Eurographics Symposium on Geometry Processing 2009
Marc Alexa and Michael Kazhdan
(Guest Editors)

Volume 28 (2009), Number 5

Multi-objective shape segmentation and labeling

P. Simari, D. Nowrouzezahrai, E. Kalogerakis, K. Singh

Dynamic Graphics Project, University of Toronto

Abstract

Shape segmentations designed for different applications show significant variation in the composition of their parts. In this paper, we introduce the segmentation and labeling of shape based on the simultaneous optimization of multiple heterogenous objectives that capture application-specific segmentation criteria. We present a number of efficient objective functions that capture useful shape adjectives (compact, flat, narrow, perpendicular, etc.*) Segmentation descriptions within our framework combine multiple such objective functions with optional labels to define each part. The optimization problem is simplified by proposing weighted Voronoi partitioning as a compact and continuous parametrization of spatially embedded shape segmentations. Separation of spatially close but geodesically distant parts is made possible using multi-dimensional scaling prior to Voronoi partitioning. Optimization begins with an initial segmentation found using the centroids of a k-means clustering of surface elements. This partition is automatically labeled to optimize heterogeneous part objectives and the Voronoi centers and their weights optimized using Generalized Pattern Search. We illustrate our framework using several diverse segmentation applications: consistent segmentations with semantic labels, bounding volume hierarchies for path tracing, and automatic rig and clothing transfer between animation characters.*

Categories and Subject Descriptors (according to ACM CCS): Computational Geometry and Object Modeling [I.3.5]: Geometric algorithms, languages, and systems.—

1 Introduction

Digital 3D models, represented as polygon meshes or point clouds, are extensively used in engineering design, animation, games and medicine. These models are invariably structured into parts dictated by their application: limbs for articulated character rigging, functional components for engineering simulations, tight bounding volumes for efficient rendering or labeled anatomic structures for medical visualization. The partitioning of a model into such parts is referred to as shape segmentation, and its automation for various applications is an important area of ongoing research.

Most object classes and segmentation applications have domain-specific knowledge that can help define object parts as well as provide them with semantically meaningful labels. Humans for example [Ric08] have well-documented relationships between different body parts. The humanoids in Fig. 9, after being consistently segmented and labeled to have a head, torso, arms and legs, can be automatically rigged, clothed and even morphed into each other.

While it is not reasonable for users to manually segment and label hundreds of shapes, one can imagine that a user-specified description of segments and labels can suffice to automatically segment and label an entire class of objects consistently. Current shape segmentation algorithms, unfortunately, are designed to optimize fixed objectives such as concave boundaries [KT03] and ellipsoidal [SS05], symmetric [MGP06] or convex [KJS07] parts. These approaches typically limit user input to the number of desired segments and a handful of threshold values. The current approach to incorporate domain knowledge is to design and implement a new segmentation algorithm, which is infeasible for most users, *e.g.* artists, animators, medical experts.

Contributions

We present a framework for shape segmentation that allows for the incorporation of domain-specific knowledge through arbitrary objective functions that geometrically describe each segment (with an optional label). These objectives can be unary, asserting properties of an individual part,

e.g. that it should be narrow, compact, flat, or they can by n-ary, referring to part inter-relations, *e.g.* a set of parts should be parallel to each other, or have the same approximate size, or two parts should be perpendicular to each other. We thus cast the segmentation problem as a black-box optimization minimizing an aggregate objective function over the space of segmentations.

Current approaches to shape segmentation are surface-based in that they explicitly represent segmentations as partitions of mesh faces, the connectivity of segments being separately enforced. The combinatorial complexity of all possible surface segmentations is enormous, and motivates both a more compact representation of surface segmentations as well as efficient approaches to label assignment and objective function minimization. We propose a volume-based segmentation, where a weighted Voronoi partitioning of space implicitly defines a segmentation of any embedded shape (see Fig. 2 and 4). Shape segmentations are thus continuously parameterized by the Voronoi centers and weights defining each segment. Objects with complex articulations are handled by embedding geodesic distances into Euclidean space using multi-dimensional scaling (MDS) prior to Voronoi partitioning (see Fig. 3).

The Voronoi centers representing each part are initialized using a k-means clustering of the given shape, where k is the known number of parts. The initial segmentation is then automatically labeled according to the given objectives. We show optimal label assignment, when binary objectives are involved, to be NP hard but adapt an $O(n^3)$ algorithm for optimally labeling unary objective functions and present an evolutionary approach to label n-ary objectives. We then optimize the Voronoi centers and weights using Generalized Pattern Search.

Voronoi partitioning can easily be made to preserve useful geometric properties such as symmetry, allowing us to (if desired) ensure a symmetric segmentation as well as reduce the dimensionality of the optimization domain for symmetric parts. Optimization complexity can also be reduced within our framework by hierarchical segmentation.

Finally, we show the generality of our approach by applying it to several diverse segmentation problems: consistent segmentations with semantically meaningful labels, bounding volume hierarchies for path tracing, automatic character rigging, and automatic transfer of clothing between animation characters.

2 Related work

Current segmentation algorithms can be broadly classified into the following groups.

Affinity: In affinity-based approaches, affinity metrics define the likelihood of every pair of surface elements being in the same segment. Segmentation algorithms seek to maximize intra-segment similarity and inter-segment dissimilarity based on these metrics. Examples include graph cut [KT03,KLT05,PSG*06,LZ07,LZSCO09,GF08] and watershed approaches [MW99, ZTS02]. Spectral embeddings in which Euclidean distances correlate to the affinities between surface elements [LZ07] can allow for affinity-based approaches to lend themselves well to volume-based segmentation.

Model fitting: In model fitting approaches, every segment of the mesh is assumed to be independently generated by a different parameterized model. Model parameters can be fit to candidate parts and, conversely, surface elements classified given a configuration of models. Optimization techniques for such segmentations include region growing [LMM98], variational [SS05,JKS05] and hierarchical approaches [AFS06]. Our weighted Voronoi parametrization of segmentations shares the parametric continuity of model fitting approaches.

Property-based: Properties such as symmetry [TW05, MGP06,SKS06], convexity [CDST97,LA05,KS06,KJS07], tubularity [MPS*04], texture [LMLR06, LMLR07], diffusion over surfaces [dGV08], and modes of vibration [HWAG09], are often held to some degree locally by the segments but not by the overall surface. These properties are difficult to capture through pairwise affinities of surface elements or generative model approaches.

All these approaches have intrinsic limitations corresponding to their category. Affinity-based methods require that it be possible to evaluate the objective function for pairs of surface elements. This may be difficult for functions that require a larger shape context to sensibly evaluate, such as being developable or co-axial. Moreover, these approaches are not amenable to optimizing heterogenous objectives, *e.g.* one part being convex and the other concave. Approaches based on generative models, unfortunately, are also not applicable to some common segmentation objectives such as convexity or symmetry that are hard to formulate generatively. Finally, property-based approaches tend to be specifically tailored to produce homogeneous segmentations, *i.e.* in which all parts possess the given property. Recent surveys of segmentation algorithms [AKM*06, Sha08] show extensive research addressing a wide range of segmentation criteria but also a lack of general, automated frameworks to heterogeneously combine existing and new segmentation criteria.

