Consistent mesh partitioning and skeletonisation using the shape diameter function

Article *in* The Visual Computer · April 2008

DOI: 10.1007/s00371-007-0197-5 · Source: DBLP

3 authors, including:

Ariel Shamir
Interdisciplinary Center Herzliya
106 PUBLICATIONS **8,180** CITATIONS

SEE PROFILE

Daniel Cohen-Or
Tel Aviv University
273 PUBLICATIONS **23,888** CITATIONS

SEE PROFILE

Visual Comput (2008) 24: 249–259
DOI 10.1007/s00371-007-0197-5

ORIGINAL ARTICLE

Lior Shapira
Ariel Shamir
Daniel Cohen-Or

Consistent mesh partitioning and skeletonisation using the shape diameter function

Published online: 8 January 2008

L. Shapira (✉) · D. Cohen-Or
Tel-Aviv University
Tel-Aviv, Israel
liors@post.tau.ac.il, dcor@tau.ac.il

A. Shamir
The Interdisciplinary Center
Herzeliya, Israel
arik@idc.ac.il

Abstract Mesh partitioning and skeletonisation are fundamental for many computer graphics and animation techniques. Because of the close link between an object's skeleton and its boundary, these two problems are in many cases complementary. Any partitioning of the object can assist in the creation of a skeleton and any segmentation of the skeleton can infer a partitioning of the object. In this paper, we consider these two problems on a wide variety of meshes, and strive to construct partitioning and skeletons which remain consistent across a family of objects, not a single one. Such families can consist of either a single object in multiple poses and resolutions, or multiple objects which have a general common shape. To achieve consistency, we base our algorithms on a volume-based shape-function called the shape-diameter-function (SDF), which remains largely oblivious to pose changes of the same object and maintains similar values in analogue parts of different objects. The SDF is a scalar function defined on the mesh surface; however, it expresses a measure of the diameter of the object's volume in the neighborhood of each point on the surface. Using the SDF we are able to process and manipulate families of objects which contain similarities using a simple and consistent algorithm: consistently partitioning and creating skeletons among multiple meshes.

Keywords Mesh decomposition · Skeleton extraction · Geometry processing

1 Introduction

Partitioning of 3D meshes and extracting a skeleton-like structure for such meshes are two basic and highly important algorithms in computer graphics. Most previous partitioning and skeletonisation techniques concentrated on one object and rely mostly on surface attributes of the object's boundary. These include curvature, normal directions and average geodesic distances (AGD is defined as the integral over the whole surface of the geodesic distance of a point on the surface to all other points). Such surface attributes often depend on the pose of the object and on its topology; therefore, it may change, for instance, if the pose of the object changes. This, in turn, often leads to different results of partitioning or skeletonisation of similar objects. In contrast, one of the attributes that remains invariant under pose and often even topology changes is the volume of the object.

Still, the dominating representation of 3D objects in graphics is a 2D surface mesh embedded in 3D, defining its boundary. Therefore, in our approach we employ the shape-diameter function (SDF, Fig. 1a), that provides a link between the object's volume and its boundary, mapping volumetric information onto the surface boundary mesh.

The geometric intuition behind the SDF is to create a kind of low pass filtering to a shape-diameter measure by examining the diameter in the neighborhood of

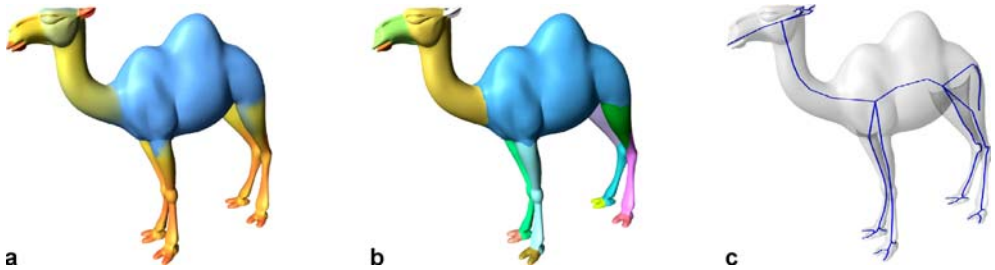

Fig. 1a–c. The local shape diameter function (SDF) provides a link from the surface of the mesh to its volume. **a** A color map of the function values on the mesh from *red* (narrow diameter) to *blue* (wide diameter). The SDF is related to the medial axis transform and can be used to find a consistent hierarchical partitioning of objects (**b**) and to create simple robust skeletons (**c**)

each point on the surface. This measure relates to the medial axis transform (MAT) [4]. However, computing the medial axis and the MAT of a surface mesh can be an expensive process, and the medial axis itself is difficult to handle [1, 9]. In general, the medial axis of a 3D object is a collection of non-manifold sheets and does not resemble a one-dimensional curve skeleton, which is often sought after in graphics and animation. Still, the connection from the boundary to the axis using local shape-radius in the MAT is extremely informative for partitioning, skeletonisation and manipulation. Therefore, in the SDF we replace the local shape-radius by a measure of the local *shape-diameter*, which is simpler and faster to compute.

In this paper we concentrate mostly on families of articulated objects, although other types of objects can benefit from SDF analysis as well (Fig. 16). This can include human and animals figures, or more generally, objects representing characters used in animation and graphics. We present the algorithms to partition such objects to parts consistently and to extract their simple one-dimensional curve skeletons (Fig. 1b,c). These algorithms are *consistent* as they are largely insensitive to pose changes of the same object, and furthermore, they present similar results in analogue parts of different objects. Moreover, as the SDF is a simple scalar field, both partitioning and extracting a skeleton are implemented as very simple and fast algorithms, allowing greater interactivity and ease of use.

2 Related work

Many previous geometry processing algorithms are based on attributes obtained from surface 3D models (polygonal meshes). Particularly, the use of surface geometric attributes is still the most prevalent in works targeting partitioning of meshes to parts (See [27] for a survey on partitioning techniques). These attributes include curvature and geodesic distances [13, 16, 21, 25, 26, 37], dihedral angles [28], planarity and normal direction [3, 5], slippage [10] etc. Such attributes are sensitive to local surface features and to pose changes. Therefore, they are not suitable for linking between the same object in different poses or for finding analogies between different objects consistently. Topology-based approaches as in [24], spectral analysis as in [20], and global attributes such as average geodesic distance (AGD) [11] and Reeb-graphs [2] can support some level of consistency over pose differences of the same shape. Nevertheless, they are vulnerable to topological and connectivity changes and do not distinguish well between shape differences. Geodesic distances are sensitive to shape changes even in remote parts, and the topology may be altered considerably if the geometry is changed slightly by connecting or disconnecting parts.

Our approach focuses on the consistency over a family of objects; it is insensitive to topological changes and simple in terms of computation. Furthermore, we also demonstrate consistency across families of *different* objects. We base our work on the definition of the shape diameter function (SDF), exploring the connection from the mesh to the volume of the object instead of on surface attributes. Other methods which use this connection include [3] in which geometric primitives are fitted to the mesh to partition the object, and [23] in which the partitioning is based on blowing a spherical bubble at each vertex and studying how the intersection of that bubble with the surface evolves. Recently, in [14] a meaningful part decomposition is achieved using an approximate convex decomposition. Each model is partitioned hierarchically, attempting to preserve part convexity and compactness. A later step finds the best part correspondence between a group of models.

