into simpler sub-meshes benefits many algorithms in computer graphics, in areas as diverse as modeling [8], metamorphosis [11, 36], compression [14], simplification [10], 3D shape retrieval [37], collision detection [22], texture mapping [21], and skeleton extraction [15]. Some of these applications require segmentation into higher level meaningful components while others require segmentation into lower level disk-like patches [5, 10, 26, 34].

This paper presents an algorithm for segmenting a mesh into visually meaningful sub-meshes, following the minima rule [13], which states that the human vision defines part boundaries along negative minima of principal curvatures. For instance, in Fig. 1a, the sumo wrestler is segmented into his limbs, torso and head.

Several approaches to automatic mesh segmentation into meaningful components have been proposed in the past. Many methods use clustering techniques, such as region growing [4, 24], iterative clustering [31], spectral clustering [23], feature point-based clustering [33, 35], and fuzzy clustering with graph cuts [15]. Other techniques include skeleton-based methods [22] and snakes-based methods [20]. See [29] for a good survey on mesh segmentation.

The earlier algorithms tend to generate over-segmentations [4, 24], mostly due to local concavities. Some of latest algorithms overcome the fundamental problem of over segmentation and produce meaningful and nice-looking segmentations [15, 20, 22, 23]. However, some problems still remain. First, the algorithms are sensitive to the pose of the model. For instance, different segmentations will be produced for models of humans with folded arms and humans with straight arms. This is due to the vital role that curvature (or dihedral angles) plays in these algorithms. Second, clustering algorithms which are based on (geodesic) distances [15, 23] might generate different segmentations to similar objects having different proportions between their components. Third, the hierarchies, when they exist, are sometimes incorrect. For instance, at

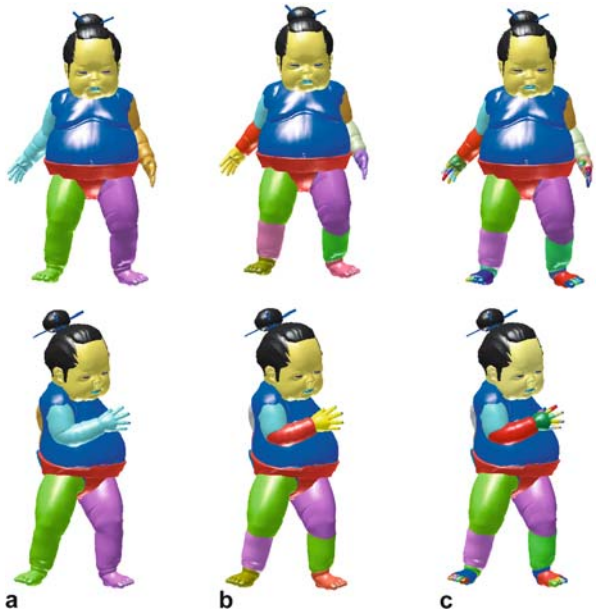

Fig. 1a–c. Pose-invariant segmentations: Two sumo wrestlers in different poses are segmented separately (top and bottom). The segmentations are similar in all levels of the hierarchy (the wrestler's hair, facial features, nails, and mawashi (belt) originally belong to different connected components). **a** First level **b** Third level **c** Sixth level

a certain hierarchical level, the legs are extracted but the arms are not. Fourth, the extraction of very small features, like parts of the fingers of a human, might still be difficult. Finally, in some of these algorithms the cuts between segments do not always go along the natural boundaries between components.

The algorithm proposed here produces hierarchical segmentations into meaningful components of orientable meshes, and overcomes the above problems. In particular, Fig. 1 illustrates pose invariance. The two sumo wrestlers, who are differently posed, were segmented separately, yet they have similar segmentations. Invariance to pose and proportions is important in modeling (where similar parts replace existing parts of a model), in metamorphosis (where similar parts are transformed from one to the other), and in 3D shape retrieval. In addition, the ability to construct the right hierarchy, to extract small features and to compute exact boundaries is vital to skeletonization.

Our approach is based on three key ideas. First, the mesh vertices are transformed into a pose-invariant representation, based on the theory of *multi-dimensional scaling (MDS)* [6, 19]. Second, this representation facilitates the robust extraction of *prominent feature points*. Intuitively, points on the tips of components, such as the tail, the legs and the head of an animal, are prominent feature points. Third, this representation also aids in the extraction of the core component of the mesh. The core component, together with the feature points, provide sufficient information for generating a segmentation.

The paper makes the following contributions:

1. A novel hierarchical pose-invariant mesh segmentation algorithm is presented.
2. A robust feature points extraction algorithm is described. This algorithm does not require any parameters. It is based on the observation that feature points can be characterized by local as well as global conditions, in terms of their geodesic distances. These feature points can be utilized in matching, metamorphosis and animation.
3. A novel core extraction algorithm is introduced. It is based on a new operation on meshes – *spherical mirroring*.

The rest of the paper is structured as follows. Section 2 gives an overview of the algorithm. Sections 3–6 discuss different parts of the algorithm. Section Sect. 7 presents some results. Section Sect. 8 concludes and discusses future directions.

2 Overview

Given an orientable mesh S, the goal is to hierarchically segment S into meaningful, face-wise disjoint, connected sub-meshes whose union gives S.

The algorithm proceeds from coarse to fine. Each node in the hierarchy tree is associated with a sub-mesh and the root is associated with the whole input mesh. For each node in the hierarchy tree, the algorithm consists of the following stages.