3 Multi-objective shape segmentation

While current approaches work well for their intended applications, the incorporation of object- and application-specific knowledge can result in better quality and consistently labeled segmentations (see Fig. 6, 7 and 9).

Implementing a segmentation algorithm from scratch is a non-trivial task beyond the capability of most users. Users can thus convey domain knowledge to a general segmentation framework either by providing example labeled segmentations [SSSCO08], or by encoding this knowledge using a set of geometric objective functions. In this paper we

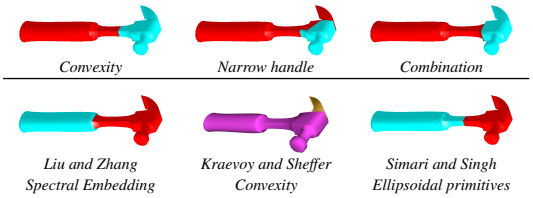
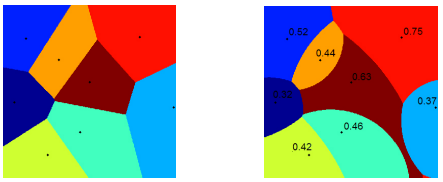


Figure 1: ***Above:*** *Our multi-objective approach. Left: optimizing for convex parts. Center: optimizing only for the narrowness of the handle. Right: optimizing a combination of these objectives and perpendicularity between head and handle. See Sec. 8.* ***Below:*** *Alternative approaches [LZ07, KJS07, SS05].*



Standard Voronoi partition *Weighted Voronoi partition*

Figure 2: *Comparison of the standard Voronoi partition induced by a random set of points and the multiplicatively weighted Voronoi partition induced by points with the same locations but varying associated weights.*

take the latter, novel approach by providing a general multi-objective segmentation and labeling framework and a working set of useful geometric objectives (see Fig. 13) that users can combine or augment to suit their application. Each part of a segmentation description is uniquely identified by a label and a multi-objective function that geometrically describes the part and (possibly) its relation to other parts. Objectives for a labeled part may thus be unary (*eg.* narrow, flat, symmetric convex, compact, developable), binary (*eg.* perpendicular to, bigger than, adjacent to), or n-ary (*eg.* parallel, coaxial, similar in size).

As a simple example, consider a hammer. This object is defined by Merriam-Webster as "a hand tool consisting of a solid head set crosswise on a handle and used for pounding." This naturally suggests two segmentation labels: *handle* and *head*. Moreover, it suggests the segmentation objective that the part labeled *head* be perpendicular to the part labeled *handle*. Our knowledge might further suggest that the parts should be approximately convex and that the part labeled *handle* should be narrow. The effects of incorporating multiple objectives into the segmentation of a hammer shape can be seen in Fig. 1.

Given a segmentation description, shape segmentation can be cast as an optimization that minimizes an aggregate objective function which combines, as the sum of squares, the objective functions describing each part label. The segmentation optimization algorithm thus decoupled from its labeled description has the following characteristics: 1) Heterogeneous objectives for different part labels allows these parts to be distinguished from each other and overcomes the drawback of a homogenous segmentation criterion assumed by existing segmentation techniques. 2) The objective functions are evaluated on fully instantiated segmentations, avoiding the need of affinity-based approaches to evaluate objectives sensibly on single face pairs. 3) As long as objective function evaluation is possible, a generative model is not strictly necessary (though certainly possible). 4) The framework easily allows for assertions not just about parts but of inter-part relations using n-ary objectives. 5) Users wishing to define a new segmentation description can select from a set of existing objective functions (see Sec. 8) and new objective functions seamlessly integrate into the framework.

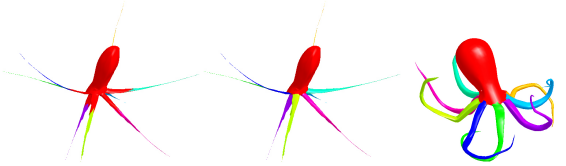
4 Parameterizing segmentations

In order to optimize a segmentation with respect to an aggregate objective function as described above, we need to be able to describe segmentations by means of some set of parameters. The set of all possible segmentations can be parameterized directly by a vector indicating the segment label of each face explicitly, but such a parametrization would be cumbersome, since the vast majority of the segmentations would be undesirable, being unconnected, having complicated boundaries, *etc.* We would also like to avoid a surface parametrization, since this is its own difficult problem, and places topological constraints on the shape representation as well as introducing distortion.

We thus propose the use of a Voronoi space partition to define the surface segmentation of an embedded shape. A set of 3D points, each corresponding to a part label, naturally induces a segmentation on the shape by classifying each surface element according to the Voronoi region it occupies. We classify polygon faces based on their centroid but a segmentation at sub-face resolution can also be defined.

A natural limitation of this representation is that Voronoi regions are only capable of describing planar boundaries. Augmenting the centers with a distance scaling weight allows curved boundaries and non-convex regions. Formally, a point p's distance to a center c with weight w is given by $||p-c||/w$. The result is known as a multiplicatively weighted Voronoi partitioning [OBSC00] (see Fig. 2 and 4).

Ensuring segment connectedness: The set of Voronoi points and weights defines a continuous-domain parametrization over a subset of all possible surface segmentations. In practice, the compact shape of Voronoi cells naturally tends to induce connected and compact surface segmentations. It is, however, possible to obtain disconnected segments. To achieve connectivity we use a priority queue flooding scheme as introduced by Cohen-Steiner *et al.* [CSAD04] and successfully used in at least one other segmentation approach [SS05].



Initial seg. in MDS | *Optimized seg. in MDS* | *Corresponding seg. in initial shape*

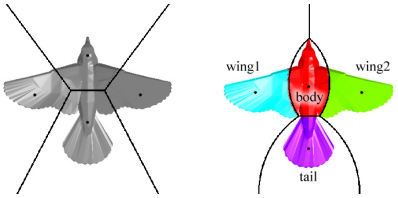
Figure 3: *Segmentation induced by the partition in MDS space is easily mapped through correspondence with the original shape. See Sec. 8 and Fig. 13 for objective details.*

Segmenting shapes under complex articulation: As expected, surface elements within a Voronoi region will tend to be proximal in Euclidean space. For many desired surface segmentations, however, geodesic proximity is more important than Euclidean proximity. While ensuring segment connectivity alleviates the problem of geodesically distant surface elements belonging to the same segment, it does not address the fact that a part with complex articulation may be hard to capture using a single Voronoi cell. We address this concern using MDS as a pre-processing step, which has proven useful in other segmentation approaches [KLT05, LZ07]. We pre-compute all pairs of approximate geodesic distances between surface elements (using shortest paths on the connectivity graph) and then use MDS to find a 3D Euclidean embedding of the shape with these distances. The result is an unfurling of articulation such that Euclidean distances in the embedded shape approximate geodesic distances in the original shape. One may now find a weighted Voronoi partition in the MDS space and optionally evaluate objective functions in this space or the original undistorted one as is appropriate for the given objective. The resulting segmentation is trivially mapped by element correspondence to the original surface. Fig. 3 illustrates this approach by cleanly segmenting a highly articulated octopus model. Figures 7 and 9 were also obtained using this technique.

