# Part-Based Segmentation by Skeleton Cut Space Analysis

Cong Feng[1], Andrei C. Jalba[2], and Alexandru C. Telea[1]

[1] Institute Johann Bernoulli, University of Groningen, the Netherlands
c.feng@rug.nl, a.c.telea@rug.nl
[2] Department of Mathmatics and Computer Science, TU Eindhoven, the Netherlands
a.c.jalba@tue.nl

**Abstract.** We present a new method for part-based segmentation of voxel shapes that uses medial surfaces to define a segmenting cut at each medial voxel. The cut has several desirable properties – smoothness, tightness, and orientation with respect to the shape's local symmetry axis, making it a good segmentation tool. We next analyze the space of all cuts created for a given shape and detect cuts which are good segment borders. Our method is robust to noise, pose invariant, independent on the shape geometry and genus, and is simple to implement. We demonstrate our method on a wide selection of 3D shapes.

**Keywords:** Part-based segmentation, medial surfaces, skeletonization

## 1 Introduction

Shape segmentation aims to decompose a 3D shape into a set of parts that obey certain application-related properties, and is used in many contexts such as image analysis, registration, and 3D modeling [27]. *Patch-based* segmentation detects quasi-flat segments whose borders follow local curvature maxima on the shape surface, and is most used for faceted shapes [24]. *Part-based* segmentation follows a semantics-oriented approach, aiming to find shape parts that one would intuitively perceive as being logically distinct, and is used for natural shapes [22].

For a shape $\Omega \subset \mathbb{R}^3$, part-based segmentations (PBS) using *partitioning cuts* create a set of cuts $c \subset \partial\Omega$ that divide the shape boundary $\partial\Omega$ into disjoint parts. Desirable PBS properties, *e.g.* smoothness, orientation, tightness, and position of cuts that create segments, can be stated in terms of the cut-set $\mathcal{B} = \{c\}$. Finding a good segmentation is thus mapped to finding a cut-set $\mathcal{B}$ having such properties, a hard problem due to the high dimensionality of the cut space.

We present a new way to produce PBS of 3D voxel shapes by *skeleton cuts*. First, we construct, at any shape point, a cut that is locally and globally smooth, tightly wraps around the surface, is self-intersection free, and is locally orthogonal to the shape's local symmetry axis. For this, we use the shape's medial surface. Next, we construct the cut-space $\mathcal{S} \subset \partial\Omega$ that contains all such cuts for a given shape. We extract the cut-set $\mathcal{B} \subset \mathcal{S}$ yielding our PBS by analyzing the global distribution of cut properties over $\mathcal{S}$. We demonstrate our method on a variety of 3D shapes and compare our results with eight existing PBS methods.

Section 2 reviews related work. Section 3 presents our method. Section 4 illustrates our method on a wide variety of 3D shapes and also compares it with related methods. Section 5 discusses our method. Section 6 concludes the paper.

## 2 Related Work

Two main shape segmentation types exist [1, 4, 28]: *Patch-based* methods segment a shape's surface into quasi-flat patches bounded by sharp surface creases, and are suitable for synthetic shapes. *Part-based* segmentation (PBS), our focus, partitions a shape's surface into its logical components, useful for shapes consisting of articulated parts, like human bodies, plants, and other natural structures.

Most PBS methods define segments along what a human would see as logical shape parts, in two steps: (a) find *where* to cut the shape to isolate a part; and (b) find *how* to build a cut, once its location is set. These steps are addressed in different ways. As the topology of the shape skeleton or medial axis matches the part-whole shape structure [31], many methods use medial axes to place cuts. Au *et al.* use curve skeletons [5], where each skeleton branch maps to a part. Cuts are created by optimizing for cut concavity and length via minimal cuts [12]. Golovinskiy *et al.* create a large randomized cut-set and find part borders as the cuts on which most surface edges lie [10]. Shapira *et al.* note that skeletonization and segmentation are complementary, and segment a shape by computing a scalar shape-diameter function (SDF) on its surface, finding segments by clustering shape faces with similar SDF values [29]. Tierny *et al.* segment shapes hierarchically by topological and geometrical analysis of their structure using Reeb graphs, which are similar to curve skeletons [33]. Chang *et al.* segment shapes by computing their medial surfaces, separating their different manifolds, and back projecting each manifold on the shape surface to yield a segment [6]. Dey and Sun extract curve skeletons as the maxima of the so-called medial geodesic function (MGF) which encodes the length of the shortest path between feature points of each point in the shape [8]. They next segment tubular parts as those which minimize the eccentricity of such paths. Reniers *et al.* construct a part for each branch of the shape's curve skeleton [22]. Part borders correspond to curve-skeleton junction points, and are created by the same shortest paths as in [8]. Such paths are piecewise smooth and locally orthogonal to the curve skeleton, and thus are good PBS borders. Creating a part for each curve-skeleton branch is however delicate, as curve skeletons can contain many spurious junctions which change widely when the shape is slightly perturbed. Reniers *et al.* alleviate this by heuristics that shift cut-points along the curve skeleton to optimize for cut stability and planarity [23]. Yet, this method cannot segment shapes having large geometric, but little topological, variability. For example, consider a pawn chess piece: Its curve skeleton has no junction points, so, [23] cannot separate the pawn's head, body, and base, although these have different thicknesses.

Summarizing, the two elements of a good PBS (*where* to cut, and *how* to cut) are targeted in complement by different methods: Skeleton-based methods construct good partitioning cuts efficiently, *e.g.* by shortest-paths [8, 23]. Yet, curve

skeletons do not encode enough of the shape geometry. Global search methods that analyze a wide set of shape cuts offer good ways to select where to partition [10, 29]. Yet, they do not offer explicit constraints for the cut shapes, and exhaustive cut-space search is expensive. Our method combines the advantages of the two above classes of methods, while minimizing their limitations.