1. Mesh coarsening: Mesh coarsening is applied as a preprocessing step [9]. 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-invariant representation: Multi-dimensional scaling is used to transform the mesh S into a canonical mesh S_{MDS}, as shown in Fig. 2a. 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-invariant, because folded organs (i.e., arms) are "straightened" up by the transformation. It is important to note that since only the mesh vertices are used, S_{MDS} might be self-intersecting without having any effect on the subsequent steps of the algorithm.
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 Fig. 2b.
4. Core component extraction: The core component is extracted using a new *spherical mirroring* operation (Fig. 2c).

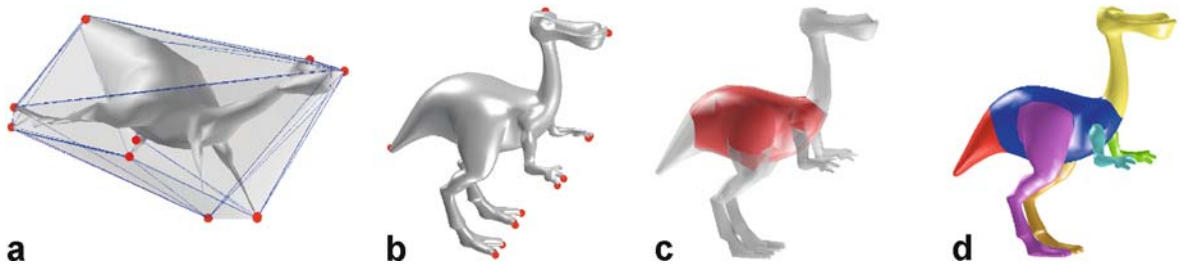

Fig. 2a–d. Algorithm outline **a** MDS transform and its convex hull **b** Feature points **c** Core **d** Segmentation & Cut refinement

5. Mesh segmentation: The algorithm computes the other segments, each representing at least one feature point, as illustrated in Fig. 2d.
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. For instance, a cube, a sphere, or a cone are not segmented, since they are each perceptually a single part.

The following sections elaborate on each of the key ideas of the algorithm, presented in Steps 2–6.

3 Pose-invariant representation

Multi-dimensional scaling (MDS) is used to transform the vertices of the mesh into a pose-invariant representation. MDS uncovers the geometric structure of a set of data items from (dis)similarity information about them, by representing dissimilarities as distances in an $m-$dimensional Euclidean space. The more dissimilar two items are, the larger the distance between them in this space. For more details and an historical perspective on MDS, see [6, 18, 19, 30].

In our solution, we define the dissimilarity between points on the mesh as the *geodesic distance* between them, i.e., the distance along the surface on which the points reside. The reason for using the geodesic MDS representation of a mesh is its invariance to pose. Bending has only a minor influence on the geodesic distances [7]. This property is demonstrated in Fig. 3, where the tail of the monkey is "unfolded" and the arms are "straightened".

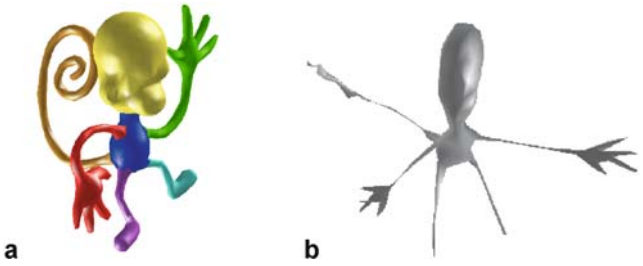

Fig. 3a,b. Transformation to MDS space **a** The monkey (segmented) **b** The monkey in MDS space

Given a set of vertices of a mesh $\{v_i | 1 \leq i \leq n\}$, the dissimilarities matrix is defined as $\{\delta_{ij} = GeodDist(v_i, v_j)\}$, $1 \leq i, j \leq n$. The geodesic distances can be efficiently computed by the *fast marching method* [28].

There are two major types of MDS methods: *metric MDS* and *non-metric MDS*. Metric MDS attempts to preserve the intervals and the ratios between the dissimilarities. Non-metric MDS preserves only the order of the dissimilarities, rather than the exact intervals and ratios.

Our empirical studies have discovered that metric MDS is too restrictive. Better results, both visually and in terms of the error function (described below) are achieved when a non-metric MDS is used. This is therefore our method of choice.

Stress functions are used to measure the degree of correspondence of the distances between vertices. Let δ_{ij} be the geodesic distance between v_i and v_j on S and d_{ij} be the Euclidean distance between their corresponding points in the MDS space. We use the following stress function:

$$F_s = \frac{\sum_{i<j}(f(\delta_{ij}) - d_{ij})^2}{\sum_{i<j} d_{ij}^2}. \quad (1)$$

The MDS algorithm attempts to optimize this stress function by minimizing the sum of distances between the optimally scaled data $f(\delta_{ij})$ and d_{ij}, where f is an optimal monotonic function of the dissimilarities.

This optimization proceeds as follows. In an initialization stage, the algorithm finds an initial configuration of the points in the $m-$dimensional MDS space. This can be done either by random sampling from a normal distri-

bution or by using the solution to the metric MDS. We use the second option, which is derived by an eigenvalue decomposition [3].