Several works explore the strong connection between part-partitioning and skeletonizing [19, 22, 30, 31, 34]. In [30] feature points are extracted from a mesh, and are used to calculate an invariant mapping function, revealing important parts in the mesh. Geometrical and topological analysis using Reeb graphs enable the authors to extract a visually meaningful skeleton. This skeleton was employed in a follow-up work [31] to calculate a hierarchical segmentation. In [13] object part decomposition facilitates the definition of a skeleton, which in turn is used for deformations and animation. In a later work [12] multi-dimensional scaling is used to create a pose-oblivious representation of the mesh, which is used to find fea-

ture points and perform a hierarchical decomposition. In [19] an iterative procedure is used, simultaneously performing an approximate convex decomposition of the object [18] and extracting a skeleton using the principal axis of each part. The procedure is repeated until the skeleton is of satisfactory quality. In [32] skeletal curves are constructed from scattered points on the surface of an object. The procedure involves expensive construction of a nearest-neighbors and geodesic distances graphs, and produces a tree-like structure which approximates the object's skeleton.

The two leading approaches for defining skeletons are based either on the MAT or voxelization. The MAT-based approach such as [8] gives a high level of accuracy, yet the MAT is a complex non-manifold structure, which is difficult to extract and hard to manipulate and utilize. Voxelization techniques such as in [29, 36] create a discrete approximation. These techniques are prone to discretization errors and rely heavily on the grid size. A recent method for skeleton extraction combined several approaches by first extracting feature points of the skeleton, mapping them to the surface and then linking the surface partitioning to create the domain connected graph of an object [34]. Nevertheless, such techniques are cubic in nature while the SDF maps a description of the volume to the surface, therefore presenting 2D complexity.

3 The shape diameter function

An effective link from the object's volume to its surface boundary is provided by the medial axis transform (MAT) [4] of the object. The MAT represents an object by the distance of each point to the medial axis. The distance from a point on the boundary to the medial axis is the radius of the maximal ball, whose center lies on the medial axis, touches the boundary at the point, and is completely contained in the object. This ball is called the *medial ball*, and its radius can be seen as a form of local *shape radius* connecting the boundary to the medial axis. Nevertheless, the definition and extraction of the medial axis or even of discrete approximations using skeletons are complex and often error prone. Instead, we define a measure, which is much simpler to compute, that connects the local shape's volume to the surface by measuring the *shape diameter*.

Let M be a manifold mesh surface defining a volumetric object. We define a scalar function on the surface $f_v : M \to \mathbb{R}$ which we call the *shape diameter function (SDF)* as the *neighborhood diameter* of the object at each point $p \in M$. On a smooth surface the exact diameter can be defined by the distance to the antipodal surface point using the inward-normal direction. On a piecewise linear mesh, it is difficult to define the exact antipodal point. Moreover, we want to express the neighborhood shape diameter, which is different than the exact distance to the antipodal point.

Given a point on the surface mesh, we use a cone centered around its inward-normal direction (the opposite direction of its normal), and send several rays inside this cone to the other side of the mesh (Fig. 3). For each such ray we check the normal at the point of intersection, and ignore intersections where the normal at the intersection points in the same direction as the point-of-origin (the same direction is defined as an angle difference less than 90°). This is done to remove false intersections with the 'outside' of the mesh. The SDF at a point is defined as the weighted average of all rays lengths which fall within one standard deviation from the median of all lengths. The weights used are the inverse of the angle between the ray to the center of the cone. This is because rays with larger angles are more frequent, therefore having smaller weights.

This definition of the SDF is invariant to rigid body transformations of the whole mesh. Furthermore, the SDF is oblivious to any deformation that does not alter the volumetric shape locally. This includes articulated character deformations, skeleton-based movements or piecewise-rigid transformations. In essence, the SDF remains largely *pose oblivious* (Fig. 2).

Using a small cone angle will not create a good discrimination between different object parts and would be too sensitive to local features of the mesh. On the other hand, using a large opening angle, close to 180°, exposes the SDF measure to noise and errors as some of the rays find their way to unrelated parts of the model. We have tested the effect of various parameter settings on the consistency of the SDF on over 1000 meshes: using different opening angles and various number of rays. In practice, we set the default values to an opening angle of 120° and send 30 rays per point, relieving the user from any parameter tuning. Although we combine the results from

Fig. 2. The shape diameter-function remains largely consistent through pose differences of the same object

Fig. 3. Examples of the cone of rays sent to the opposite side of the mesh (*left*). *Green rays* are within accepted range of the median while *red rays* are rejected outliers. The progression of images on the *right* shows the robustness gained by increasing the cone angle: from a single ray to the antipodal point (*top right*), up to sampling of a 120° cone, which is used in practice (*bottom right*)

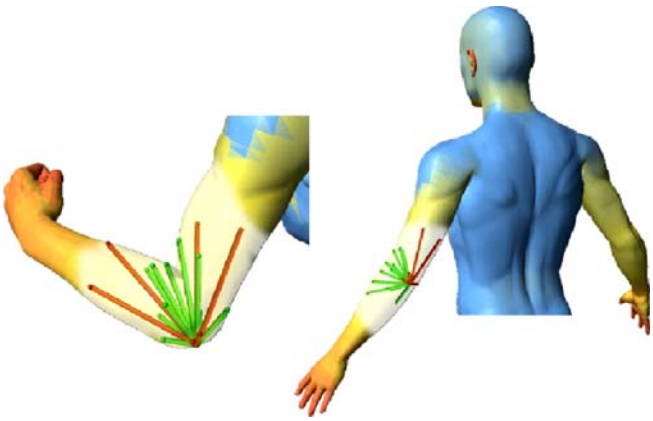

Fig. 4. In some cases, different poses of the object may imply different SDF values in the same position on the mesh. By using robust statistics and smoothing, we can overcome this problem

multiple rays, there are still positions on the mesh where the measure can change after pose changes (Fig. 4). To overcome this, we perform smoothing based on a small number of bilateral filtering iterations of the SDF values on small mesh neighborhoods around each vertex.

$$f_v^0(x) = f_v(x)$$
$$f_v^{i+1}(x) = \frac{f_v^i(x) + \sum_y c\big(\big|f_v^i(x) - f_v(y)\big|\big) w_x(y) f_v(y)}{1 + \sum_y c\big(\big|f_v^i(x) - f_v(y)\big|\big) w_x(y)},$$

where $f_v^{i+1}(x)$ is the SDF value of point x in iteration $i+1$. It is calculated as the average sum of all y neighbors of x, weighted using a geodesic distance-based Gaussian function w, and clumped in the function domain by c ($c(x) = 0$ for $x > t$).