## 3 Method

Our method has a simple intuition: Say that we want to cut the shape in Fig. 1 a at, or close to, points $A \dots E$. Which properties should the ensuing cuts have, apart from their location (to be discussed separately next), to yield a 'natural' PBS? In other words: How would a human draw such cuts? Figure 1 a shows five *undesirable* cuts: $A$ is noisy, although it crosses a perfectly smooth surface zone; $B$ is self-intersecting; $C$ and $D$ are too loose (long); and $E$ is unnaturally slanted – a human asked to cut the shape at that point would arguably do it so across the finger's symmetry axis. Figure 1 b shows five cuts for the same points, computed with the method in this paper. We argue that these cuts are more suitable for PBS than those in Fig. 1 a, as they are (1) tight, (2) locally smooth, (3) self-intersection free, (4) and locally orthogonal to the shape's symmetry axis. An additional property that cuts should satisfy is (5) being closed curves, so that they divide the shape's surface into different parts. We construct such cuts as follows: First, we compute a simplified medial surface of the input shape (Sec. 3.1). For each medial point, we next construct a cut having the above properties (Sec. 3.2). This answers the question "how to cut". By analyzing the resulting cut-space, we next select a small cut-set that gives us the borders of salient shape-parts (Sec. 3.3). This answers the question "where to cut".

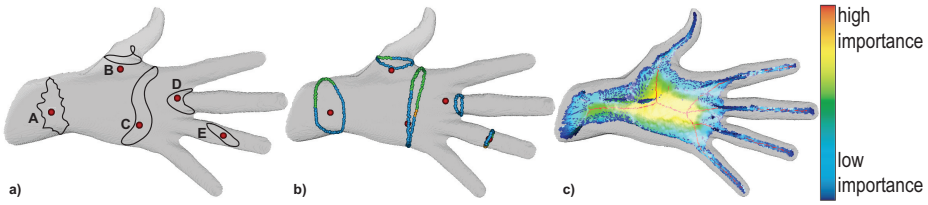


**Fig. 1.** Possible cuts for part-based segmentation. Suboptimal cuts (a). Cuts created by our method (b). Medial surface colored by its importance metric (c).

### 3.1 Skeletonization

The Euclidean distance transform $DT_{\partial\Omega} : \Omega \to \mathbb{R}_+$ of a shape $\Omega \subset \mathbb{Z}^3$ with boundary $\partial\Omega$ is

$$DT_{\partial\Omega}(\mathbf{x} \in \Omega) = \min_{\mathbf{y} \in \partial\Omega} \|\mathbf{x} - \mathbf{y}\|. \quad (1)$$

The medial surface, or surface skeleton, of $\partial\Omega$ is defined as

$$S_{\partial\Omega} = \{\mathbf{x} \in \Omega | \exists \{\mathbf{f}_1, \mathbf{f}_2\} \subset \partial\Omega, \mathbf{f}_1 \neq \mathbf{f}_2, \|\mathbf{x} - \mathbf{f}_1\| = \|\mathbf{x} - \mathbf{f}_2\| = DT_{\partial\Omega}(\mathbf{x})\} \quad (2)$$

where $\mathbf{f}_1$ and $\mathbf{f}_2$ are the contact points with $\partial\Omega$ of the maximally inscribed ball in $\Omega$ centered at $\mathbf{x}$ [9, 25], also called feature points [26, 31]. These define the feature transform $FT_{\partial\Omega} : \Omega \to \mathcal{P}(\partial\Omega)$

$$FT_{\partial\Omega}(\mathbf{x} \in \Omega) = \underset{\mathbf{y}\in\partial\Omega}{\operatorname{argmin}} \|\mathbf{x} - \mathbf{y}\|. \tag{3}$$

Medial surfaces are sensitive to small-scale noise on the input shape, especially when using voxel-based models. To alleviate this, medial surfaces can be *regularized* by a metric $\rho : S_{\partial\Omega} \to \mathbb{R}_+$ which marks important medial points by high values, and medial points capturing shape details by low values respectively. One such metric is the medial geodesic function (MGF) which sets $\rho(\mathbf{x})$ to the length of the shortest path on $\partial\Omega$ between the two feature points of $\mathbf{x}$ [8]. As the MGF monotonically increases from the medial surface boundary to its center, upper thresholding it yields connected and noise-free simplified medial surfaces (though tunnel preservation requires additional work),[25]. Figure 1 c shows a regularized medial surface using the method in [25].

### 3.2 Cut model

The first step of our PBS is to compute a rich set of cuts, or cut space $\mathcal{S}$, which could all be segment borders, *i.e.* satisfy properties (1-5) listed at the beginning of Sec. 3. To build a cut $c \in \mathcal{S}$, consider a point $\mathbf{x} \in S_{\partial\Omega}$. By definition, $\mathbf{x}$ has at least two feature points $\mathbf{f}_1$ and $\mathbf{f}_2$ on $\partial\Omega$ (Eqn. 2). Consider, for now, that there are precisely two such points. We first trace the shortest path $\gamma_1 \subset \partial\Omega$ between $\mathbf{f}_1$ and $\mathbf{f}_2$ (Fig. 2 a). The length of $\gamma_1$ is precisely the MGF value for $\mathbf{x}$ (Sec. 3.1). Secondly, we find the midpoint $\mathbf{m}$ of $\gamma_1$, *i.e.* the voxel of $\gamma_1$ located furthest in arc-length distance from both $\mathbf{f}_1$ and $\mathbf{f}_2$. We then trace a ray passing through $\mathbf{x}$ and oriented in the direction $\mathbf{x} - \mathbf{m}$, and find the point $\mathbf{o}$ where this ray 'exits' $\Omega$ (Fig. 2 b). Intuitively, $\mathbf{o}$ is a point on the 'other side' of $S_{\partial\Omega}$ as opposed to $\mathbf{m}$. Finally, we construct the two shortest paths on $\partial\Omega$ connecting $(\mathbf{f}_1, \mathbf{o})$ and $(\mathbf{f}_2, \mathbf{o})$ respectively (Fig. 2 c,d). Our final cut $c$ for point $\mathbf{x}$ is given by $\gamma_1 \cup \gamma_2 \cup \gamma_3$.