Then, the MDS algorithm iterates on the following steps, until the stress is sufficiently small. First, the Euclidean distances d_{ij} between the points in the MDS space are computed. Next, using the *pool adjacent violators* algorithm [2], the optimal monotonic function of the dissimilarities, f, is found, in order to obtain optimally scaled similarities. Finally, each vertex is re-mapped to a point in the MDS space, by minimizing Eq. 1. These points are the input to the next iteration.

The MDS algorithm can map the vertices to any m-dimensional Euclidean space. In our case, we aim at "straightening" the object's components, while keeping the general shape. Mapping the vertices onto the original dimension ($m = 3$), manages to achieve this goal. Conversely, mapping onto $\mathbb{R}^2$ causes loss of information, while in $\mathbb{R}^m$, $m > 3$, too many mapped vertices are positioned on the convex hull, thus changing the general shape.

4 Feature point detection

This section defines prominent feature points and describes an algorithm for finding them. The algorithm proposed requires neither prior knowledge regarding the number of feature points nor any user parameters.

Intuitively, feature points should reside on tips of prominent components of a given model. For instance, in Fig. 4, feature points are found on the tips of the horns, the ears, the mouth, and the beard. Moreover, feature points should be invariant to the pose of the model.

Feature points are useful in many applications, including metamorphosis [1], deformation transfer [32], mesh retrieval [37], cross-parameterization [16, 25, 27], texture mapping [17, 33], and segmentation [33, 35].

In this paper, feature points are used both for establishing the position of the segments and for determining whether the segmentation process should terminate.

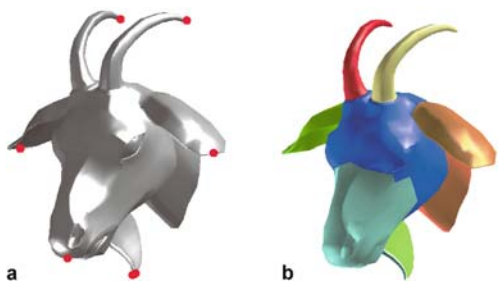

Fig. 4a,b. Feature points and segmentation **a** Feature points **b** Derived segmentation

In [33, 35], the local extrema of the average geodesic distance function [12] define the *critical points*. Filtering is necessary in the presence of many small features on the mesh, thus requiring a user-defined threshold. In our proposed method, user parameters are avoided by using a global condition, in addition to a local one.

To formally define the vertices on the tips, we require that these vertices satisfy the conditions described below.

$\forall v \in S$, let N_v be the set of neighboring vertices of vertex v. Let $GeodDist(v_i, v_j)$ be the geodesic distance between vertices v_i and v_j of mesh S. The local condition that a feature point should satisfy is that $\forall v_n \in N_v$

$$\sum_{v_i \in S} GeodDist(v, v_i) > \sum_{v_i \in S} GeodDist(v_n, v_i) \; . \qquad (2)$$

In other words, a vertex resides on a tip of a component if it is a local maximum of the sum of the geodesic distance functional.

Although this condition manages to extract a set of points that contains the prominent feature points, it may contain also many additional points that are not prominent feature points, due to small noise. Therefore, a global condition that automatically filters this set of points, is added. We note that on S_{MDS}, tip vertices tend to be extreme in some direction. This is so because MDS "unfolds" folded components of a model. This observation leads to the second condition: A feature point should reside on the convex-hull of S_{MDS}.

Definition 1. ***Feature point:*** *A mesh vertex is a feature point if it satisfies Eq. 2 and it resides on the convex-hull of S_{MDS}.*

This definition embeds in it an algorithm for computing the feature points of meshes. Given a mesh S and its transformed mesh S_{MDS}, the algorithm first computes the convex hull of S_{MDS} and then finds the vertices of the convex hull that satisfy Eq. 2. Finally, these vertices are mapped to their corresponding vertices on S.

5 Core extraction and mesh segmentation

Once the feature points are found, they are used to guide the segmentation. The mesh is segmented into its core component and its prominent components. Each prominent component is defined by one or more of the feature points.

The segmentation algorithm consists of three steps:

1. Spherical mirroring of S_{MDS},
2. Extraction of the core component of S,
3. Extraction of the other segments of S.

We elaborate on each step below.

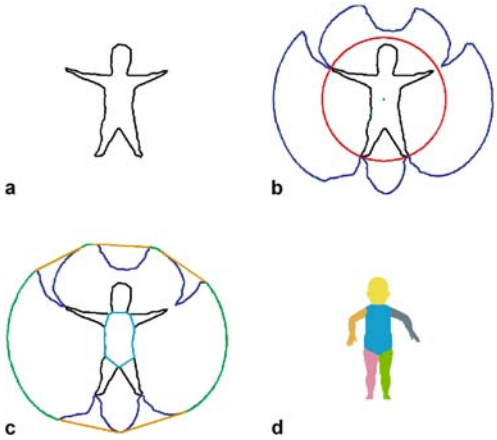

Fig. 5a–d. Core extraction steps: **a** A shape in MDS space (S_{MDS}). **b** Spherical mirroring. The red circle is the bounding sphere. The blue curve is the mirrored shape. **c** Convex hull of the mirrored shape in green and orange. The green curves are the parts of the mirrored image that lie on the convex hull and the cyan curve is the core. **d** The core, mapped to S, in cyan. Colors represent connected component created when subtracting the core

Spherical mirroring. As observed, prominent feature points on S_{MDS} tend to be extreme in some direction, while vertices of the core component tend to be closer to the center of S_{MDS} (Fig. 2a). Note that this is the case both when the features reside on convex regions on the original mesh S (the toes in Fig. 11c) and when they reside on concave regions on S (the eyes in Fig. 11a,b).