As an example, using the five different poses of the same human model of Fig. 2, we measure the SDF value at the center of each face. We then calculate the mean of the SDF value for each face over the five poses. Next, we measure the difference between the SDF value of each face and its mean in all five poses, and average it over all mesh faces. Figure 5 shows a plot of this divergence and its standard deviation. The SDF values in this examples are normalized to [0, 1] and so the differences are in the order of 0.1% while the standard deviation are extremely low.

The basic operation in the SDF calculation is a ray-mesh intersection, well known in ray-tracing. This oper-

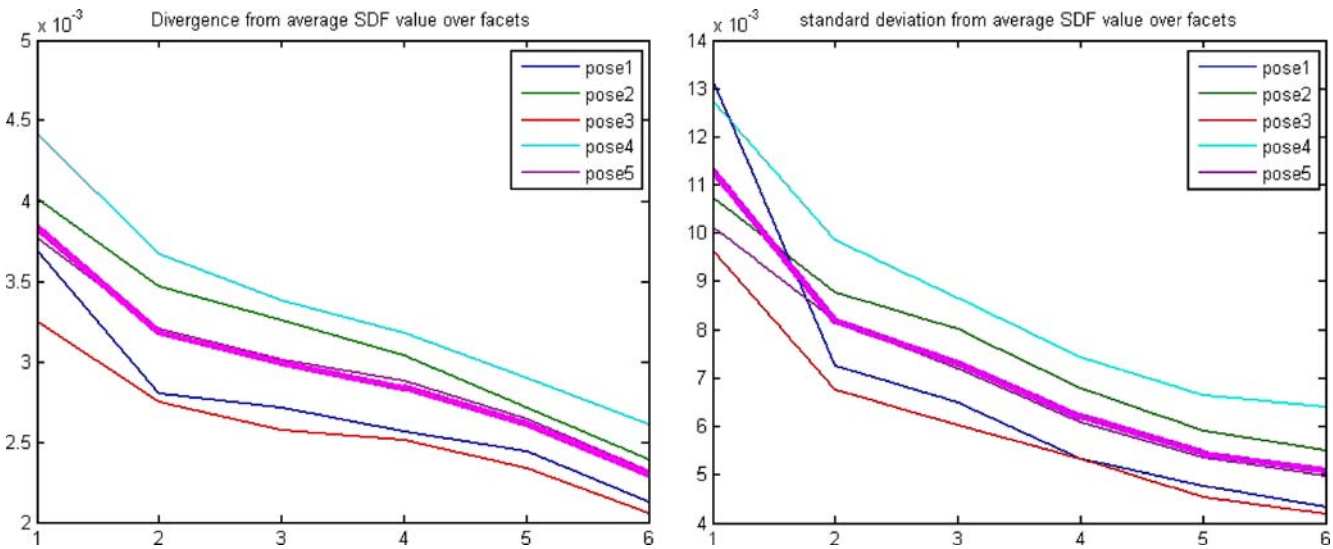

Fig. 5. The effect of enlarging the cone angle and adding more rays on the consistency of the SDF over different poses

Table 1. Timing results for computing the shape diameter function using 30 rays for each point. Testing done on Pentium 4 1.8 GHz with 2 GB RAM and Nvidia Ge-Force FX-5600

Mesh	Vertices	Faces	Time (sec.)
Low-res man	2500	4000	1.5
Dino-pet	3300	6500	1.7
Frog1	6700	13000	4.68
Horse	20000	40000	18
Hi-res man	25000	45000	26.8
Feline	50000	100000	55
Armadilo	175000	350000	110

ation can be accelerated using a spatial search structure, in our case an octree built around the mesh. In general, the intersection calculations along with construction of the octree do not take more than a few seconds even on large meshes. Consequently, computing the SDF even on large meshes takes on the order of seconds (see Table 1). The outliers removal method removes rays with no intersection at all (infinite length rays), making the SDF calculation robust to cracks and holes in non-watertight meshes.

4 Consistent partitioning

Many mesh partitioning algorithms use surface attributes such as curvature and dihedral angles. Using such attributes the results of partitioning may change if the pose of the object changes, and may be disparate on rather similar objects. One option to preserve consistency is to transform the shape into a 'canonical' position (e.g., using multi-dimensional scaling [6]) and then apply the partitioning [12]. Nevertheless such transformation are often unstable and may change drastically if the mesh changes its connectivity or topology even slightly.

Our approach is to use a shape property that provides consistency over pose changes – the SDF. The SDF also distinguishes natural object parts. One can observe (Fig. 6) that points belonging to a specific part of an object, e.g., the fingers, hands, legs, torso, and head in humans and animals, have similar SDF values which are characteristic of the part itself. Lastly, the SDF is similar across similar objects providing consistency on matching parts of similar shapes.

At a pre-processing stage, we calculate the SDF value of the middle of each facet. To maintain compatibility over different meshes, which may have different scales and resolutions, we normalize and smooth the function. We also perform the partitioning in log-space, enhancing the importance of delicate parts (which tend to have low characteristic SDF values). The normalized SDF value nsdf of facet f is defined:

$$\mathrm{nsdf}(f) = \log\left(\frac{\mathrm{sdf}(f) - \min(\mathrm{sdf})}{\max(\mathrm{sdf}) - \min(\mathrm{sdf})} * \alpha + 1\right) / \log(\alpha + 1),$$

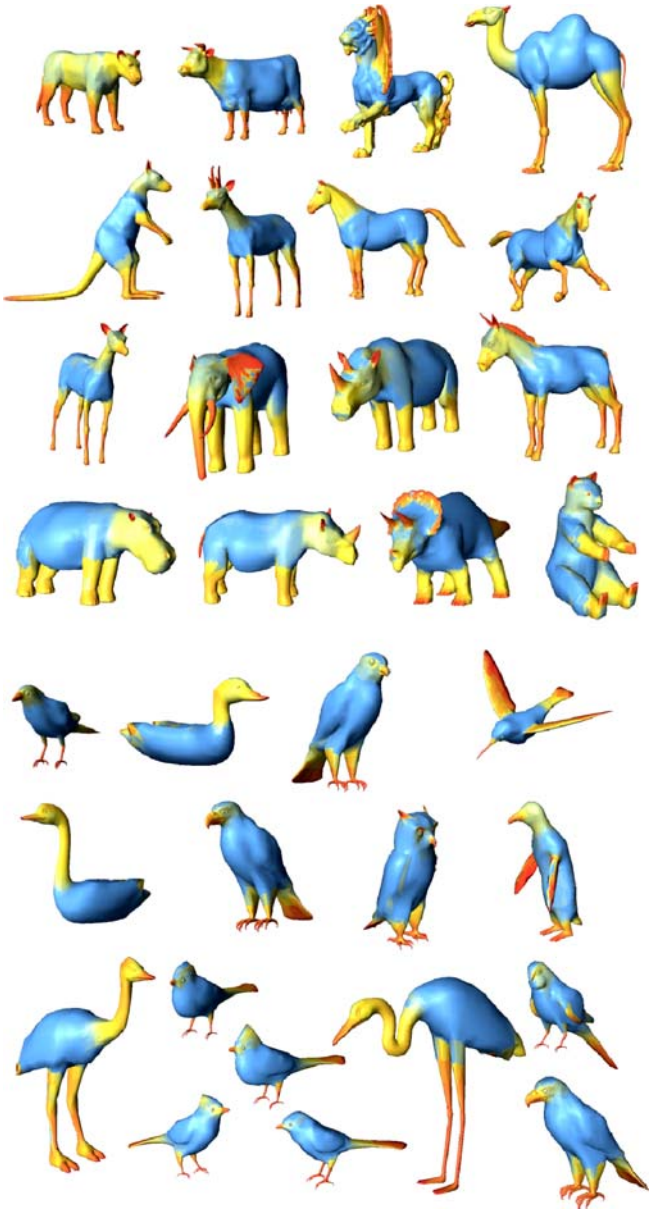

Fig. 6. The shape diameter function highlights similar parts in similar 3D shapes, for instance in animals (*top*) or birds (*bottom*). Colors indicate the value of the diameter function – from *red* (small) to *blue* (large)