While $c$ is piecewise geodesic (so locally smooth), it can be non-smooth at the three endpoints $\mathbf{f}_1, \mathbf{f}_2$ and $\mathbf{o}$ of $\gamma_i$. Also, our construction does not make $c$ as tight as possible globally. To fix both issues, we perform 5 iterations of a constrained Laplacian smoothing pass over $c$, with a kernel size of 10 voxels. We prevent $c$ leaving the surface, by reprojecting its voxels to their closest points on $\partial\Omega$ after each iteration. This smooths out possible 'kinks' at $\mathbf{f}_1, \mathbf{f}_2$ and $\mathbf{o}$, thus making $c$ globally smooth and tight. If such kinks are very small or inexistent, smoothing has no effect, as $c$ is globally geodesic. In that case, Laplacian smoothing shifts $c$'s points along the surface normal, since $c$'s acceleration $c''$ is normal to the surface, so reprojection moves the smoothed points back to their original location.

**Cut properties:** Our cuts meet the desired properties we require for PBS:

1. *tight:* Cut parts $\gamma_i$ are piecewise-geodesic, thus shortest curves on $\partial\Omega$. Also, the constrained Laplacian smoothing shortens potential kinks present at the geodesic endpoints, thus making the entire $c$ wrap tightly around the shape;

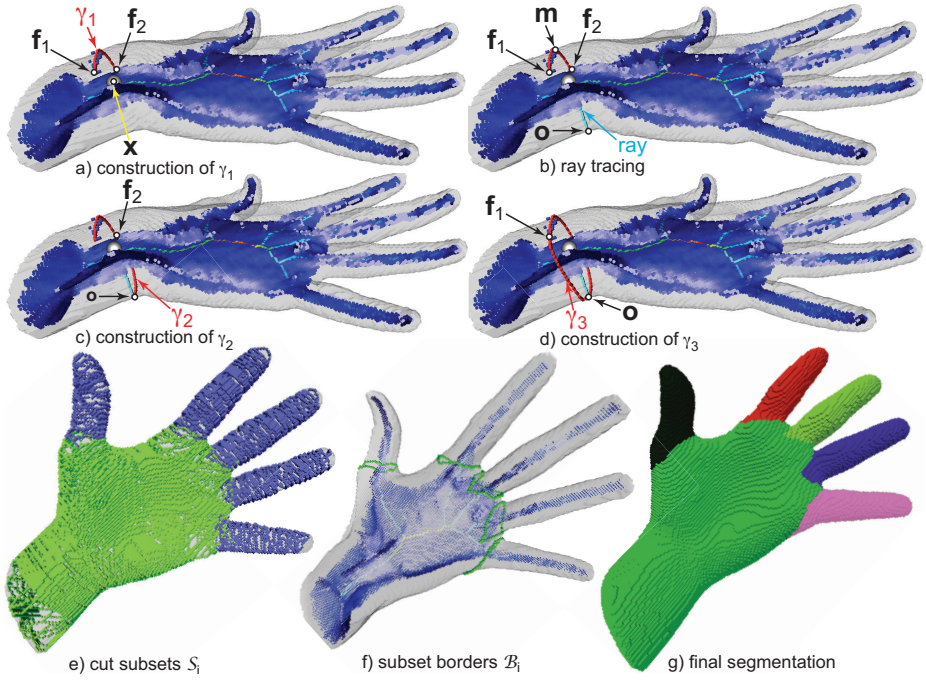


**Fig. 2.** Cut construction (a-d) and cut-space analysis (e-g) for part-based segmentation.

2. *smooth:* Guaranteed by the same properties as for tightness – piecewise geodesicness and constrained Laplacian smoothing;
3. *self-intersection free:* $c$ is a geodesic triangle (three geodesics linking three *different* points on $\partial\Omega$) whose edges do not intersect except at endpoints;
4. *locally orthogonal* to the symmetry axis: The cut $c(\mathbf{x})$ surrounds the medial surface $S_{\partial\Omega}$ around point $\mathbf{x}$, by construction. Hence, it also surrounds the so-called *curve skeleton* of $\partial\Omega$, which is a 1D structure locally centered within $S_{\partial\Omega}$ with respect to its boundary $\partial S_{\partial\Omega}$. While we do not have a formal proof of local orthogonality, we observed in practice that our construction always creates cuts that are visually orthogonal to the curve skeleton;
5. *closed:* The cut $c$ is a closed (Jordan) curve by construction.