Spherical mirroring aims at reversing the situation, so that vertices of the core become external and can be easily extracted. To do it, a bounding sphere is computed and the vertices of mesh S_{MDS} are "mirrored", such that vertices of the core component reside on the convex hull of the mirrored vertices, and feature points become internal to the hull.

Let C and R (defined below) be the center and radius of this bounding sphere (which needs not be the minimal bounding sphere). Treating this sphere as a mirror, each vertex v of S_{MDS} is transformed to its image outside the sphere according to the following equation:

$$v_{mirror} = v + 2(R - ||v - C||)\frac{(v - C)}{||v - C||} .$$

This way, the image of the vertices of the core reside on the convex-hull, while vertices of the features become internal.

This procedure is illustrated in 2D in Fig. 5. The mirrored limbs and head are closer to the center while the mirrored core of the object lies on the convex hull.

It remains to explain how to compute C and R. There are many ways to compute a center. In our implementation, C is approximated in the first level of hierarchy by the center of mass of a subset of the vertices of S_{MDS}, which comply with a couple of properties. First, each vertex in this subset should be far from all the feature points. Second, the ratio between the maximal distance to a feature point and the minimal distance to a feature point is small. Though not a requirement, our experiments show that the center computed in this way is almost always internal to the core component of S_{MDS}. In the fine levels of the hierarchy, C is the center of the cut between the submesh being segmented and the core of the parent node.

The radius R of the bounding sphere is computed by measuring the maximum distance from the vertices of S_{MDS} to the center C: $R = \max_v ||v - C||$.

Figure 6 illustrates the results of spherical mirroring on several 3D meshes.

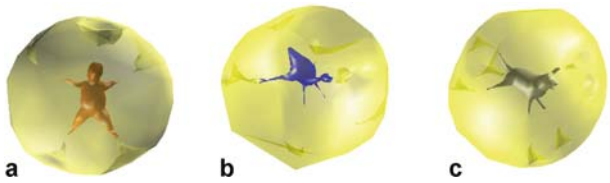

Fig. 6a–c. Spherical mirroring in 3D **a** (a) Sumo **b** Dino-pet **c** Dog

Core component extraction. The convex hull of the mirrored vertices is computed. The vertices that reside on the convex hull, along with the faces they define on S, are considered the initial core component. In most cases, the initial core separates the features (Fig. 5). When this is not the case, the core component is extended.

Core extension is an iterative process of adding the neighboring faces of the current core, until one of two conditions is satisfied. The first condition is that the current core separates all the feature points, thus creating disconnected components that define the segments, as demonstrated in Fig. 4 (the core is the blue segment).

The second condition occurs when the extension reduces the distance from the core to the closest feature point by more than a constant factor (0.5). This case happens when a few feature points are close to each other and thus cannot (and should not) be separated by the core, as illustrated in Fig. 2, where each leg contains a couple of feature points.

In this case, the algorithm backtracks to the state where the last feature point separation occurred and the core extension process stops. This backtracking has the effect of clustering feature points that correspond to small features. These features are not considered prominent at the current level of hierarchy. Obviously, they will be extracted at a finer level of the hierarchy. For example, in the first level of hierarchy, the arm of the sumo wrester contains a few feature points, one on each finger. Nevertheless, the backtracking results in the extraction of the whole arm as a single component. The fingers are extracted at a finer level of the hierarchy, as shown in Fig. 1.

Extraction of the other segments. Once the core component is found, the other segments of the mesh are extracted by "subtracting" the core component from the mesh (Fig. 5d). This subtraction, performed by breadth first search (BFS) that starts from the feature points, segments the mesh into connected components.

A connected component that contains at least one feature point, is considered a segment of the mesh. A connected component that does not contain any feature point, joins the core component.

6 Cut refinement

The previous stage resulted with a segmentation that might have coarse boundaries between segments. The current stage aims at smoothing these boundaries.

Cut refinement has been done by finding a minimum cut [15], by computing a constrained least cost path [8], or by using snakes [20]. Our algorithm uses minimum cuts, since it guarantees separation between the given components and it is simple to implement.

For each coarse boundary between segments (found during the previous step), a flow graph needs to be constructed. To do it, it is necessary to define a *search region* and a *capacity function* on the arcs of the flow graph. The dual graph of the search region, along with new source and target nodes, define the flow graph. Our minimum-cut implementation differs from that of [15] in both definitions.

The *search region* is defined to contain all the faces whose shortest distance to the boundary is smaller than a factor of the distance to the nearest feature point. This factor is a user-defined parameter.

The capacity function is defined as follows. Let v_i and v_j be vertices of the flow graph, whose dual faces on the mesh, f_i and f_j, are adjacent. Let θ_{ij} be the dihedral angle between f_i and f_j, $edge_{ij}$ be the length of the edge common to f_i and f_j and $angW_{ij} = (1-(1-|\cos(\theta ij)|)*convexityFac)^2$. The variable $convexityFac$ is small for concave dihedral angles and 1 for convex angles. Let AVG_{edge} be the average of $edge_{ij}$ and AVG_{angW} be the average of $angW_{ij}$. The capacity of the arc between v_i and v_j is defined as:

$$w_{ij} = \alpha\left(\frac{angW_{ij}}{AVG_{angW}}\right) + (1-\alpha)\frac{edge_{ij}}{AVG_{edge}},$$

where α is a user parameter.