where $\mathrm{sdf} : F \to \mathbb{R}$ is the SDF value for each facet f. α is a normalizing parameter, which is set to 4 in all our examples.

If we view the surface of the mesh as the domain, then the SDF is a scalar function over this domain. The iso-values of the SDF create iso-contours on the mesh and could be used to separate parts with distinctively different SDF values. Choosing several such iso-values can create a hierarchal partitioning of the object. For example, one iso-value may separate the arms, legs and head of a human

model from the body, while the second and third values might separate the hands and the fingers. This means that there is also a simple and intuitive mapping between the number of separating SDF values and the partitioning hierarchy – more values will create finer partitioning.

Our partitioning algorithm is composed of two steps. The first uses soft-clustering of the mesh elements (faces) to k clusters based on their SDF values, and the second finds the actual partitioning using k-way graph-cut to include local mesh geometric properties. Note that k, the number of clusters chosen, is more naturally related to the number of levels in the hierarchy and not to the number of parts.

In the first step we use a Gaussian mixture model (GMM) fitting k Gaussians to the histogram of SDF values of the faces. This is achieved using the expectation-maximization (EM) algorithm, which takes very little time since the histogram is only 1D. For further details regarding GMM and EM, we refer the reader to [7, 33]. The result of this clustering process for each face is a vector of length k, signifying its probability to be assigned to one of the SDF clusters. Note that each SDF cluster may contain multiple mesh parts such as legs or fingers. Using different values of k results in different hierarchies of parts (Fig. 7).

In the final hard partitioning of the object we would like to smooth the boundaries between parts and adhere to local mesh features such as concave areas or creases. The second step of our partitioning algorithm employs an alpha expansion graph-cut algorithm [35] to arrive at the final partitioning. The k-way graph cut assigns a single partition for each face, taking into account both the probability vector from the EM step, and the quality of the boundaries (smoothness and concave areas, where we measure concavity as the positive dihedral angle between two faces, normalized to [0, 1]). The graph cut optimization minimizes the following energy functional, which is built from e_1 the *data term*, and e_2, the *smoothness term*:

$$E(\overline{x}) = \sum_{f \in F} e_1(f, x_f) + \lambda \sum_{\{f,g\} \in N} e_2(x_f, x_g)$$

$$e_1(f, x_p) = -\log(P(f|x_f) + \varepsilon)$$

$$e_2(x_f, x_g) = \begin{cases} -\log(\theta(f, g)/\pi) & x_f \neq x_g \\ 0 & x_f = x_g, \end{cases}$$

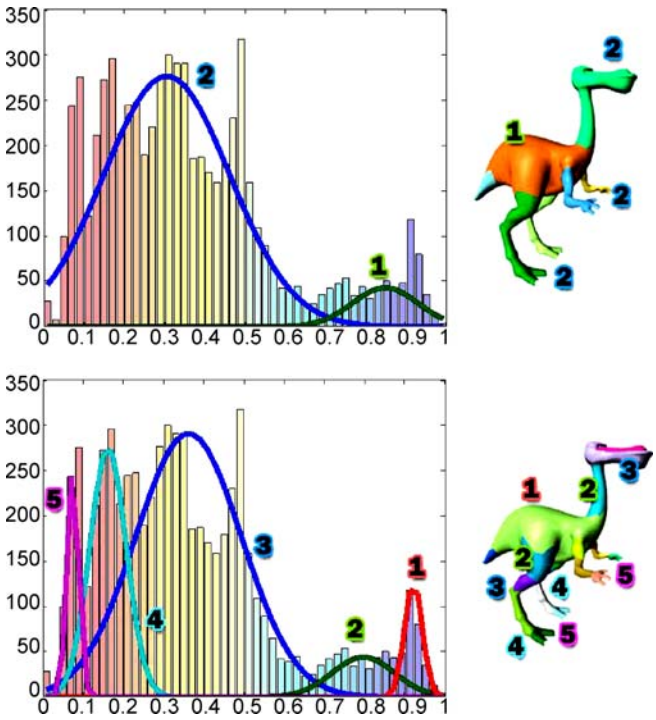

Fig. 7. Fitting the histogram of the SDF with more Gaussians results in finer partitioning of the object to parts

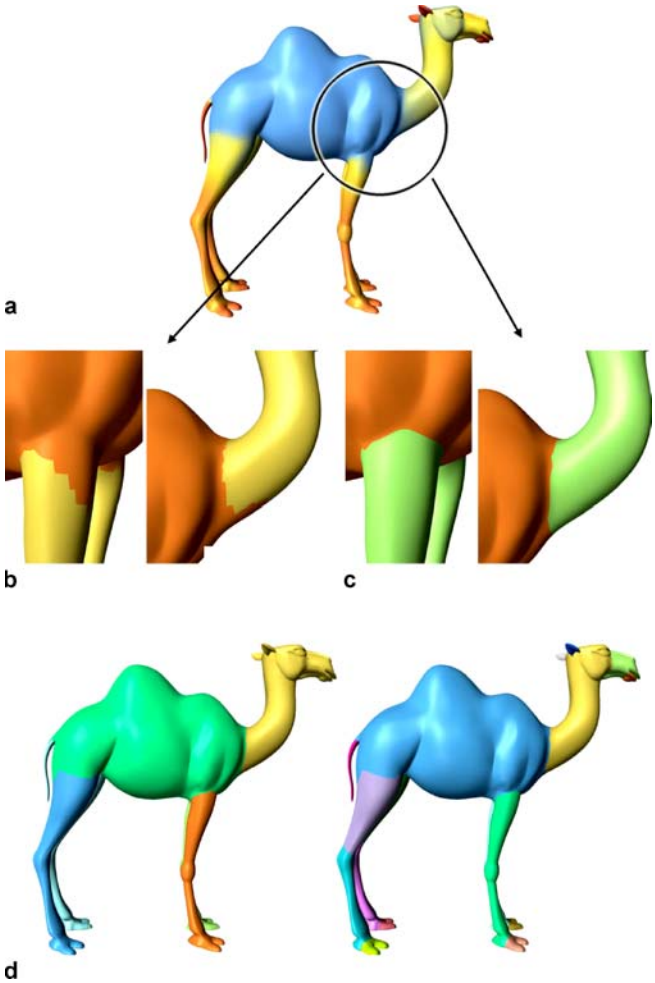

Fig. 8a–d. The two step partitioning algorithm: **a** SDF values are clustered to create soft partitioning of the mesh. **b** After the first step we show the soft partitioning using simple thresholding. **c** The partition is refined by the second step of k-way graph-cut algorithm that creates smooth boundaries and adheres to local features such as concave areas on the mesh. **d** The hierarchical partitioning is robust, pose oblivious and fast