**Implementation:** To build $\gamma_1$, we need two feature points $\mathbf{f}_1$ and $\mathbf{f}_2$. Two issues exist here: (1) Computing the feature transform $FT(\mathbf{x})$ on digital shapes cannot be done via Eqn. 3, given the finite voxel grid resolution [21, 25]. To fix this, we compute the so-called extended feature transform $EFT(\mathbf{x})$ which finds all closest-points on $\partial\Omega$ to all 26 neighbors of $\mathbf{x}$, and which is a superset of $FT(\mathbf{x})$ [25]. From this superset, we select exactly two feature points that best represent the symmetric embedding of $S_{\partial\Omega}$ in $\Omega$. For this, we select the two feature points $\{\mathbf{f}_1, \mathbf{f}_2\} \subset EFT(\mathbf{x})$ that maximize the angle $\widehat{\mathbf{f}_1\mathbf{x}\mathbf{f}_2}$. We trace the ray used to find $\mathbf{o}$ by Bresenham's 3D line-tracing algorithm on the voxel shape. We compute geodesics by Dijkstra's shortest-path algorithm on the connectivity

graph of voxels of $\partial\Omega$, using $A^*$ heuristics to speed the search, and using edge weights that approximate neighbor-voxel distances by Eppstein's scheme [13] for better path-length accuracy. Finally, we reproject Laplacian-smoothed points on the shape surface by using the fast ANN library for finding nearest-neighbors [19].

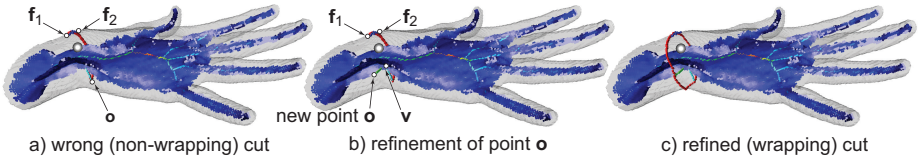


**Fig. 3.** Refinement of cut construction.

In a very few cases, point **o** found as above does not lie on the opposite side of the medial surface with respect to point **m**, so the resulting cut will not wrap around the medial surface (Fig. 3 a). When this happens, we trace a ray in the direction $\mathbf{f}_1 - \mathbf{f}_2$ from the midpoint **v** of the current ray, and set **o** to the voxel where this new ray exits $\Omega$ (Fig. 3 b). If the new point **o** still does not yield the desired wrapping cut, we repeat the above refinement (Fig. 3 c). We verified that this refinement produces cuts wrapping around the medial surface for all our test shapes within 3 up to 4 refinement steps.

### 3.3 Cut space analysis

We can create a cut $c(\mathbf{x})$ for any voxel $\mathbf{x}$ of a shape's medial surface $S_{\partial\Omega}$, which has good properties for PBS. Intuitively, $c(\mathbf{x})$ is a good way to cut the shape at point $\mathbf{x}$, *if* we want a cut there. We now must decide *where* we want to cut to get a PBS with desired global properties. Let $\mathcal{S} = \{c(\mathbf{x}) | \mathbf{x} \in S_{\partial\Omega}\}$ be the space of all cuts created from $S_{\partial\Omega}$. Given our cut properties, cuts on the same shape-part share similar properties *e.g.* orientation and length. Cuts for different parts have different properties. Consider our hand model: Finger cuts are short; wrist cuts have average length; and palm cuts are longest. For a shape consisting of a rump and protruding parts, cuts for parts are shorter than cuts for the rump.

We use these insights to partition $\mathcal{S}$ in several subsets $\mathcal{S}_i$ so that $\cup_i \mathcal{S}_i = \mathcal{S}$ and $\mathcal{S}_i \cup \mathcal{S}_{j\neq i} = \varnothing$. For this, we compute the histogram of cut lengths over $\mathcal{S}$. Histogram peaks show large similar-length cuts. If we partition the histogram by placing thresholds in the valleys between peaks, we get our desired subsets $\mathcal{S}_i$. To find such thresholds robustly, we filter the histogram by the mean shift method [7]. This 'sharpens' the cut-distribution and separates peaks from valleys more clearly. We define a peak as a histogram value exceeding $\lambda$ times the cut count $\|\mathcal{S}\|$, and a valley as a value less than $\mu = \lambda/3$. Setting $\lambda = 0.01$ gave good results for all shapes in this paper. Figure 4 shows the cut-length histogram for the hand model. Its three main peaks describe cuts on the fingers, wrist, and palm; the two valleys give the two thresholds needed to separate fingers from the palm and the palm from the wrist.

Subsets $\mathcal{S}_i$ do not (yet) coincide with our desired segments. Indeed, an $\mathcal{S}_i$ can contain logically disjoint cuts of similar lengths – *e.g.* all cuts on the fingers

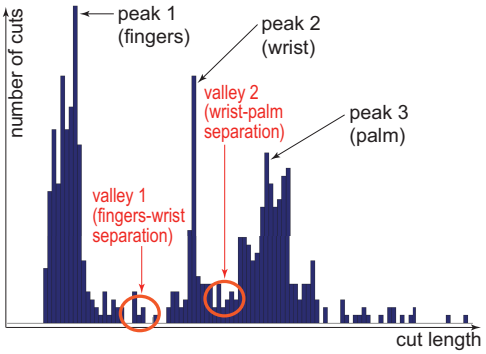


**Fig. 4.** Cut-length histogram for hand model (Sec. 3.3).

(blue in Fig. 2 e) are in the same subset. Also, $\mathcal{S}$ does not fully cover $\partial\Omega$, since we compute it from the simplified medial surface. This is shown by the gaps between cuts in Fig. 2 d. To fix this, we first define a cut $c(\mathbf{x})$ as being a border $\mathcal{B}_i$ of subset $\mathcal{S}_i$ if $c(\mathbf{x})$ belongs to a different subset than any of the cuts $c(\mathbf{y})$, where $\mathbf{y}$ are the 26-neighbors of $\mathbf{x}$ on $S_{\partial\Omega}$. Using this definition, we find the set of cuts $\{\mathcal{B}_i\}$ that represent the borders of our final segments (Fig. 2 f). Note that, if a cut is marked as border, at least one of its neighbor cuts will be in a different cut subset, by definition. Hence, that neighbor cut will also be a border, so more than one border will be produced from a $3^2$ voxel neighborhood. To remove such duplicates, we keep, for each such neighborhood, the shortest border. We compute our final segments by finding the connected components of $\partial\Omega$ separated by borders, via a simple flood-fill algorithm on $\partial\Omega$ (Fig. 2 g).