Since the minimum cut tends to pass through arcs with small capacities, the boundary will pass through concave short edges.

Recall that the boundaries are found for the coarse-resolution model (Step 6 in Sect. 2). To compute the boundaries between segments of the input, fine-resolution, model (Step 7), the coarse cuts are mapped to the original model, search regions are defined, and the minimum cuts are calculated in a similar fashion.

7 Results

Figures 7 and 9 show some hierarchical segmentations. Figure 10 shows the first hierarchical level of additional objects. It can be seen that similar segmentations are computed for similar objects. For instance, the first hierarchical level of the four-legged animals consists of the animals' torso, head, tail and legs; humans are segmented into their torso, head, legs, and arms, etc. Moreover, very fine features, such as toes and facial features, are extracted in finer levels of hierarchy, as demonstrated in Fig. 11. These features are often difficult to extract with current segmentation algorithms.

Figure 8 shows a mesh that is difficult to segment, since it has many concave dihedral angles. Some algorithms that rely on curvatures might segment the mesh in each of its links. Since the algorithm is guided by prominent feature point and core extraction, it manages to segment the mesh into reasonable components. In particular, notice the blue core component that is extracted as a single component, despite the deep concavity it contains.

Figures 12 and 13 demonstrate invariances. Similar segmentations are produced for models in different poses or models having similar components with different proportions. The pose invariance is due to the use of MDS, while the invariance to different proportions is due to the feature point and core extraction. Even when the

Fig. 7. Hierarchical segmentation of a finch

Fig. 8. Segmentation of a scanned dinosaur (258 048 faces) that has many concavities

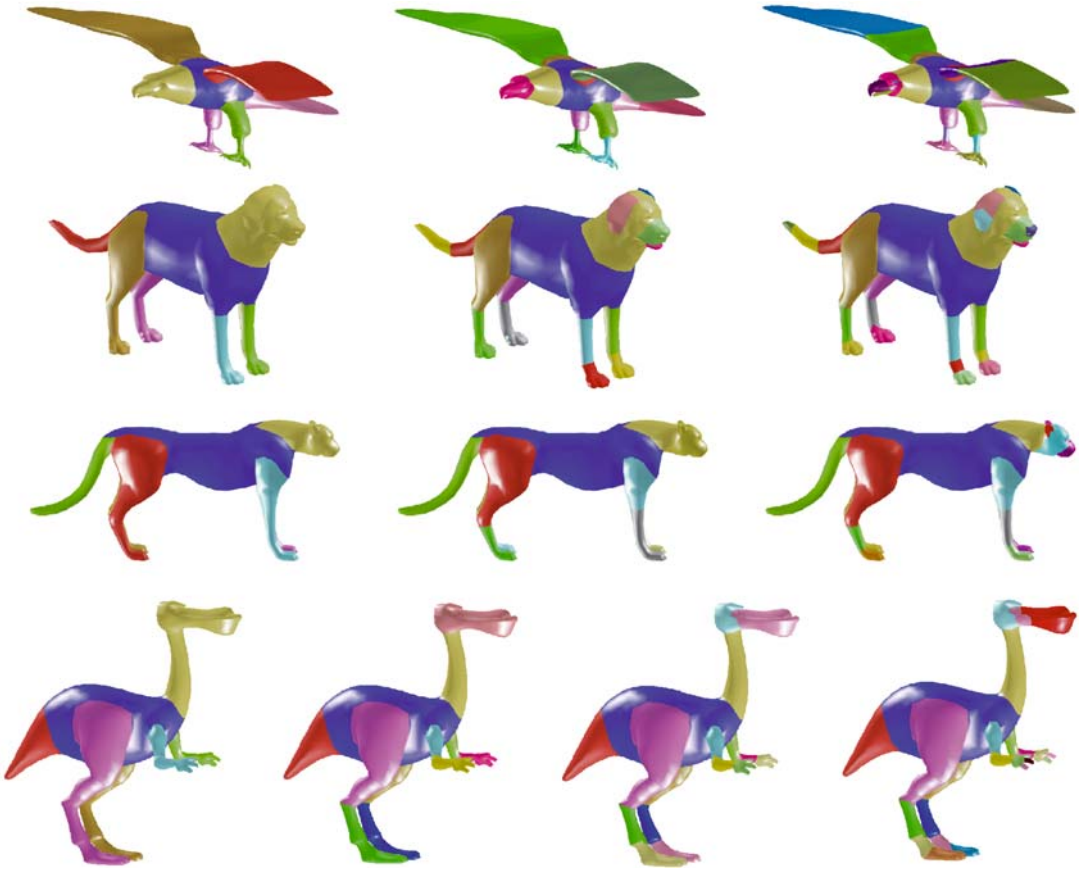

Fig. 9. Hierarchical segmentations of an eagle, a dog, a cheetah, and the dino-pet