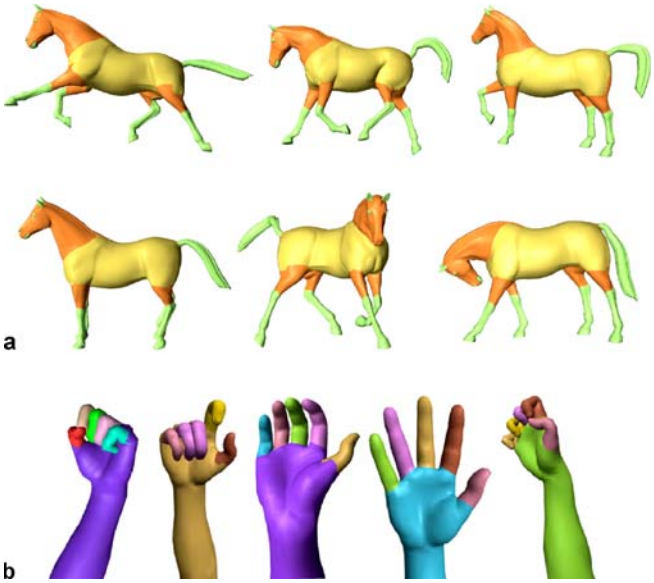

Fig. 9a,b. Despite pose changes, partitioning based on the SDF function remains consistent as can be seen on the horse model (**a**) and human hand (**b**)

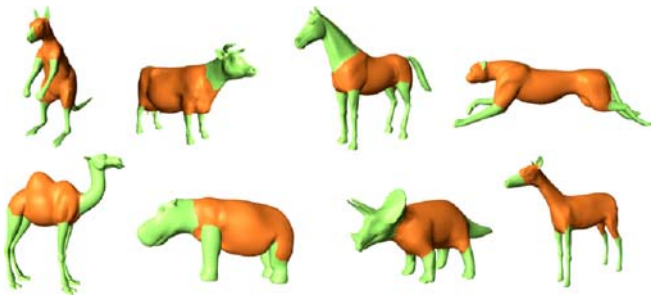

Fig. 10. Simple partitioning (one level only) on various animal meshes reveals similar parts in all of them

where x_f is the cluster assigned to face f. $P(f|x_p)$ represents the probability of assigning face f to cluster p. These value are derived from the GMM fitted in the first step of the algorithm. N is a set of adjacent face pairs in the mesh, λ is a parameter defining the degree of smoothness, and $\theta(f, g)$ is the dihedral angle between facets f and g. The result of the graph cut algorithm is a smooth partitioning of the mesh, clearly separating distinct parts. In many cases, by sorting the means of the GMM model from large to small, we can also define a hierarchical partitioning of mesh elements. The first level of the partitioning separates the largest value (and its associated faces) from all other values. The second level adds a separation between the second largest value and all the other values, and so on (Fig. 8).

Because it is based on the pose oblivious SDF, this partitioning technique remains consistent through pose differences of the same object (Fig. 9). To demonstrate consistency, we measured the percent of faces assigned to different parts in different poses on all the pairs of the horse poses and found it is around 2% (Table 2). Moreover, the results of our technique can create consistent parts in meshes of different objects as long as their

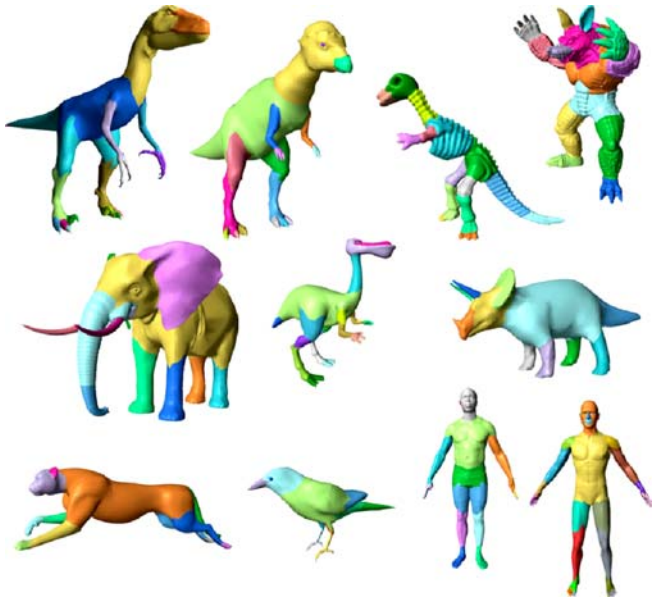

Fig. 11. Additional partitioning results

Table 2. Percent of triangles assigned to a different part between all pairs of poses of the horse in Fig. 9. The average percent is 2.2013%

Pose	1	2	3	4	5	6
1	0	2.314	3.234	2.761	1.677	2.842
2	2.314	0	4.0789	3.605	2.521	3.719
3	3.234	4.078	0	0.909	2.875	2.842
4	2.761	3.605	0.909	0	2.456	2.129
5	1.677	2.521	2.875	2.456	0	1.655
6	2.842	3.719	2.842	2.129	1.655	0

Table 3. Timing results for the steps needed to perform consistent partitioning. Timing results are for Pentium 4 1.8 GHz with 2 GB RAM and Nvidia Ge-Force FX-5600

Model	# Faces	Build GMM	Soft partitioning	K-way graph cut
Dino	5 k	20 ms	20 ms	30 ms
Triceratops	5 k	21 ms	20 ms	28 ms
Cheetah	15 k	22 ms	23 ms	31 ms
Parasour	15 k	23 ms	21 ms	31 ms
Horse	40 k	20 ms	25 ms	45 ms
Elephant	40 k	15 ms	30 ms	50 ms
Hi-Man	40 k	30 ms	25 ms	90 ms
Camel	75 k	25 ms	25 ms	100 ms
Armadilo	350 k	40 ms	35 ms	180 ms

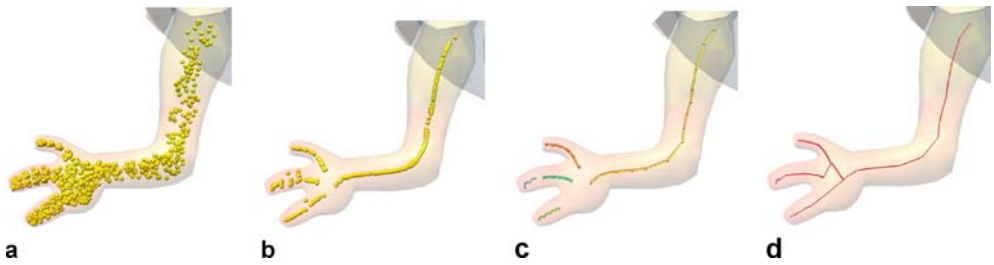

Fig. 12. **a** Projected points inside the Dino-Pet hand. **b** The points after moving least squares (MLS) projections. **c** The ordered poly-segments. **d** The final connected skeleton

shapes are similar (Fig. 10). Figure 11 shows some more partitioning results and Table 3 gives indication for the running time of the algorithm, which is on the order of seconds.