5 Initial center placement

The first step in our optimization is to automatically find a coarse initial placement of partition centers which will serve as the initial guess for the optimization. We achieve this by way of k-means clustering where k is the known number of labels. This approach has the advantages of simplicity and the fact that it produces a Voronoi partition by construction, given that it is based on distance to cluster centers. It also naturally produces compact, similarly-sized segments.

The centers are initialized using furthest point initialization: choose the farthest pair of surface points and then, while centers remain to be initialized, select the point with maximum closest distance to the points chosen thus far. If adjacency information is available and connectivity of segments is desired, the classification step of the k-means algorithm can be done using the previously-mentioned queue



Unlabeled K-means seeding | *Labeled and Optimized result*

Figure 4: ***Left:*** *Result of k-means approach to initialization.* ***Right:*** *Labeled and optimized result. See Sec. 8 and Fig. 13 for objective details.*

approach. Should a segment become empty, we select as its new center the surface element furthest from the current non-empty centers. Initial segmentations resulting from this approach are illustrated in Fig. 3 (left), and 4 (left).

6 Assigning labels to segments

A set of partition centers induces a shape segmentation. In many, though not all (see Sec. 8) cases, however, the segmentation will refer to objectives of a heterogenous nature. For instance, in the hammer example of Fig. 1 the *narrow* objective term refers only to the segment labeled *handle*. This means that labels must be assigned to each center prior to objective evaluation, as different labelings produce different objective function values. When there are relatively few labels and the objective is not computationally expensive, an exhaustive approach of all permutations may be feasible. As the number of segments grows, this approach quickly becomes intractable. In the following, we present two alternatives to address this labeling problem.

Optimal unary objective labeling

If the segmentation makes use of only unary objectives, or if we consider only the set of unary objective terms for the purposes of addressing the assignment problem, it is possible to efficiently find an optimal labeling.

Assume that for each label j we have a set of k unary objective terms $\mu_{j,1}, \mu_{j,2}, \ldots, \mu_{j,k}$. We can build an $n \times n$ cost matrix C such that

$$C_{i,j} = \sum_k \mu_{j,k}(i)$$

which represents the cost of assigning label j to the part associated with center i for all i, j pairs.

Given this cost matrix, we can now cast part labeling as an assignment problem which can be optimally solved in $O(n^3)$ by the Hungarian algorithm [Kuh55, Mun57]. Given C the algorithm will return an assignment vector $\mathbf{a}$ such that $\mathbf{a}[i] = j$ indicates that the part associated with segment i should receive label j to minimize the sum cost.

Evolutionary labeling

When objectives of higher arity are involved, the problem grows in complexity. In fact, binary objectives suffice to reduce the Traveling Salesman Problem (TSP) to our labeling problem. Assume there are n cities and map them to single-point segments. Further assume we have labels $l_1, l_2, \ldots, l_n$ indicating that the city labeled l_i should be the i-th visited on the tour, and that the binary objective $dist(l_i, l_j)$ denotes the distance between the cities labeled l_i and l_j. The labeling which minimizes the aggregate objective

$$\sum_{i=1}^{n} dist(l_i, l_{(i \bmod n)+1})$$

is the solution to the given TSP instance.

Thus motivated, we address the n-ary objective labeling problem with an evolutionary approach. Such approaches have had success in the past with problems such as TSP and other ordering problems [Mic98].

An individual in our population is naturally represented using permutation encoding: a vector $\mathbf{x}$ of length n where $\mathbf{x}[i] = j$ indicates segment i will receive label j. The fitness of an individual will be determined by the value the aggregate objective function takes when using the labeling specified by the individual. We use tournament selection as our selection strategy, in which a small random subset of the previous population is considered and the individual with highest fitness is selected. An individual can be mutated by uniform-randomly selecting two of its entries and interchanging them. Finally, given two individuals $\mathbf{x}_1$ and $\mathbf{x}_2$, we can produce an offspring candidate $\mathbf{y}$ using uniform crossover. In particular, we generate a uniformly random bit vector $\mathbf{b}$ and let $\mathbf{y}[i] := \mathbf{x}_1[i]$ if $\mathbf{b}[i] = 0$ and $\mathbf{y}[i] := \mathbf{x}_2[i]$ if $\mathbf{b}[i] = 1$ unless $\mathbf{x}_2[i]$ already appears in $\mathbf{y}$ from $\mathbf{x}_1$. After crossover, any such conflicting entries are repaired, assigning them randomly from the set of remaining (unassigned) labels.

This evolutionary labeling approach is specified in Algorithm 1. In particular we use a population size of 20, a crossover fraction of 80%, a tournament size of 4, and typically achieve an optimal labeling in under 10 iterations.

Note that the labeling objective is free to differ from the segmentation objective. The labeling objective may describe part adjacencies and similarities, while the segmentation objective describes the desired final configuration of parts.

Algorithm 1 Evolutionary optimal objective labeling

```
// Initialize population P
for i = 1 to populationsize do
    P[i] ← random permutation of the n labels
loop
    // Evaluate fitness
    for i = 1 to populationsize do
        F[i] ← aggregate obj. function value with assignment P[i]
        Remember best fitness and individual so far
    i ← 1
    // Generate crossover individuals
    while i ≤ crossoverfraction * populationsize do
        x1 ← tournamentselection(P, F)
        x2 ← tournamentselection(P, F)
        P'[i++] ← uniformcrossover(x1, x2) // w/repair
    // Generate mutation individuals
    while i ≤ populationsize do
        x ← tournamentselection(P, F)
        P'[i++] ← mutate(x)
    P ← P'
return individual with best fitness observed
```

7 Segmentation optimization

Having addressed the matters of center initialization and label assignment, we now focus on segmentation optimization. If wish to obtain a segmentation consisting of n segments, it can be parameterized by a real vector $\mathbf{x}$ of dimensionality $m = 4n$ of the form

$$\mathbf{x} = (x_1, y_1, z_1, w_1, x_2, y_2, z_2, w_2, \ldots, x_n, y_n, z_n, w_n)$$

where (x_i, y_i, z_i) are the 3D coordinates of the i-th Voronoi center and w_i is its associated weight. The task is now to search for a value of $\mathbf{x}$ that minimizes the aggregate objective function when evaluated on the segmentation induced by $\mathbf{x}$.