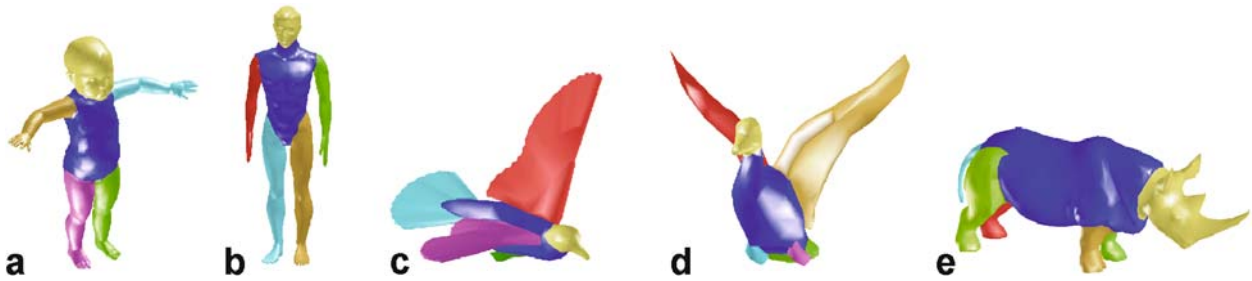

Fig. 10a–e. First hierarchical level segmentations of several meshes **a** Baby **b** Man **c** Dove **d** Duck **e** Rhino

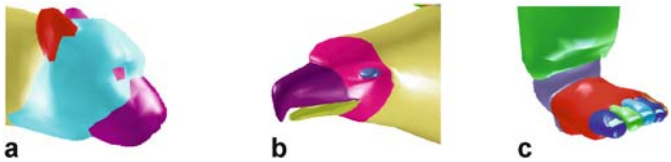

Fig. 11a–c. Zoom into the fine features **a** Cheetah head (Fig. 9) **b** Eagle head (Fig. 9) **c** Sumo foot (Fig. 1)

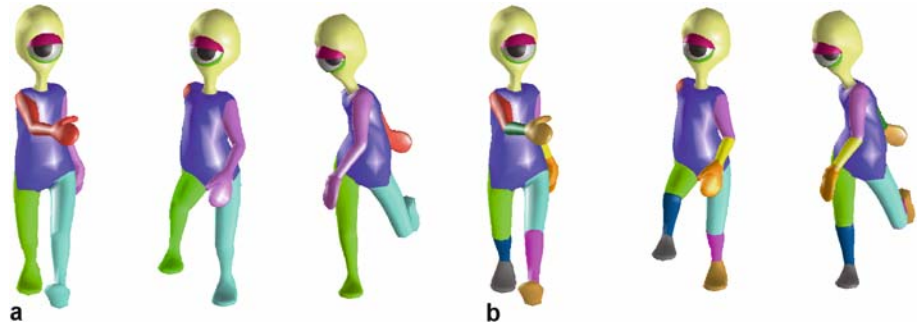

Fig. 12a,b. Pose-invariance: each model was segmented separately (the eye, eyeball, and eyelid originally belong to different connected components). **a** Pose-invariance – first hierarchical level **b** Pose-invariance – third hierarchical level

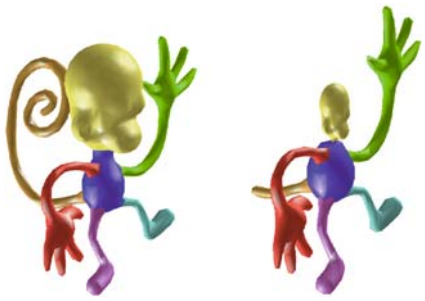

Fig. 13. Invariance to different proportions: each model was segmented separately

components grow or shrink, the core keeps separating them.

The only parameters that the user provided to segment these models, are related to the cut refinement stage and to the stopping condition. We view this as a desirable attribute of our algorithm, since almost no tuning is necessary. Moreover, the major parts of the algorithm – feature point extraction and meaningful component computation – are completely automatic.

The algorithm was implemented in MATLAB 7 and run on a 3.0 Ghz Pentium 4 laptop with 512 Mb RAM. The running times of sample objects are as follows: segmenting the dino-pet, consisting of 4000 faces, took 28 seconds, out of which MDS took 26 seconds. Segmenting the sumo wrestler, consisting of 26 792 faces, took 30 seconds, out of which MDS took 14 seconds and computing the flow network graph of the fine model and the minimum cut took 14 seconds (the coarse sumo has less faces than the coarse dino-pet, hence the reduction in MDS time). Segmenting the dinosaur, which consists of 258 048 faces, took 430 seconds, out of which constructing the flow graph of the fine model and the minimum cut took 370 seconds, MDS took 46 seconds and simplification took 12 seconds. Generally, on the coarse model, the time is dominated by the computation of the MDS. For very large models, the time is dominated by the computation of the minimum cut of the fine models. However, this time can be significantly shortened if the algorithm is implemented in C rather than MATLAB.

Let n be the number of faces in the original model, N be the number of faces in the coarse model (typically up to 1000 faces), m be the number of faces in the search region of the fine model, and M be the number of faces in the search region of the coarse model. Step 1: mesh coarsening, takes $O(n \log n)$. Step 2: MDS, is bounded by $O(N^2 \times no\_iterations)$, where $no\_iterations$ is usually 30-50. Step 3: feature point extraction, takes $O(N \log N)$ to compute the convex hull plus $O(N)$ per vertex on the convex hull to compute Eq. 2. Step 4: core extraction, costs $O(N \log N)$. Step 5: mesh segmentation, costs $O(N)$. Steps 6 and 7: cut refinement, are bounded by $O(M^2 \log M)$ and $O(m^2 \log m)$, respectively. Thus, the overall time complexity is $O(n \log n + N^2 \times no\_iterations + m^2 \log m + M^2 \log M) = O(N^2 \times no\_iterations + m^2 \log m)$.