## 4 Results and Comparison

We have tested our method on several shapes provided as 3D polygon meshes, voxelized by *binvox* [20] at resolutions up to $400^3$ voxels. Figure 5 compares our results with [23], the best medial-descriptor voxel PBS method we know. We get very similar results, but find more fine-grained segments than [23] – see finger and ear details of the animal models, pig tail, dragon spikes, and microscope lens. Segment borders are smooth and locally orthogonal to the shape's symmetry axis, *i.e.*, similar to how a human would cut the shape at the respective places. Our method finds segments of various sizes, ranging from details (dragon's tail, hound's ears), to large parts (limbs of various models). Figure 6 a-k compares our method with eight PBS methods on two shapes [3, 14–17, 22, 23, 33]. Here, Reniers *et al.* (1) denotes [22], and Reniers *et al.* (2) denotes [23]. These methods span from voxel-based to mesh-based, and use various segmentation heuristics (skeleton, curvature, salience, and topology-based). We argue that our method creates equally or, in some cases, more plausible PBSs. Since both our method and [23] use medial descriptors, computed by the same underlying method [25], a relevant question is how the two methods differ. We use (a) medial *surfaces*, while [23] uses *curve* skeletons; and (b) we find segment borders by analyzing *all*

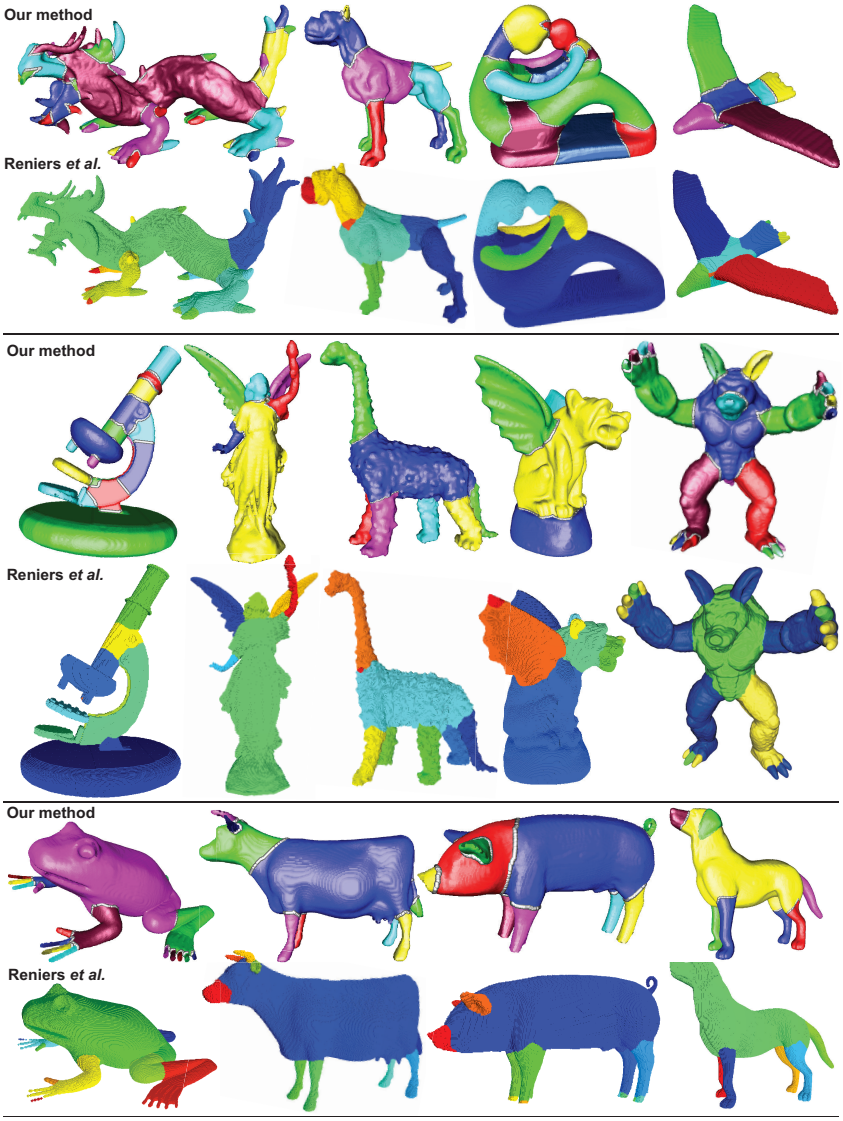


**Fig. 5.** Part-based segmentations of our method *vs* Reniers *et al.* [23] (Sec. 4).

possible cuts, while [23] places such borders around the curve-skeleton branch junctions. Fig. 6 l-p shows five examples where the public implementation of [23] fails to segment at all. We find two causes for this: The shape parts in Fig. 6 l cannot be well described by curve-skeleton branches, as they are nearly rotationally symmetric. As few (if any) such junctions exist, [23] fails. The shape in Fig. 6 n is described by a mix of medial surfaces (base plate) and curve skeletons (tubular parts). As [23] only uses curve skeletons, data on the base plate is incomplete or missing. For the shapes in Fig. 6 m-p, the many heuristics in [23]

to select cuts centered on the curve-skeleton fail, as they imply that such cuts should be nearly planar. This does not happen for the above shapes.