Gradient-descent approaches are not suitable given the discrete nature of the surface representation: a center must be altered sufficiently to induce a change in the classification of at least one surface element. We instead choose generalized pattern search, which is a derivative-free, direct search method [Tor97, AJ03].

Given a pattern size Δ, at each iteration the method evaluates the aggregate objective function at all $2m$ neighbors formed by adding and subtracting Δ to each coordinate of the current $\mathbf{x}$. If any such neighbor produces a lower objective value, the iteration is considered successful, $\mathbf{x}$ is updated and Δ is multiplied by an expansion factor. Otherwise, the iteration is considered unsuccessful, no update of $\mathbf{x}$ occurs, and Δ is multiplied by a contraction factor. The algorithm terminates when Δ falls below a given threshold. In particular, we use an initial Δ of $.2d$ where d is the shape's bounding box diagonal, an expansion factor of 1.2, a contraction factor of .5 and a Δ threshold of $10^{-4}d$. We typically observe convergence in fewer than 50 optimization iterations.

Symmetry constraints: Recent methods allow for the robust automatic detection of symmetries in 3D shape [KFR04, TW05, SKS06, MGP06, PSG*06] and, whenever possible, shape processing algorithms should leverage this redundancy. An advantageous property of our framework for parameterizing mesh segmentations is that it is easy to apply symmetry constraints to optimization parameters. For instance, if one segment is known to be symmetric to another, then the latter's Voronoi center position and weight can be generated by symmetry from the first. If a segment is

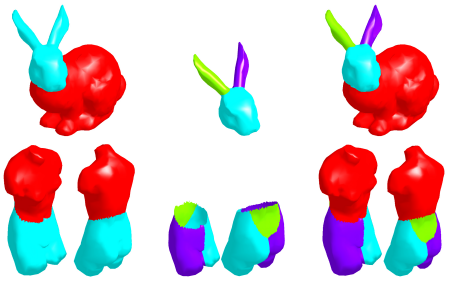


Figure 5: *Our approach can easily be used to segment by multiple objectives hierarchically.*

known to lie on a plane of global symmetry, then its Voronoi center can be constrained to lie on the plane. The advantages of exploiting this known redundancy are two-fold: first, if a shape is known to have a certain global symmetry, then the segmentation is assured to have the same symmetry by construction. Second, the dimensionality of the optimization domain is reduced by removing parameters which can effectively be generated from others. The user need only provide a constraint function which takes in the non-redundant parameters and produces the others from known symmetry. Fig. 4 illustrates this in our results on bird models by constraining one wing to be symmetric to the other and the body and tail to lie on the plane of global symmetry.

Hierarchical segmentation: Our approach can easily be used to segment by multiple objectives hierarchically. Fig. 5 illustrates a case where we first segment to maximize planar symmetry of parts, then by convex components, and then integrate the result. In Sec. 8 we will also describe how our approach can be used to obtain hierarchical bounding volumes for path tracing acceleration (Fig. 8) as well as a semantically meaningful hierarchical segmentation of humanoid characters (Fig. 9).

8 Applications

Our framework and the following applications were implemented in Matlab. Labeling plus optimization time is ~3 min. per segmentation on average, but of course this will depend on mesh density, the objective functions used, and the efficiency of their implementation. All experiments were run on a Pentium M 2.13Ghz processor with 2Gb RAM.

Semantically meaningful segmentations

Fig. 1, 3, 4, 6, 7, and 9 show examples of semantically meaningful segmentations and associated labelings obtained using our method. The decompositions are determined by user-specified objectives which encode a description of the parts of the model and their interrelations (see table in Fig. 13). We now define the objective function terms used to describe these segmentations.

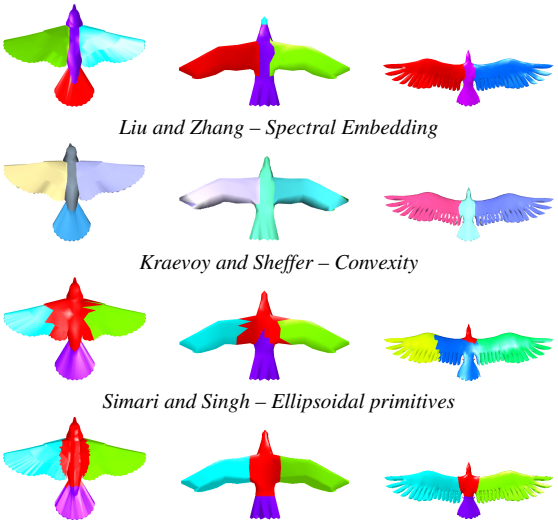


Figure 6: *Result of applying our approach (bottom row) to bird models compared to the approaches of Liu and Zhang [LZ07], Kraevoy and Sheffer [KS06] and Simari and Singh [SS05]. Note how our approach achieves semantic consistency across shapes. See Sec. 8 and Fig. 13 for objective details.*

Given a segmented surface, let P be the segment associated with a given label. Define $P.scale_i$, $i \in [1,3]$ as the part's scales resulting from PCA, *i.e.* as the square root of the eigenvalues of the surface's 3×3 covariance matrix, sorted in descending magnitude. We define the following unary objectives named according to their corresponding adjective:

$$\text{narrow}(P) := \frac{.5(P.scale_2 + P.scale_3)}{P.scale_1}$$

$$\text{flat}(P) := .5\left(\frac{P.scale_3}{P.scale_1} + \frac{P.scale_3}{P.scale_2}\right)$$

$$\text{compact}(P) := 1 - \text{narrow}(P)$$

Let us further define $P.c$ as part P's centroid and $P.axis_i$, $i \in [1,3]$ as the part P's i-th normalized eigenvector also resulting from PCA. We may then define the objectives

$$\text{planarsymmetric}(P) := \min_{\mathbf{i}} \text{surfdist}(P, \text{reflect}(P, P.c, P.axis_i))$$

$$\text{ellipsoidal}(p) := \text{surfdist}(P, \text{covarellipsoid}(P))$$

We define $\text{surfdist}(S_1, S_2)$ as the sum of area-weighted squared distance of points from S_1 to S_2 normalized by the total area of S_1 and by its bounding box diagonal length. In turn $\text{reflect}(S, p, \vec{n})$ represents the planar reflection of surface S about the plane determined by point p and normal $\vec{n}$. This is similar to the symmetry metric used by Simari *et al.* [SKS06] but is non-iterative. Naturally, we define the $\text{covarellipsoid}(P)$ as the covariance ellipsoid of part P determined by its centroid, and PCA scales and axes.