5 SDF-guided skeleton extraction

The curve skeleton of a 3D object is a one-dimensional graph structure consisting of nodes (joints) and edges (bones) supporting the mesh object from within. In graphics and animation it is important to link between the skeleton and the boundary of the object. Similar to the partitioning consistency problem, most previous skeletonizing algorithms may produce different skeletons for different poses of the same object. Based on the SDF we propose a fast and simple algorithm to create a skeleton structure based on the SDF. The skeleton remains consistent over pose changes of the object and produces analogue skeletons in analogue parts of different objects.

Our key idea is to exploit the fact that the SDF approximates twice the distance to the medial axis in most points on the mesh surface. We begin by selecting a set of N random points on the surface (not more than 10 000 points are used in the examples in this paper), and project them in an inward normal direction to a distance which is *half* of their SDF value. This creates a point cloud P inside the object that lies near the medial axis. The points in P already approximate the general structure of the skeleton (Fig. 12a), and retain a connection to their origin points on the surface.

Next, we use the moving least squares (MLS) method [17] to fit a single non-intersecting high-degree curve onto the noisy point set P. This algorithm is similar to [15], compensating for outliers and filtering out regions which do not have a good candidate for a one-dimensional skeleton. However, in contrast to [15], we search for a *tree-like* structure of line-segments connected at sharp points. Each projected point p in the point cloud P retains its distance to the surface as radius(p). This value approximates the medial axis radius at this point. For each $p \in P$ we compute the principal component analysis (PCA) of the point's neighborhood within radius(p) and project p onto its regression line. Next, we define a confidence measure using all three principal components' eigen-values e_1, e_2, e_3 as follows:

$$\text{confidence}(p) = \frac{\max(e_1, e_2, e_3)}{e_1 + e_2 + e_3}.$$

Intuitively, we are looking for points which would fit well on a curve-skeleton; therefore, the confidence measure is based on the relative strength of the first principal component to the other two. A low confidence indicates that no definite skeleton can be extracted in this region. These parts are usually where several skeleton segments connect (e.g., the base of the hand where the fingers meet), or parts which are not generally cylindrical. These parts will be connected at the final stage of the algorithm. We iterate the projection procedure several times until convergence (usually only 2 to 4) and then use only points with high confidence measure. The final output of this algorithm is a cloud of points T which closely resembles a one-dimensional skeleton (Fig. 12b). Following the iterations enforces a minimal distance ε between the points in T, removing points which are too close together.

In the next stage we cluster and order all points in the thinned point cloud T to create skeleton seg-

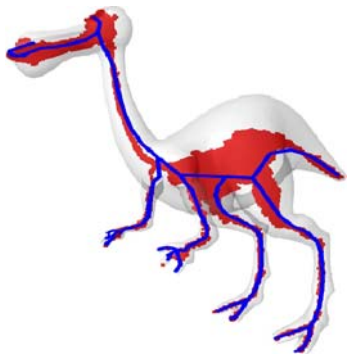

Fig. 13. The SDF based skeleton (superimposed in *blue*) closely resembles the medial axis (in *red*, calculated using a 256^3 voxelized grid)

ments. We choose the point with highest confidence $\max_{\text{confidence}}\{p \in T\}$ and grow its skeleton segment by clustering all points near its regression line. This process continues recursively in both directions of the regression line by growing poly-segments. Once we cannot extend the poly-segment anymore, we choose the next best point that is still not clustered and proceed to create more poly-segments until all points have been processed. The result of this stage is a small set S of disconnected poly-segments $(s_1, \ldots, s_n)$, each representing one skeleton part (Fig. 12c).

Table 4. Timing results for the steps needed to perform skeleton extraction. Timing results are for Pentium 4 1.8 GHz with 2 GB RAM and Nvidia Ge-Force FX-5600

Model	# Faces	Point proj.	MLS thinning	Segment connection
Dino	5 k	20 ms	1.5 s	1.2 s
Triceratops	5 k	20 ms	1.8 s	1.5 s
Cheetah	15 k	21 ms	2 s	2.5 s
Parasour	15 k	25 ms	2.5 s	4 s
Horse	40 k	20 ms	2.6 s	1.8 s
Elephant	40 k	20 ms	2.4 s	4 s
Hi-Man	43 k	25 ms	5 s	3 s
Camel	75 k	25 ms	3.7 s	2.5 s
Armadilo	350 k	40 ms	5 s	2.2 s

Finally, we connect the poly-segments together to form a single one-dimensional skeleton. Our goal is to minimize the number of joints between segments while still retaining the shape of the skeleton. Hence, we construct more poly-segments connecting the ones created in the previous stage. We use a k-terminals shortest-paths algorithm between any k disconnected poly-segments in