We can also produce a *multiscale* PBS: For this, we simply change the values of $\lambda$ and $\mu$ used to partition the cut space via its length histogram (Sec. 3.3). High $\lambda$ values and low $\mu$ values yield fewer and more differentiated segments (in terms of local thickness); closer values of *lambda* and $\mu$ yield a finer-grained segmentation. Figure 6 r shows three such scales for the armadillo shape.

Our method is *pose invariant*, as illustrated in Fig. 6 s. Indeed, our cut space histogram captures local shape thickness, which does not depend on pose.

We coded our method in C++ and ran it on an 8-core 3.5 GHz 32 GB RAM Linux PC. Table 1 shows the time for computing cuts ($t_{cuts}$), medial surface ($t_{skel}$), cut-space analysis ($t_{space}$), total time for our method ($t_{total}$), and total time for [23] ($t_{Reniers}$). Empty cells in $t_{Reniers}$ show shapes where [23] failed. As cuts are computed independently, we parallelized our implementation using *pthreads*. Compared to a serial implementation, this is roughly 7 times faster, close to the optimal value of 8 implied by our hardware. Compared to [23], on the same hardware, we are slightly faster in most cases. More importantly, we could successfully segment all tested shapes, unlike [23].

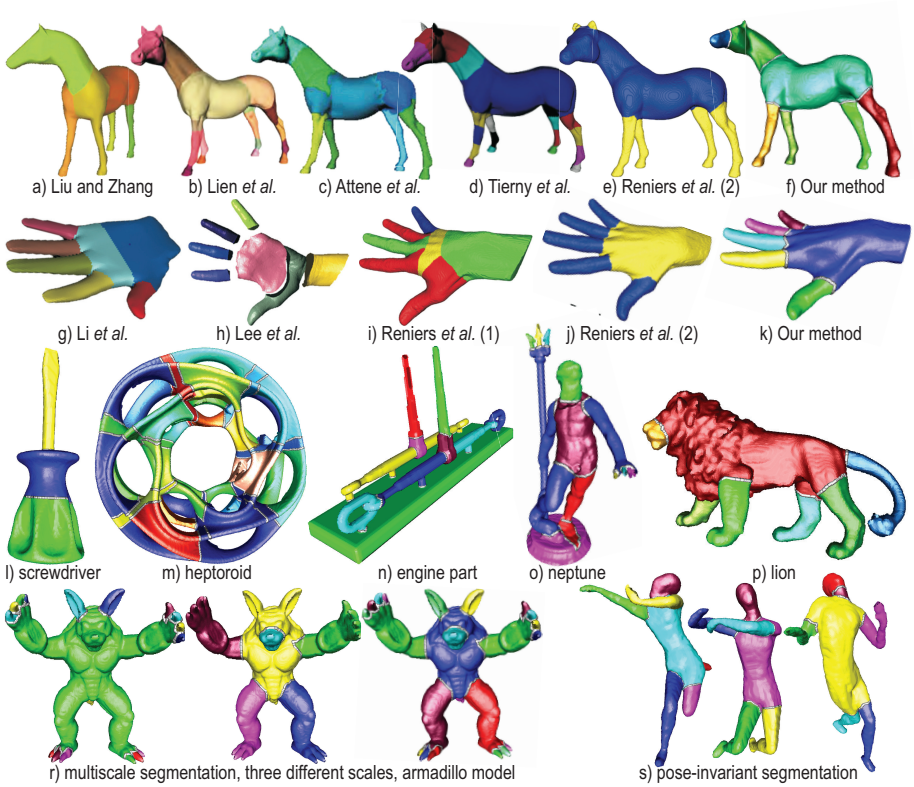


**Fig. 6.** Comparison of our method with eight PBS methods (a-k). Our results for shapes where Reniers *et al.* fails (l-p). Multiscale (r) and pose-invariant (s) segmentations.