Let us also define the following n-ary objectives that refer

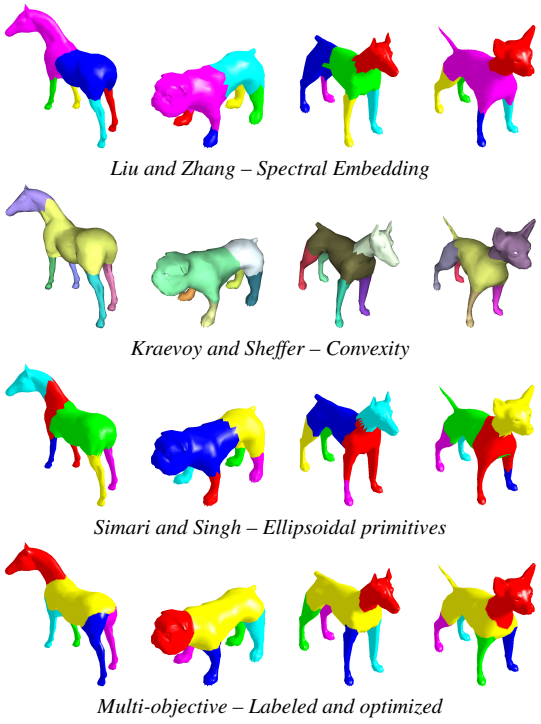


Figure 7: *Comparison of our approach to the result of applying Liu and Zhang [LZ07], Kraevoy and Sheffer [KS06] and Simari and Singh [SS05]. Note that in our results the head, body and legs are consistently segmented and labeled across models. Leg labels are considered interchangeable. See Sec. 8 and Fig. 13 for objective details.*

to label interrelations. Specifically

$$\text{perpendicular}(P_1, P_2) := |P_1.axis_1 \cdot P_2.axis_1|$$

$$\text{similarsize}(P_1, P_2, \ldots, P_k) := \frac{1}{d}\sum_i ||P_i.scale - \hat{s}||$$

where d is the global surface's bounding box diagonal, $P_i.scale$ refers to the part's entire 3×1 scale vector and $\hat{s} = \frac{1}{k}\sum_i P_i.scale$.

Finally we define the global objective

$$\text{convexparts}(Seg) := \left(\frac{1}{V}\left|\left(\sum_{P \in Seg} H(P)\right) - V\right|\right)^{\frac{1}{3}}$$

where Seg refers to the segmentation, $H(P)$ is part P's convex hull volume and V is the volume enclosed by the total original surface.

Fig. 13 describes the specific objectives and constraints (where applicable) used in each of the results of Fig. 1, 3, 4, 6 7 and 9. The first four were automatically labeled using the optimal unary approach while the last two were done using the evolutionary approach. Objective weights were found experimentally by trial and error, though there was not need of much refinement. Note that they do not change per model, but rather are kept fixed for all models in a class.

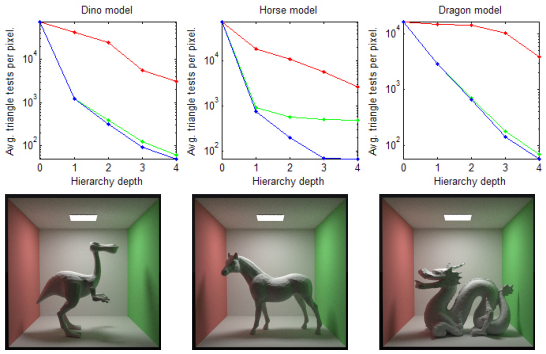


Figure 8: *Comparison of average number of triangle intersections per pixel computed (on $\log_{10}$ scale) as a function of axis-aligned bounding box hierarchy depth. Curves correspond to octree (red) and our method's results when optimizing volume tightness (green) and a combined objective of volume tightness and uniform partitioning (blue).*

Bounding volume hierarchies for path tracing

Bounding volume hierarchies are often used to accelerate path tracing. A common way to obtain such hierarchies is through the use of an octree which at each level divides the space into axis-aligned octants through the object's centroid.

We use our segmentation approach to obtain axis-aligned bounding box hierarchies by segmenting a model using an objective which optimizes bounding volume tightness:

$$\text{bbox}_1(Seg) := \sum_{P \in Seg} AABB(P)$$

where $AABB(P)$ is P's axis-aligned bounding box volume.

Alternatively, our method allows us to optimize bounding volume tightness simultaneously with distribution of faces:

$$\text{bbox}_2(Seg) := \left(\frac{1}{H(P)}\sum_{P \in Seg} AABB(P)\right)^{\frac{1}{3}} + cv(\{|P| : P \in Seg\})$$

where $H(P)$ is P's convex hull volume and cv is the coefficient of variation defined as the ratio of the standard deviation to the mean of a set. This term favors segmentations where all segments have a similar number of faces.

Fig. 8 shows a comparison of the bounding boxes obtained with octree partitioning and those obtained by our method using either objective on three scenes. In each case, we show an order of magnitude improvement in triangle intersection tests per pixel rendered, with the combined objective function showing further improvement over the single objective. Note that in this application labeling is unnecessary since the objectives are homogenous to all parts.