We compare our results to those presented in [15], which also proposes an algorithm for hierarchical segmentation. Figure 14 compares the results of the first level of hierarchy. As can be seen, the current algorithm manages to extract the meaningful components already at this level, compared to the extraction of only three components in [15] (the other components are extracted in [15] in the next levels of hierarchy).

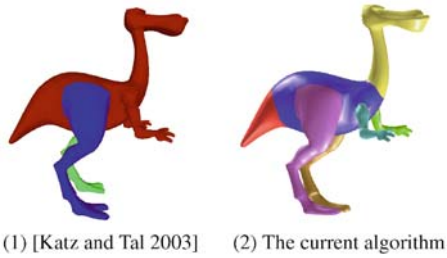

Fig. 14. Comparison – correctness of the first level of hierarchy

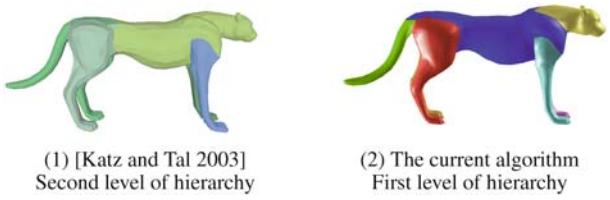

Fig. 15. Comparison – boundaries

Fig. 16. Segmentation of a genus-2 mesh

Figure 15 shows that the components are better separated by our algorithm – the tail does not join the back. This is due to the extraction of the core component performed by the current algorithm.

Figure 16 demonstrates a limitation of our algorithm. Though the correct segments were found, the boundary between the back leg and the tail is misplaced. This can be explained by fact that the tail forms a closed loop which our algorithm found to be a component. Another limitation occurs for CAD models. Since concavities, which are vital for these models, might be lost in the MDS representation, these models should not be transformed to MDS space prior to the segmentation.

8 Conclusion

We have presented an algorithm for hierarchical segmentation of meshes. The algorithm produces correct hierarchies and manages to extract even tiny features. It is invariant both to pose and to different proportions of components. Finally, the algorithm avoids jagged boundaries as well as over-segmentation.

The segmentation approach is based on three key ideas: a pose-invariant representation of meshes based on MDS, feature point extraction, and core component extraction.

To make this approach viable, a robust algorithm for prominent feature points extraction, as well as a novel algorithm for core component computation, are proposed. These algorithms can be utilized not only in mesh segmentation, as done here, but also in metamorphosis, matching, and texture mapping.

Our major future direction is designing an algorithm for compatible segmentation: given a couple of meshes, the goal is to segment them in a similar manner. As can be seen in the examples, our algorithm already segments the meshes very similarly. For instance, all four-legged animals are segmented into a core, a tail, a head, and four legs. All humans are segmented into their torso (core), head, two legs, and two arms. We believe that the current algorithm has the potential to be useful in compatible segmentation algorithms.

Acknowledgement We are grateful to Stanford Computer Graphics Laboratory for allowing us to use their models.

References

1. Alexa, M.: Merging polyhedral shapes with scattered features. Visual Comput. **16**, 26–37 (2000)
2. Barlow, R., Bartholomew, D., Bremner, J., Brunk, H.: Statistical Inference Under Order Restrictions. Wiley, New York (1972)
3. Borg, I., Groenen, P.: Modern Multi-dimensional Scaling: Theory and Applications. Springer, Berlin Heidelberg New York (1977)
4. Chazelle, B., Dobkin, D., Shourhura, N., Tal, A.: Strategies for polyhedral surface decomposition: an experimental study. Comput. Geom. Theory Appl. **7**(4–5), 327–342 (1997)
5. Cohen-Steiner, D., Alliez, P., Desbrun, M.: Variational shape approximation. ACM Trans. Graph. (SIGGRAPH) **23**(3), 905–914 (2004)
6. Cox, M., Cox, T.: Multidimensional Scaling. Chapman and Hall, London (1994)
7. Elad, A., Kimmel, R.: On bending invariant signatures for surfaces. IEEE Trans. Pattern Anal. Mach. Intell. **25**(10), 1285–1295 (2003)
8. Funkhouser, T., Kazhdan, M., Shilane, P., Min, P., Kiefer, W., Tal, A., Rusinkiewicz, S., Dobkin, D.: Modeling by example. ACM Trans. Graph. (SIGGRAPH) **23**(3), 652–663 (2004)
9. Garland, M., Heckbert, P.: Surface simplification using quadric error metrics. In: Proceedings of SIGGRAPH 1997, pp. 209–216 (1997)
10. Garland, M., Willmott, A., Heckbert, P.: Hierarchical face clustering on polygonal surfaces. In: Proceedings of ACM Symposium on Interactive 3D Graphics, pp. 49–58 (2001)
11. Gregory, A., State, A., Lin, M., Manocha, D., Livingston, M.: Interactive surface decomposition for polyhedral morphing. Visual Comput. **15**, 453–470 (1999)
12. Hilaga, M., Shinagawa, Y., Kohmura, T., Kunii, T.: Topology matching for fully automatic similarity estimation of 3D shapes. In: SIGGRAPH 2001, pp. 203–212 (2001)
13. Hoffman, D., Richards, W.: Parts of recognition. In: Pinker, S. (ed.) Visual Cognition, pp. 65–96. MIT Press, London (1985)
14. Karni, Z., Gotsman, C.: Spectral compression of mesh geometry. In: Proceedings of SIGGRAPH 2000, pp. 279–286 (2000)
15. Katz, S., Tal, A.: Hierarchical mesh decomposition using fuzzy clustering and