| **Shapes** | **cuts** $\|\mathcal{S}\|$ | **voxels** $\|\Omega\|$ | **voxel volume** | $t_{cuts}$ | $t_{skel}$ | $t_{part}$ | $t_{total}$ | $t_{Reniers}$ |
|---|---|---|---|---|---|---|---|---|
| Dragon | 2789 | 283238 | 400*400*400 | 50.8 | 1.90 | 0.03 | 52.73 | 40.26 |
| Hound | 1530 | 245759 | 300*300*300 | 23.24 | 1.51 | 0.01 | 24.76 | 25.1 |
| Hyptoroid | 4873 | 651478 | 400*400*400 | 400.5 | 3.36 | 0.04 | 403.90 | - |
| Fertility | 1354 | 199581 | 300*300*300 | 20.85 | 2.02 | 0.01 | 22.88 | 22.89 |
| Gargoyle | 488 | 129420 | 300*300*300 | 12.62 | 3.26 | 0.005 | 15.885 | 69.89 |
| Microscope | 1397 | 307863 | 300*300*300 | 44.14 | 1.58 | 0.01 | 45.73 | 198.02 |
| Lucy | 6201 | $1.04 \times 10^6$ | 300*300*300 | 68.01 | 0.63 | 0.09 | 68.73 | 12.7 |
| Engine part | 1501 | 135416 | 300*300*300 | 15.55 | 0.27 | 0.01 | 15.83 | - |
| Frog | 41450 | $1.20 \times 10^7$ | 300*300*300 | 808.2 | 2.48 | 2.16 | 812.8 | 36.93 |
| Screwdriver | 1372 | 306480 | 300*300*300 | 13.14 | 0.60 | 0.01 | 13.75 | - |
| Noisydino | 1375 | 194117 | 300*300*300 | 14.79 | 1.19 | 0.015 | 16.00 | 20.2 |
| Cow | 1009 | 143938 | 256*256*256 | 8.15 | 0.96 | 0.01 | 9.12 | 14.34 |
| Neptune | 1908 | 211723 | 420*185*251 | 34.7 | 1.22 | 0.02 | 35.94 | - |
| Airplane | 741 | 76700 | 300*300*300 | 6.00 | 0.28 | 0.08 | 6.37 | - |
| Bird | 476 | 45638 | 300*300*300 | 2.28 | 0.18 | 0.003 | 2.47 | 7.98 |
| Hand | 584 | 58071 | 200*84*140 | 2.15 | 0.22 | 0.004 | 2.37 | - |
| Lion | 2181 | 381968 | 300*300*300 | 23.16 | 1.08 | 0.02 | 24.27 | - |
| Horse | 884 | 109555 | 142*300*251 | 9.58 | 1.24 | 0.008 | 10.83 | - |
| Pig | 959 | 145215 | 300*300*300 | 10.97 | 1.51 | 0.01 | 12.50 | 22.26 |
| Dog | 1241 | 184805 | 300*300*300 | 15.65 | 1.29 | 0.02 | 16.97 | 18.87 |
| Hippo | 838 | 166932 | 300*300*300 | 12.13 | 2.41 | 0.01 | 14.55 | 25.18 |
| Rhino | 1746 | 403399 | 300*300*300 | 25.20 | 2.15 | 0.03 | 27.39 | - |
| Armadillo | 2242 | 436933 | 300*229*252 | 47.55 | 2.67 | 0.03 | 50.26 | - |

**Table 1.** Shape sizes and segmentation times by our method and Reniers *et al.* [23].

## 5 Discussion

We next discuss several aspects of our proposed part-based segmentation method.

**Global search:** We create a PBS by finding all part-inducing cuts from the medial surface, and selecting a cut-subset by globally optimizing for part-similarity as captured by cut lengths. In contrast to purely topological PBS methods [22, 23], we search a much wider space of possible partitionings; yet, our search space is much smaller than that of other methods which look for cuts of any possible orientation [10], thereby achieving a good flexibility-performance balance.

**Simplicity:** We compute the medial surface, generate all cuts from its voxels, and cluster these based on their length-histogram. *Any* medial surface skeletonization method can be used here, *e.g.* [2, 26, 30], as long as it outputs regularized skeletons. This makes our method immediately applicable to mesh-based shape representations (and their medial surfaces) [11].

**Regularization:** We use regularized medial surfaces (Sec. 3.1), having voxels with large MGF importance values, which have far-apart feature points $\mathbf{f}_1$ and $\mathbf{f}_2$. This ensures that the ray casting used to compute our cuts can robustly find cuts that wrap around the medial surface (Sec. 3.2).

**Multiscale:** Multiscale PBS occurs at two levels: (1) Simplified medial surfaces yield cuts only for important shape parts; (2) The cut histogram analysis parameters $\lambda$ and $\mu$ select the level-of-detail where we search for cut-length differences.

**Pose invariance:** Our method is pose-invariant [23, 32], as shown by the model in Fig. 6 s (which is also used in [32] to show pose invariance).

**Robustness:** We robustly segment noisy or detail-rich surfaces, *e.g. dragon* and *dino* (Fig. 5) or *lion* (Fig. 6). Segment borders are smooth by construction (Sec. 3.2). Since our segmentation uses a subset of these cuts, and only considers *integral* cut properties (length) rather than differential ones (*e.g.* curvature),

noise and/or small-scale details are robustly handled.

**Limitations:** Our method's cost is $O(\|S_{\partial\Omega}\|\|\partial\Omega\|\log\|\partial\Omega\|)$. As our method parallelizes easily (Sec. 4), its practical cost is similar to other skeleton-based PBS method [22, 23] or cut-based methods [10]. For space constraints, we compare with only eight related methods. More PBS methods exist, and quantitative metrics can be further used to measure segmentation quality [18]. Yet, even without such extra insights, we argue that our goal of showing that *surface* skeletons have added value for PBS as opposed to *curve* skeletons is well defended.

## 6 Conclusions

We have presented a new method for part-based segmentation of 3D voxel shapes by analyzing the entire space of potential partitioning cuts constructed by using the shape's medial surface. To our knowledge, our approach is the first which uses medial surfaces for part-based segmentation, and thereby shows the added-value of medial surfaces for segmentation, as opposed to the well-known use of curve skeletons for the same task. We demonstrate our method on a wide variety of 3D shapes, and compare it with eight related segmentation methods.

Our approach opens many directions. Different ways to partition the cut space can be easily tried, *e.g.* bottom-up hierarchical clustering of cut similarities based on *e.g.* curvature, eccentricity, and orientation. This would lead to an entire family of PBS methods having a single simple implementation. Our cut-length histogram could deliver an effective shape descriptor for retrieval and matching. Finally, implementing our segmentation method for mesh-based shapes and/or on the GPU should lead to massive scalability increases.

## References

[1] A. Agates, I. Pratikakis, S. Perantonis, N. Sapidis, and P. Azariadis. 3D mesh segmentation methodologies for CAD applications. *Computer-Aided Design & Applications*, 4(6):827–841, 2007.