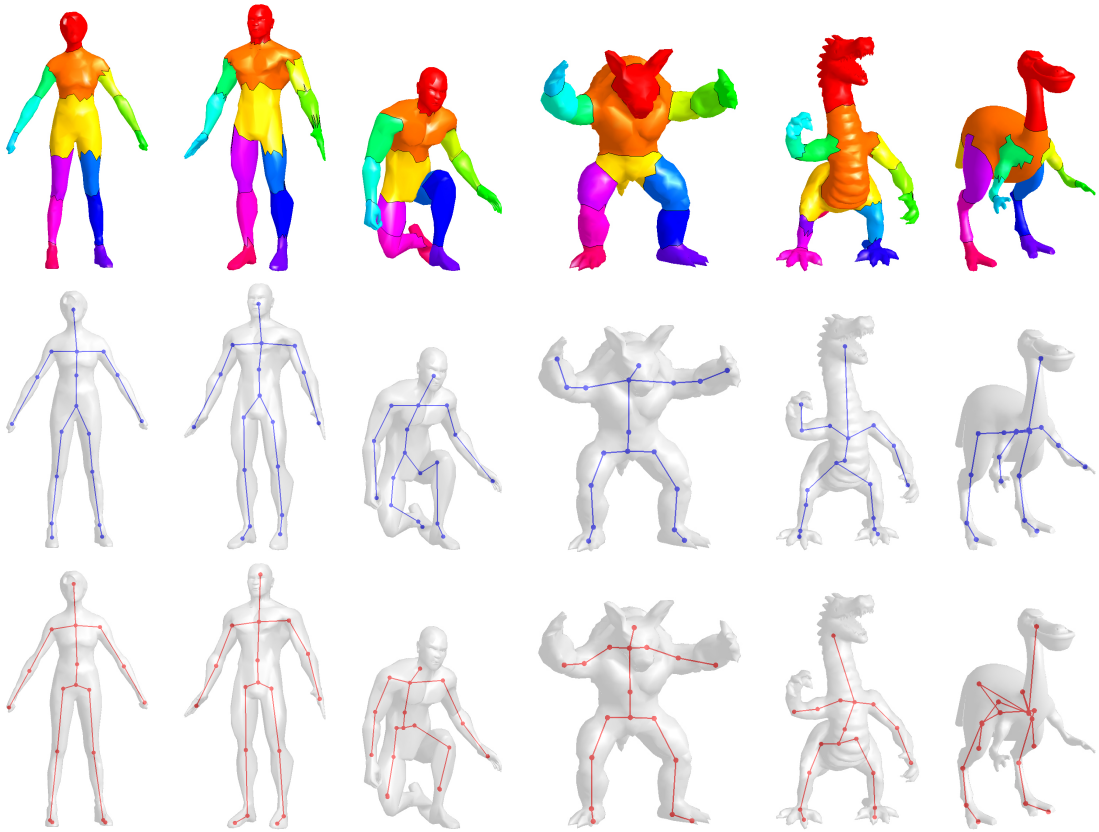


Figure 9: ***Top:*** *Our multi-objective segmentation consistent across models.* ***Center (blue):*** *Resulting skeleton placement using part and part-boundary centroids.* ***Bottom (red):*** *Skeleton placement of Baran and Popović. See accompanying video for comparison of resulting animations. See Sec. 8 and Fig. 13 for objective details.*

Automatic character rigging

Another useful application is that of correctly placing an animation skeleton, such as one available from motion capture, onto a target mesh of roughly the same class, *e.g.* bipedal humanoid. Baran and Popović [BP07] recently address this problem in their Pinocchio framework. Pinocchio discretizes the problem of embedding the skeleton by constructing a graph whose vertices represent potential joint positions and whose edges are potential bone segments. These vertices are found by packing spheres computed from the medial surface. Then, the method learns a good embedding of the skeleton into the graph and gradient descent method is used to refine the skeleton.

We propose an alternative approach using our multi-objective segmentation framework. Given a humanoid mesh:

1. Perform an MDS embedding as described in Sec. 4 to undo any articulation and perform a few steps of smoothing to remove high frequency detail that is not relevant to the coarse-level segmentation.
2. Segment the result into six parts according to a descriptive set of objectives, the part labels being *head*, *torso*, *arm_left*, *arm_right*, *leg_left*, and *leg_right*. The labeling objective asserts part adjacencies as well as similarity between the two arms and similarity between the two legs. The segmentation objectives then assert part properties and interrelationships as detailed in Fig. 13. Left and right are disambiguated assuming a close-to-canonical orientation (as Baran and Popović also do).
3. Each part is then decomposed hierarchically (see Sec. 7) in the original space into regions by means of convexity. Arms are segmented into upper arm, forearm, and hand. Legs are segmented into upper leg, lower leg and foot. The torso is segmented into chest and abdomen. These subparts are easily and unambiguously labeled based on geodesic proximity to the torso , in the case of the arms and legs, and to the head in the case of the chest and abdomen.

These objectives and hierarchy result in the segmentations shown in the left column of Fig. 9. Note the consistency achieved across a range of varied models using a general

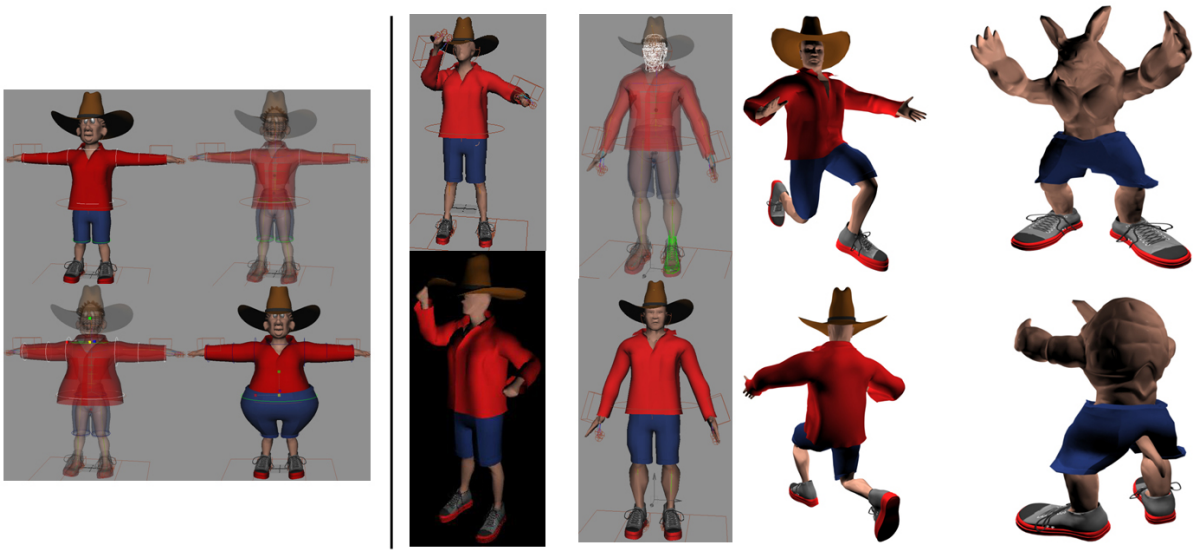


Figure 10: ***Left:*** *A character with deformers associated to its clothing.* ***Right:*** *Clothing scaled and transferred automatically onto the semantically correspondent places.*

set of objectives which describe what we know about humanoids and the animation skeleton.

The animation skeleton can now be easily placed according to this segmentation. The joints corresponding to the head, chest, abdomen, hands and feet are placed at the corresponding part centroid. Joints corresponding to shoulders, elbows, hips, knees, ankles, and mid-body are placed at the centroid of the corresponding part boundaries. The second row of Fig. 9 shows in blue the skeleton placement that results form our segmentations.

For comparison, the third row of Fig. 9 shows in red the skeleton placement produced by Pinocchio. Comparison of resulting animations can be seen in the accompanying video. Our approach results in similar placements for well posed characters, such as the first two. However, our approach needed no training since it directly received the user knowledge in the form of the segmentation objectives. Moreover, as can be observed, our approach much better handles characters under articulation as well as those with large variations in part proportions and locations such as the dino model. On the other hand, our approach will naturally fail when expected parts are not present, whereas Pinocchio can still produce a result in these situations. We thus see the two approaches as complementary.

Automatic character dressing

Another compelling application of the semantically-labeled humanoid characters above is the ability to automatically clothe such characters. Clothes and accessories parametrically defined on a segmented and labeled mannequin can be easily transferred to dress a new character.