cuts. ACM Trans. Graph. (SIGGRAPH) **22**(3), 954–961 (2003)
16. Kraevoy, V., Sheffer, A.: Cross-parameterization and compatible remeshing of 3D models. ACM Trans. Graph. **23**(3), 861–869 (2004)
17. Kraevoy, V., Sheffer, A., Gotsman, C.: Matchmaker: constructing constrained texture maps. ACM Trans. Graph. **22**(3), 326–333 (2003)
18. Kruskal, J.: Multidimensional scaling by optimizing goodness of fit to a non-metric hypothesis. Psychometrika **29** (1964)
19. Kruskal, J., Wish, M.: Multidimensional Scaling. SAGE Publications, Thousand Oaks, CA (1978)
20. Lee, Y., Lee, S., Shamir, A., Cohen-Or, D., Seidel, H.P.: Intelligent mesh scissoring using 3D snakes. In: Pacific Conference on Computer Graphics and Applications, pp. 279–287 (2004)
21. Levy, B., Petitjean, S., Ray, N., Maillot, J.: Least squares conformal maps for automatic texture atlas generation. In: Proceedings of SIGGRAPH 2002, pp. 362–371 (2002)
22. Li, X., Toon, T., Tan, T., Huang, Z.: Decomposing polygon meshes for interactive applications. In: Proceedings of the 2001 symposium on Interactive 3D graphics, pp. 35–42 (2001)
23. Liu, R., Zhang, H.: Segmentation of 3D meshes through spectral clustering. In: Pacific Conference on Computer Graphics and Applications, pp. 298–305 (2004)
24. Mangan, A., Whitaker, R.: Partitioning 3D surface meshes using watershed segmentation. IEEE Trans. Visual. Comput. Graph. **5**(4), 308–321 (1999)
25. Praun, E., Sweldens, W., Schröder, P.: Consistent mesh parameterizations. In: SIGGRAPH '01, pp. 179–184. ACM Press, New York (2001)
26. Sander, P., Snyder, J., Gortler, S., Hoppe, H.: Texture mapping progressive meshes. In: SIGGRAPH '01, pp. 409–416. ACM Press, New York (2001)
27. Schreiner, J., Asirvatham, A., Praun, E., Hoppe, H.: Inter-surface mapping. ACM Trans. Graph. **23**(3), 870–877 (2004)
28. Sethian, J., Kimmel, R.: Computing geodesic paths on manifolds. Proc. of Natl. Acad. Sci. **95**(15), 8431–8435 (1998)
29. Shamir, A.: A formalization of boundary mesh segmentation. In: Proceedings of the 2nd International Symposium on 3DPVT (2004)
30. Shepard, R.: The analysis of proximities: multi-dimensional scaling with an unknown distance function. Psychometrika **27** (1962)
31. Shlafman, S., Tal, A., Katz, S.: Metamorphosis of polyhedral surfaces using decomposition. In: Proceedings of Eurographics, pp. 219–228 (2002)
32. Sumner, R., Popovic, J.: Deformation transfer for triangle meshes. ACM Trans. Graph. **23**(3), 399–405 (2004)
33. Zhang, E., Mischaikow, K., Turk, G.: Feature-based surface parameterization and texture mapping. ACM Trans. Graph. **24**(1), 1–27 (2005)
34. Zhou, K., Synder, J., Guo, B., Shum, H.Y.: Iso-charts: Stretch-driven mesh parameterization using spectral analysis. In: Eurographics/ACM SIGGRAPH Symposium on Geometry Processing, pp. 45–54 (2004)
35. Zhou, Y., Huang, Z.: Decomposing polygon meshes by means of critical points. In: MMM, pp. 187–195 (2004)
36. Zockler, M., Stalling, D., Hege, H.C.: Fast and intuitive generation of geometric shape transitions. Visual Comput. **16**(5), 241–253 (2000)
37. Zuckerberger, E., Tal, A., Shlafman, S.: Polyhedral surface decomposition with applications. Comput. Graph. **26**(5), 733–743 (2002)

SAGI KATZ is currently a Ph.D. student at the department of Electrical Engineering, Technion. He received a B.Sc. (cum laude) and M.Sc (cum laude) degrees in Electrical Engineering from the Technion in 2001 and 2003.

GEORGE LEIFMAN received the B.Sc. degree (Summa cum Laude) in Computer Engineering from the
Technion in 2001, and the M.Sc. degree in Electrical Engineering from the Technion in 2003. Currently he is a research assistant at the department of Electrical Engineering, Technion.

AYELLET TAL is a faculty member at the department of Electrical Engineering, Technion. She holds a Ph.D. in Computer Science from Princeton University, an M.Sc. degree (Summa cum Laude) in Computer Science from Tel-Aviv University and a B.Sc degree (Summa cum Laude) in Mathematics and Computer Science from Tel-Aviv University.