[2] C. Arcelli, G. Sanniti di Baja, and L. Serino. Distance-driven skeletonization in voxel images. *IEEE TPAMI*, 33(4):709–720, 2011.

[3] M. Attene, B. Falcidieno, and M. Spagnuolo. Hierarchical mesh segmentation based on fitting primitives. *Visual Comput*, 22(3):181–193, 2006.

[4] M. Attene, S. Katz, M. Mortara, G. Patane, M. Spagnuolo, and A. Tal. Mesh segmentation - a comparative study. In *Proc. SMI*, pages 134–141, 2006.

[5] O. Au, C. Tai, H. Chu, D. Cohen-Or, and T. Lee. Skeleton extraction by mesh contraction. *ACM TOG (Proc. ACM SIGGRAPH)*, 27(3):441–449, 2008.

[6] M. Chang, F. Leymarie, and B. Kimia. Surface reconstruction from point clouds by transforming the medial scaffold. *CVIU*, 113(11):1130–1146, 2009.

[7] D. Comaniciu and P. Meer. Mean shift: A robust approach toward feature space analysis. *IEEE TPAMI*, 24(5):603–619, 2002.

[8] T. Dey and J. Sun. Defining and computing curve-skeletons with the medial geodesic function. In *Proc. SGP*, pages 143–152, 2006.

[9] P. Giblin and B. Kimia. A formal classification of 3D medial axis points and their local geometry. *IEEE TPAMI*, 26(2):238–251, 2004.

[10] A. Golovinskiy and T. Funkhouser. Randomized cuts for 3D mesh analysis. *ACM TOG*, 27(5):454–463, 2008.
[11] A. Jalba, J. Kustra, and A. Telea. Surface and curve skeletonization of large 3D models on the GPU. *IEEE TPAMI*, 35(6):1495–1508, 2013.
[12] S. Katz, G. Leifman, and A. Tal. Mesh segmentation using feature point and core extraction. *Visual Comput*, 21(8):649–658, 2005.
[13] N. Kiryati and G. Szekely. Estimating shortest paths and minimal distances on digitized three-dimensional surfaces. *Pattern Recognition*, 26:1623–1637, 1993.
[14] Y. Lee, S. Lee, A. Shamir, D. Cohen-Or, and H. P. Seidel. Mesh scissoring with minima rule and part salience. *CAGD*, 22:444–465, 2005.
[15] X. Li, T. Woon, T. Tan, and Z. Huang. Decomposing polygon meshes for interactive applications. In *Proc. I3D*, pages 35–42, 2001.
[16] J. Lien, J. Keyser, and N. Amato. Simultaneous shape decomposition and skeletonization. In *Proc. ACM SPM*, pages 219–228, 2005.
[17] R. Liu and H. Zhang. Segmentation of 3D meshes through spectral clustering. In *Proc. Pacific Graphics*, pages 298–305, 2004.
[18] Z. Liu, S. Tang, S. Bu, and H. Zhang. New evaluation metrics for mesh segmentation. *Computers & Graphics*, 37(6):553–564, 2013.
[19] D. Mount and S. Arya. Approximate nearest-neighbor search, 2015. `www.cs.umd.edu/~mount/ANN`.
[20] F. Nooruddin and G. Turk. Simplification and repair of polygonal models using volumetric techniques. *IEEE TVCG*, 9(2), 2003.
[21] D. Reniers and A. Telea. Tolerance-based feature transforms. In *Advances in Computer Graphics and Computer Vision*, pages 187–200. Springer, 2007.
[22] D. Reniers and A. Telea. Hierarchical part-type segmentation using voxel-based curve skeletons. *Visual Comput*, 24(6):383–395, 2008.
[23] D. Reniers and A. Telea. Part-type segmentation of articulated voxel-shapes using the junction rule. *CGF*, 27(7):1845–1852, 2008.
[24] D. Reniers and A. Telea. Patch-type segmentation of voxel shapes using simplified surface skeletons. *CGF*, 27(7):1837–1844, 2008.
[25] D. Reniers, J. J. van Wijk, and A. Telea. Computing multiscale skeletons of genus 0 objects using a global importance measure. *IEEE TVCG*, 14(2):355–368, 2008.
[26] J. Roerdink and W. Hesselink. Euclidean skeletons of digital image and volume data in linear time by the integer medial axis transform. *IEEE TPAMI*, 30(12):2204–2217, 2008.
[27] A. Shamir. A formulation of boundary mesh segmentation. In *Proc. 3DPVT*, 2004.
[28] A.l Shamir. A survey on mesh segmentation techniques. *CGF*, 27(8):1539–1556, 2008.
[29] L. Shapira, A. Shamir, and D. Cohen-Or. Consistent mesh partitioning and skeletonisation using the shape diameter function. *Visual Comput*, 24(4):249–259, 2008.
[30] K. Siddiqi, S. Bouix, A. Tannenbaum, and S. Zucker. Hamilton-Jacobi skeletons. *IJCV*, 48(3):215–231, 2002.
[31] K. Siddiqi and S. Pizer. *Medial representations: mathematics, algorithms and applications*. Springer, 2008.
[32] K. Siddiqi, J. Zhang, D. Macrini, A. Shoukofandeh, and S. Dickinson. Retrieving articulated 3D models using medial surfaces. *Mach. Vis. Appl.*, 19:261–275, 2008.
[33] J. Tierny, J. Vandeborre, and M. Daoudi. Topology driven 3D mesh hierarchical segmentation. In *Proc. SMI*, pages 215–220, 2007.