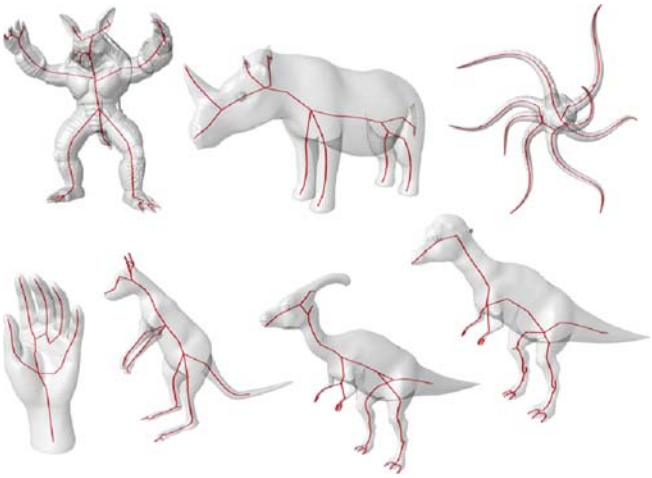

Fig. 14. Examples of curve-skeletons of different models

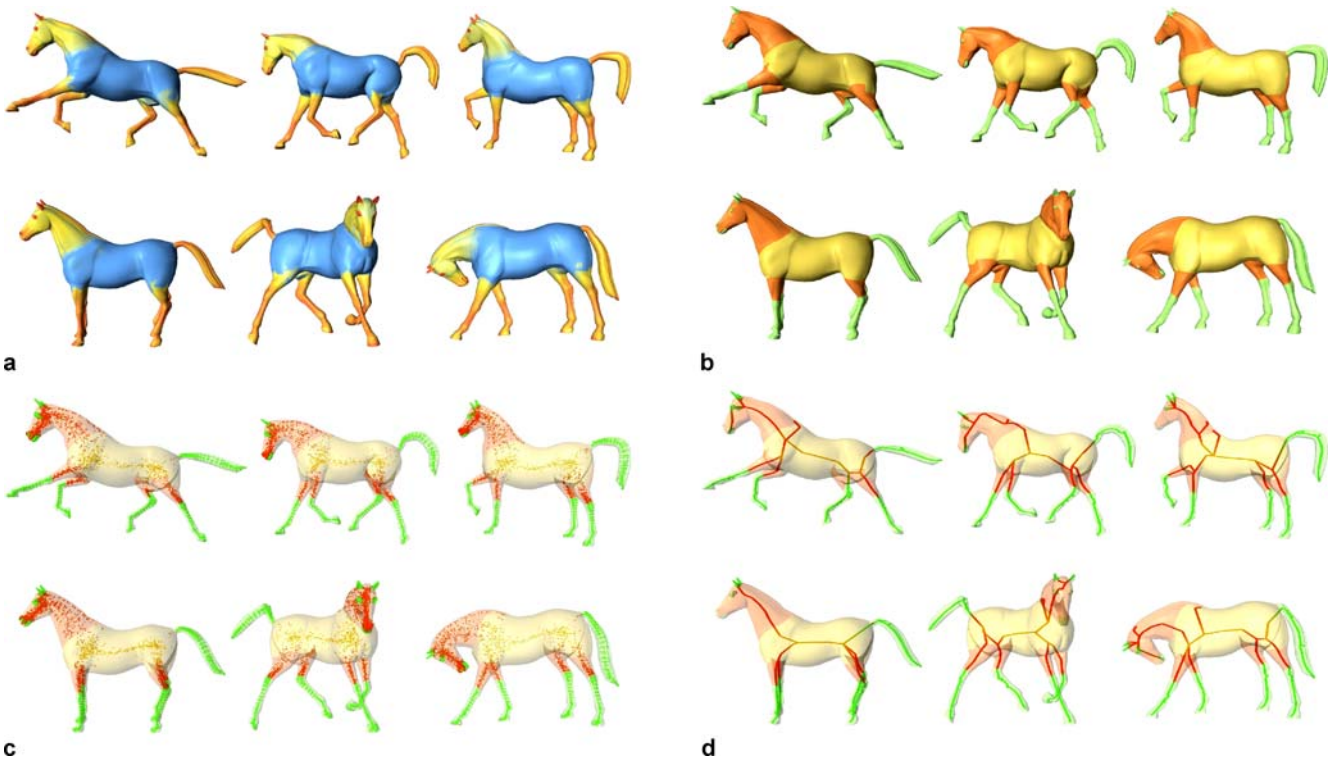

Fig. 15. **a** Consistent SDF on six horses poses. **b** Consistent partitioning. **c** The projected points retain a strong connection to the surface. Each projected point is colored in it's matching face's partitioning color. **d** The final consistent skeleton, strongly tied to the surface

Fig. 16a,b. The SDF value on non-articulated objects (**a**), and the results of partitioning and skeletonisation on such objects (**b**)

a given neighborhood. We search for the shortest path using the original points in P, since the points in P approximate the medial axis of the model being skeletonized and are thus good candidates for connecting the poly-segments.

We initialize a priority queue by finding the closest points $p \in P$ to each segment and store them with the distance from the segment. While the priority queue still holds points, we extract the closest point (minimal distance) and add its nearest neighbors to the queue. This process grows poly segments in parallel extending several segments while each new point extracted from the queue is associated with one of the original segments. When two grown poly segments meet, their two respective segments are connected. We stop the growing process when all segments have been linked (Fig. 12d) together. This process also maintains the link from the skeleton to the boundary surface through the points in P. For generalized cylinders the final results of the skeleton extraction algorithm lies inside the model and approximates the medial axis well (Fig. 13).

The whole process takes a few seconds even on large meshes as can be seen in Table 4. In Fig. 14 we demonstrate results of our skeleton extraction algorithm on a variety of models. A key advantage of using SDF for skeletonization is that the skeletons of the same model in different poses remain consistently connected to the model parts (Fig. 15).

6 Conclusions

Dealing with families of objects instead of a single one may impose further challenges on regular geometric algorithms and mesh processing. In this paper we considered the problems of partitioning and skeleton extraction of a family of 3D meshes. The challenge is to retain consistency within a given group of objects, such as different poses of the same model or different objects having similar shapes. By using the shape diameter function as the main attribute in segmentation and skeletonisation we manage to achieve such consistency. Using the SDF may have its limitation on non-cylindrical parts of objects (see e.g., Fig. 16b). Still, for most graphics and animation characters, this algorithm is successful, simple and fast.

In the future we would like to extend the scope of the consistent algorithms for partitioning and skeletonisation to a larger type of shapes, such as mechanical designed objects and more. We would like to utilize the SDF and mesh analogies to create more complex motion transfer. The SDF is a powerful tool associating *volumetric* information to the surface mesh. We believe the definition of the shape diameter-function can assist many other geometry processing and mesh related problems such as morphing and editing, matching and partial matching.

References

1. Amenta, N., Choi, S., Kolluri, R.: The power crust, unions of balls, and the medial axis transform. Comput. Geom. Theory Appl. **19**(2,3), 127–153 (2001)
2. Attene, M., Biasotti, S., Spagnuolo, M.: Shape understanding by contour-driven retiling. Visual Comput. **19**(2,3), 127–138 (2003)
3. Attene, M., Falcidieno, B., Spagnuolo, M.: Hierarchical mesh segmentation based on fitting primitives. Visual Comput. **22**(3), 181–193 (2006)
4. Choi, H., Choi, S., Moon, H.: Mathematical theory of medial axis transform. Pac. J. Math. **181**(1), 57–88 (1997)
5. Cohen-Steiner, D., Alliez, P., Desbrun, M.: Variational shape approximation. ACM Trans. Graph. **23**(3), 905–914 (2004)
6. Cox, M., Cox, T.: Multidimensional Scaling. Chapman and Hall, London (1994)
7. Dasgupta, S.: Learning mixtures of gaussians. Tech. Rep. UCB/CSD-99-1047, EECS Department, University of California, Berkeley (1999)
8. Dey, T., Giesen, J., Goswami, S.: Shape segmentation and matching with flow discretization. In: Proceedings of the Workshop on Algorithms and Data Structures (WADS 03). Lect. Notes Comput. Sci., vol. 2748, pp. 25–36. Springer, Berlin/Heidelberg (2003)
9. Dey, T.K., Zhao, W.: Approximating the medial axis from the voronoi diagram with a convergence guarantee. In: Algorithms - ESA 2002: 10th Annual European Symposium, pp. 387–398. Springer, Heidelberg (2002)
10. Gelfand, N., Guibas, L.J.: Shape segmentation using local slippage analysis. In: SGP'04: Proceedings of the 2004 Eurographics/ACM SIGGRAPH symposium on Geometry processing, pp. 214–223. ACM Press, New York, NY, USA (2004)
11. Hilaga, M., Shinagawa, Y., Kohmura, T., Kunii, T.L.: Topology matching for fully automatic similarity estimation of 3D shapes. In: Proceedings of the 28th Annual Conference on Computer graphics and Interactive Techniques (SIGGRAPH '01), pp. 203–212. ACM, New York, NY (2001)
12. Katz, S., Leifman, G., Tal, A.: Mesh segmentation using feature point and core