Accessories such as the hat and shoes (see Fig. 10) are defined with respect to parts such as the head and feet on the mannequin. Dressing a given semantically-segmented character with a desired accessory requires transforming it to the correspondingly labeled part. We define this as the affine transform that maps the oriented bounding box (OBB) of the mannequin's part to the OBB of the corresponding part on the given character.

Clothing such as the shirt and shorts (Fig. 10) and accessories that drape across parts are first coarsely deformed to fit the new character and then relaxed by simulation to drape naturally. We automatically fit spline curves to the boundaries of the parts that the item of clothing drapes over. These corresponding curves (left of Fig. 10) on the mannequin and target character define a wire deformation [SF98], that coarsely reshapes the clothing around the new character. To ensure that the entire item of clothing is deformed, the radius of influence of the wire curves is set conservatively large to 1.5 times the largest bounding box dimension of its adjacent parts on the mannequin. Finally, a cloth simulation drapes the coarsely fitted clothing over the target character.

The results on three characters, female, armadillo and muscleman, are illustrated at the right of Fig. 10. As can be seen, in the case of the armadillo the transfer only works well for the shoes and not the shirt and hat, understandably, because of the oddly shaped torso and protruding ears. The coarse fitting of the shorts cause the tail to protrude resulting in the shorts relaxing over the body with a protruding tail.

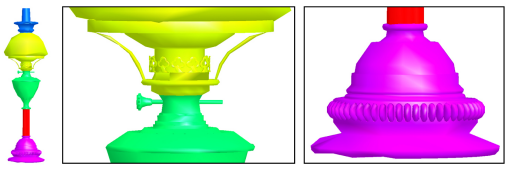


Figure 11: *Our spatial parametrization can easily handle polygon-soup models by ignoring mesh connectivity. Here we segment such a model into convex parts. Closeups show non-manifold regions with interpenetrating faces and small disconnected components. Such a model could not be handled by current methods that require manifold geometry.*

9 Limitations and future work

We recognize that there are classes of objects which our parametrization might have difficulty with and its limitations with respect to complex boundaries (Fig. 12). Our approach captures simple boundaries and more complex boundaries could be addressed in a refinement pass using min-cuts or active contours, for example.

We can perhaps look at this parametrization with the analogy of spatial volume-based vs. surface-based deformation techniques. Both have pros and cons and have been the subject of ongoing research for over 25 years. This is the first volume-based segmentation parametrization approach and is likely to stimulate further research in this direction. It also shares pros and cons analogous to those of volume-based deformations. It is simple and compact (a vector of 4k scalars, where k is the number of segments) and decouples the segmentation search from the surface resolution-dependent objective function evaluation (most objectives we use are O(n), where n is the number of surface elements).

Our approach is surface representation-agnostic, works with non-manifold or disconnected objects (Fig. 11), can address non-zero genus shapes (Fig. 12), and we show it to work well on examples typical to segmentation literature. As with the many extensions on the original box-shaped FFDs we expect work subsequent to this paper to be able to capture richer volume shapes with more complex boundaries.

We show our results on what could be considered a relatively low number of parts. However, in many real-world applications this is more than sufficient, especially (as we show) when the method is applied hierarchically. We present practical results across several such applications and show them to compare favorably to state-of-the-art methods.

Manual intervention, allowing for user placement of initial segmentation parameters, can be added to our system with no conceptual difficulty. In our work, user input can help with both the segmentation and labeling. Our goal, however, is to be able to consistently label and segment not one, but entire classes of objects, and our approach is the first to automate this process based on multiple heterogeneous black-box objectives.

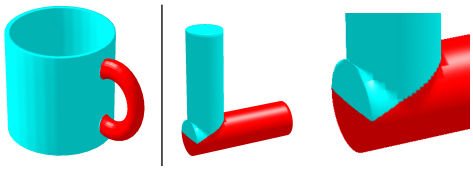


Figure 12: ***Left:*** *Our spatial parametrization can segment shapes of non-zero genus.* ***Right:*** *Our approach captures simple boundaries. More complex boundaries could be addressed in a refinement pass using,* e.g. *min-cuts or active contours. Both models are segmented for convex parts.*

The formulation of the objective functions is not an automated process. However, a single weighted combination of simple objectives, once defined, can suffice to segment and label an entire class of objects, as illustrated by Fig. 6, 7, and 9. While the objective functions we provide have intuitive meaning and are easy to reuse, we acknowledge that the choosing of these objectives and the fine tuning of their weights can be made more user friendly, or perhaps be learned using statistical methods from a set of human-segmented models [CGF09]. We hope to address this in the future.

Acknowledgements: We would like to thank Richard (Hao) Zhang, Frank (Rong) Liu, Alla Sheffer and Vladislav Kraevoy for providing us with the segmentation results of their methods. Also, thank you to Ilya Baran and Jovan Popović for providing their Pinocchio library for automatic character rigging. Meshes were obtained from the Princeton Shape Benchmark, Aim@Shape Project, and 3D Cafe. Research supported by MITACS.

References

[AFS06] ATTENE M., FALCIDIENO B., SPAGNUOLO M.: Hierarchical mesh segmentation based on fitting primitives. *Visual Computer 22*, 3 (2006), 181–193.

[AJ03] AUDET C., JR. J. E. D.: Analysis of generalized pattern searches. *SIAM Journal on Optimization 13*, 3 (2003), 889–903.

[AKM*06] ATTENE M., KATZ S., MORTARA M., PATANE G., SPAGNUOLO M., TAL A.: Mesh segmentation - a comparative study. *IEEE Shape Modeling International* (2006).

[BP07] BARAN I., POPOVIĆ J.: Automatic rigging and animation of 3d characters. *ACM SIGGRAPH* (2007).

[CDST97] CHAZELLE B., DOBKIN D. P., SHOURABOURA N., TAL A.: Strategies for polyhedral surface decomposition: an experimental study. *Computational Geometry: Theory and Applications. 7*, 5-6 (1997), 327–342.

[CGF09] CHEN X., GOLOVINSKIY A., , FUNKHOUSER T.: A benchmark for 3D mesh segmentation. *ACM SIGGRAPH* (2009).

[CSAD04] COHEN-STEINER D., ALLIEZ P., DESBRUN M.: Variational shape approximation. In *ACM SIGGRAPH* (2004).

[dGV08] DE GOES F., GOLDENSTEIN S., VELHO L.: A hierarchical segmentation of articulated bodies. *Eurographics Symposium on Geomtetry Processing* (2008).

[GF08] GOLOVINSKIY A., FUNKHOUSER T.: Randomized cuts for 3D mesh analysis. *ACM SIGGRAPH Asia* (2008).