extraction. Visual Comput. (Pac. Graph.) **21**(8–10), 865–875 (2005)
13. Katz, S., Tal, A.: Hierarchical mesh decomposition using fuzzy clustering and cuts. ACM Trans. Graph. (Proceedings SIGGRAPH 2003) **22**(3), 954–961 (2003)
14. Kraevoy, V., Julius, D., Sheffer, A.: Shuffler: Modeling with interchangeable parts. Visual Comput. (2007) (to appear)
15. Lee, I.K.: Curve reconstruction from unorganized points. Comput. Aided Geom. Des. **17**(2), 161–177 (2000)
16. Lee, Y., Lee, S., Shamir, A., Cohen-Or, D., Seidel, H.P.: Intelligent mesh scissoring using 3D snakes. In: Proceedings of the 12th Pacific Conference on Computer Graphics and Applications, pp. 279–287. IEEE Computer Society, Washington DC (2004)
17. Levin, D.: The approximation power of moving least-squares. Math. Comput. **67**(224), 1517–1531 (1998)
18. Lien, J.M., Amato, N.M.: Approximate convex decomposition of polygons. In: SCG '04: Proceedings of the twentieth annual symposium on Computational geometry, pp. 17–26. ACM Press, New York, NY, USA (2004)
19. Lien, J.M., Amato, N.M.: Simultaneous shape decomposition and skeletonization. Tech. rep., Texas AM University (2005)
20. Liu, R., Zhang, H.: Segmentation of 3D meshes through spectral clustering. In: The 12th Pacific Conference on Computer Graphics and Applications (PG'04), pp. 298–305. IEEE Computer Society, Seoul (2004)
21. Mangan, A., Whitaker, R.: Partitioning 3D surface meshes using watershed segmentation. IEEE Trans. Vis. Comput. Graph. **5**(4), 308–321 (1999)
22. Mortara, M., Patanè, G.: Shape-covering for skeleton extraction. Int. J. Shape Modeling **8**(2), 139–158 (2002)
23. Mortara, M., Patanè, G., Spagnuolo, M., Falcidieno, B., Rossignac, J.: Blowing bubbles for multi-scale analysis and decomposition of triangle meshes. Algorithmica **38**(1), 227–248 (2003)
24. Ni, X., Garland, M., Hart, J.C.: Fair morse functions for extracting the topological structure of a surface mesh. ACM Trans. Graph. **23**(3), 613–622 (2004)
25. Page, D., Abidi, M., Koschan, A., Zhang, Y.: Object representation using the minima rule and superquadrics for under vehicle inspection. In: Proceedings of the 1st IEEE Latin American Conference on Robotics and Automation, pp. 91–97 (2003)
26. Page, D., Koschan, A., Abidi, M.: Perception-based 3D triangle mesh segmentation using fast marching watersheds. In: Conference on Computer Vision and Pattern Recognition (CVPR '03) – Volume II, pp. 27–32. IEEE Computer Society, Los Alamitos, CA (2003)
27. Shamir, A.: Segmentation and shape extraction of 3D boundary meshes. In: State-of-the-Art Report, Proceedings Eurographics 2006, The Eurographics Association (2006)
28. Shlafman, S., Tal, A., Katz, S.: Metamorphosis of polyhedral surfaces using decomposition. Comput. Graph. Forum **21**(3) (2002) (Proceedings Eurographics 2002)
29. Svensson, S., di Baja, G.S.: Using distance transforms to decompose 3d discrete objects. Image Vis. Comput. **20**(8), 529–540 (2002)
30. Tierny, J., Vandeborre, J.P., Daoudi, M.: 3d Mesh Skeleton Extraction Using Topological and Geometrical Analyses. In: 14th Pacific Conference on Computer Graphics and Applications (Pacific Graphics 2006), pp. 85–94. Taipei, Taiwan (2006) (URL http://www.lifl.fr/~tierny/pacific06.html)
31. Tierny, J., Vandeborre, J.P., Daoudi, M.: Topology driven 3D mesh hierarchical segmentation. In: IEEE International Conference on Shape Modeling and Applications (SMI'2007). IEEE, Lyon (2007) (URL http://www.lifl.fr/~tierny/smi07.html)
32. Verroust, A., Lazarus, F.: Extracting skeletal curves from 3D scattered data. Visual Comput. **16**(1), 15–25 (2000) (URL citeseer.ist.psu.edu/verroust97extracting.html)
33. Vlassis, N., Likas, A.: A greedy EM algorithm for Gaussian mixture learning. Neural Process. Lett. **15**(1), 77–87 (2002) (URL citeseer.ist.psu.edu/article/vlassis00greedy.html)
34. Wu, F.C., Ma, W.C., Liang, R.H., Chen, B.Y., Ouhyoung, M.: Domain connected graph: the skeleton of a closed 3d shape for animation. Visual Comput. **22**(2), 117–135 (2006)
35. Zabih, R., Kolmogorov, V.: Spatially coherent clustering using graph cuts. cvpr **02**, 437–444 (2004) (DOI http://doi.ieeecomputersociety.org/10.1109/CVPR.2004.238)
36. Zhu, S.C., Yuille, A.L.: Forms: A flexible object recognition and modeling system. Int. J. Comput. Vis. **20**(3), 187–212 (1996)
37. Zuckerberger, E., Tal, A., Shlafman, S.: Polyhedral surface decomposition with applications. Comput. Graph. **26**(5), 733–743 (2002)

LIOR SHAPIRA is a PhD candidate and research assistant at the Tel-Aviv Univeristy in Israel. Shapira received his masters in computer science from the Tel-Aviv University in 2005. His research interest are computer graphics and geometric modeling. Contact him at liors@post.tau.ac.il.

ARIEL SHAMIR is a senior lecturer at the interdisciplinary center in Herzliya, Israel. Shamir studied mathematics and computer science at the Hebrew University in Jerusalem and received his PhD in computer science in 1999. Shamir is currently a visiting scientist at Mitsubishi Electric Research Labs. His research interest include geometric modeling, computer graphics, visualization and machine learning. Contact him at arik@idc.ac.il.

DANIEL COHEN-OR is a professor at the School of Computer Science at Tel-Aviv University. He received a B.Sc. in both mathematics and computer science (1985), an M.Sc. in computer science (1986) from Ben-Gurion University, and a Ph.D. from the Department of Computer Science (1991) at State University of New York at Stony Brook. His current research interests include shape modeling, visibility, and image synthesis. Contact him at dcor@tau.ac.il.