Summary of objectives used
narrow$(\cdot)$, flat$(\cdot)$, planarsymmetric$(\cdot)$, ellipsoidal$(\cdot)$, perpendicular$(\cdot,\cdot)$, similarsize$(\cdot,\cdot)$, convexparts$(\cdot)$

Model	Labels	Segmentation objectives
Hammer	*handle*, *head*	5∗narrow(*handle*), perpendicular(*handle*, *head*), convexparts(*Seg*)
Quadruped	*head*, *body*, $leg_1, \ldots, leg_4$	$\forall_i$narrow(leg_i), similarsize($leg_1, \ldots, leg_4$), compact(*head*), 10∗convexparts(*Seg*) In dogs, compactness of head is emphasized with 8∗compact(*head*)
Bird	*body*, $wing_1$, $wing_2$, *tail*	narrow(*body*), $\forall_i$flat($wing_i$), similarsize($wing_1, wing_2$), flat(*tail*), … compact(*tail*), 10∗convexparts(*Seg*) Constraints: body and tail lie on plane of global symmetry and parameters for $wing_2$ are reflected from $wing_1$.
Octopus	*head*, $arm_1, \ldots, arm_8$	ellipsoidal(*head*), $\forall_i$narrow(arm_i), similarsize($arm_1, \ldots, arm_8$), 10∗convexparts(*Seg*)
Humanoid	*head*, *torso*, *arm_left* *arm_right*, *leg_left*, *leg_right*	narrow(*arm_left*), narrow(*arm_right*), narrow(*leg_left*), narrow(*leg_right*),… compact(*head*), similarsize(*arm1*, *arm2*), similarsize(*leg1*, *leg2*), 10∗convexparts(*Seg*) Subparts (upper arm, forearm, hand, etc.) are obtained hierarchically using convexity.

Figure 13: *Objectives used to obtain segmentations of each model.*

[HWAG09] HUANG Q., WICKE M., ADAMS B., GUIBAS L.: Shape decomposition using modal analysis. *Eurographics* (2009).

[JKS05] JULIUS D., KRAEVOY V., SHEFFER A.: D-charts: Quasi-developable mesh segmentation. *Eurographics* (2005).

[KFR04] KAZHDAN M., FUNKHOUSER T., RUSINKIEWICZ S.: Symmetry descriptors and 3d shape matching. In *Eurographics Symposium on Geomtetry Processing* (2004).

[KJS07] KREAVOY V., JULIUS D., SHEFFER A.: Model composition from interchangeable components. *Pacific Graphics* (2007).

[KLT05] KATZ S., LEIFMAN G., TAL A.: Mesh segmentation using feature point and core extraction. *The Visual Computer 21*, 8-10 (2005), 649–658.

[KS06] KRAEVOY V., SHEFFER A.: Variational, meaningful shape decomposition. *ACM SIGGRAPH Sketches* (2006).

[KT03] KATZ S., TAL A.: Hierarchical mesh decomposition using fuzzy clustering and cuts. *ACM SIGGRAPH* (2003).

[Kuh55] KUHN H. W.: The hungarian method for the assignment problem. *Naval Research Logistics Quarterly 2*, 1-2 (1955), 83–97.

[LA05] LIEN J.-M., AMATO N. M.: Approximate convex decomposition of polyhedra. *ACM Symposium on Solid and Physical Modeling* (2005).

[LMLR06] LIU S., MARTIN R. R., LANGBEIN F. C., ROSIN P. L.: Segmenting reliefs on triangle meshes. *ACM Symposium on Solid and Physical Modeling* (2006).

[LMLR07] LIU S., MARTIN R. R., LANGBEIN F. C., ROSIN P. L.: Segmenting geometric reliefs from textured background surfaces. *Computer-Aided Design and Applications 4*, 5 (2007).

[LMM98] LUKÁCS G., MARTIN R., MARSHALL D.: Faithful least-squares fitting of spheres, cylinders, cones and tori for reliable segmentation. *ECCV* (1998).

[LZ07] LIU R., ZHANG H.: Mesh segmentation via spectral embedding and contour analysis. *Eurographics* (2007).

[LZSCO09] LIU R. F., ZHANG H., SHAMIR A., COHEN-OR D.: A part-aware surface metric for shape analysis. *Eurographics* (2009).

[MGP06] MITRA N. J., GUIBAS L. J., PAULY M.: Partial and approximate symmetry detection for 3d geometry. *ACM SIGGRAPH* (2006).

[Mic98] MICHALEWICZ Z.: *Genetic Algorithms + Data Structures = Evolution Programs*, 3 ed. Springer, 1998.

[MPS*04] MORTARA M., PATANÈ G., SPAGNUOLO M., FALCIDIENO B., ROSSIGNAC J.: Plumber: a method for a multi-scale decomposition of 3d shapes into tubular primitives and bodies. *ACM Symposium on Solid Modeling and Applications* (2004).

[Mun57] MUNKRES J.: Algorithms for the assignment and transportation problems. *Journal of the Society of Industrial and Applied Mathematics 5*, 1 (1957), 32–38.

[MW99] MANGAN A. P., WHITAKER R. T.: Partitioning 3d surface meshes using watershed segmentation. *IEEE Transactions on Visualization and Computer Graphics 5*, 4 (1999), 308–321.

[OBSC00] OKABE A., BOOTS B., SUGIHARA K., CHIU S. N.: *Spatial Tessellations: Concepts and Applications of Voronoi Diagrams*, 2 ed. Wiley, 2000.

[PSG*06] PODOLAK J., SHILANE P., GOLOVINSKIY A., RUSINKIEWICZ S., FUNKHOUSER T.: A planar-reflective symmetry transform for 3d shapes. *ACM SIGGRAPH* (2006).

[Ric08] RICHARDSON T.: *The Art Student's Guide to the Proportions of the Human Form*. Tom Richardson, 2008.

[SF98] SINGH K., FIUME E.: Wires: a geometric deformation technique. In *ACM SIGGRAPH* (1998).

[Sha08] SHAMIR A.: A survey on mesh segmentation techniques. *Computer Graphics Forum 27*, 6 (2008), 1539–1556.

[SKS06] SIMARI P., KALOGERAKIS V., SINGH K.: Folding meshes: Hierarchical mesh segmentation based on planar symmetry. *Eurographics Symposium on Geomtetry Processing* (2006).

[SS05] SIMARI P., SINGH K.: Extraction and remeshing of ellipsoidal representations from mesh data. *Graphics Interface* (2005).

[SSSCO08] SHALOM S., SHAPIRA L., SHAMIR A., COHEN-OR D.: Part analogies in sets of objects. *Eurographics Workshop on 3D Object Retrieval* (2008).

[Tor97] TORCZON V.: On the convergence of pattern search algorithms. *SIAM Journal on Optimization 7*, 1 (1997), 1–25.

[TW05] THRUN S., WEGBREIT B.: Shape from symmetry. *ICCV* (2005).

[ZTS02] ZUCKERBERGER E., TAL A., SHLAFMAN S.: Polyhedral surface decomposition with applications. *Computers and Graphics 26*, 5 (2002), 733–743.