The Visual Computer manuscript No.
(will be inserted by the editor)

Dennie Reniers · Alexandru Telea

Hierarchical Part-type Segmentation using Voxel-based Curve Skeletons

Abstract We present an effective framework for segmenting 3D shapes into meaningful components using the curve skeleton. Our algorithm identifies a number of critical points on the efficiently-computed curve-skeleton, either fully automatically as the junctions of the curve skeleton, or based on user input. We use these points to construct a partitioning of the object surface using geodesics. Because the segmentation is based on the curve skeleton, it intrinsically reflects the shape symmetry and articulation, and can handle shapes with tunnels. We describe a voxel-based implementation of our method which is robust and noise resistant, able to handle shapes of complex articulation and topology, produces smooth segment borders, and delivers hierarchical level-of-detail segmentations. We demonstrate the framework on various real-world 3D shapes. Additionally, we discuss the use of both curve and surface skeletons to produce part-type, respectively patch-type, segmentations of 3D shapes.

Keywords shape segmentation, curve skeletons

1 Introduction

Shape segmentation is the task of decomposing a 3D shape into its meaningful components. In this context, meaningful components are those that a human being intuitively perceives as distinct, logical parts of the shape. Segmentations are useful in shape analysis, shape matching, medical imaging, collision detection, and other geometric processing methods employing divide-and-conquer strategies.

In defining what characterizes meaningful components, several directions have been pursued in the past. One of these is to use curve skeletons. The curve skeleton is a compact shape descriptor, much like a stick-figure representation. The curve skeleton of a 3D shape is a 1D connected structure that is centered within the shape, reflects its circular symmetries, and efficiently captures the topology and articulation of the shape [3]. The structure of the curve skeleton, consisting of branches and junctions, reflects the hierarchy of the meaningful components. The branches are associated with the components, whereas the junctions reflect the relationship between components.

In this paper we present a new framework for hierarchical shape segmentation using the curve skeleton. The curve skeleton is formally defined in terms of geodesics on the object's surface [20]. For each curve skeleton point these geodesics divide the boundary into a set of connected components, providing a natural skeleton-to-boundary mapping that is intrinsic to the definition of the curve skeleton. Being geodesics, segment borders are smooth and minimally twisting. After computing the meaningful components, we combine them into a hierarchical level-of-detail segmentation. Although we use the curve skeleton junctions to provide the meaningful components, other points can be used as well, depending on the target application. Besides the segmentation framework, this paper contributes a robust algorithm for computing the curve skeleton of voxelized objects which enhances the method in [20], and a method to robustly detect junctions.

Figure 1 provides an overview of our approach, consisting of four stages. First, the curve skeleton is computed from the voxelized input shape. Second, the *critical points* are chosen either automatically as the junctions of the curve skeleton, or manually by the user. Third, we compute the component sets of the critical points. Finally, a level-of-detail segmentation is created from the component sets.

The outline of this paper is as follows. In the next section we review related work. In Section 3 we present some preliminary definitions. In Section 4 we provide the details of the curve skeleton definition and computation. In Section 5 we elaborate on the computation of the component sets, and how based on them junctions can be detected robustly. Section 6 deals with creating the hierarchical segmentation from the component sets. In Section 7 we present the results of the algorithm. Section 8 presents a discussion and we conclude in Section 9.

D. Reniers
Department of Mathematics and Computer Science, Eindhoven University of Technology, The Netherlands, E-mail: d.reniers@tue.nl

A. Telea
Institute for Mathematics and Computing Science, University of Groningen, The Netherlands, E-mail: a.c.telea@rug.nl

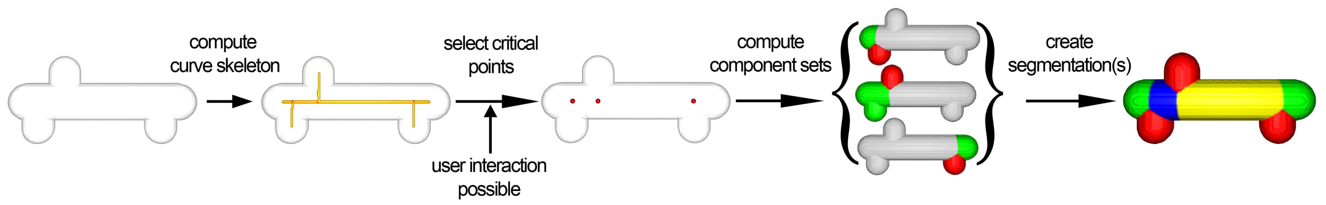


Fig. 1 Overview of our segmentation framework

This paper extends our previous work on curve-skeleton-based shape segmentation [17] in a number of directions: 1) we present an optimized curve-skeleton computation which speeds up the overall segmentation process, 2) we propose an improved junction-detection method to handle objects with tunnels, and 3) we compare our curve-skeleton-based segmentation method, which produces part-type segmentations, with a new surface segmentation method producing patch-type segmentations, and discuss the relative specifications of the two skeleton-based segmentations.

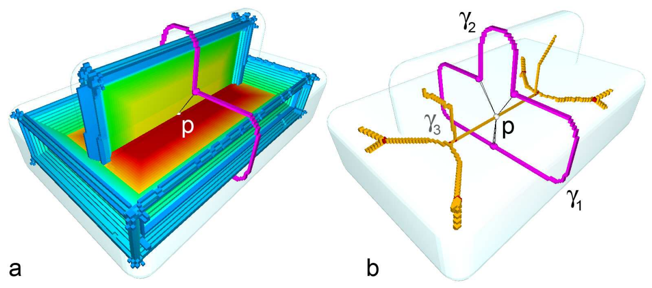


Fig. 2 The surface skeleton with rainbow color map indicating geodesic length (blue is short, red is long) (a). The curve skeleton $\mathscr{C}$, a selected point $p \in \mathscr{C}$, and $\Gamma(p) = \{\gamma_1, \gamma_2, \gamma_3\}$ (b)

2 Related work

Shapes can be segmented by considering their boundary or interior. Approaches that are based on the boundary usually define segment borders at object-surface concavities. Katz and Tal [10] use fuzzy clustering on geodesic and angular distances between surface elements to obtain a hierarchical mesh decomposition with non-jaggy borders. They show that their segmentation can be used to compute a (control) curve skeleton. In [9], pose-invariant segmentations are produced by extracting feature points and cores. In [2], a fuzzy clustering on quasi-flat surface features separated by curvature extrema is used to obtain a multiscale segmentation of point sets. In [13], an automatic scissoring scheme is proposed based on 3D snakes.

Segmentation methods using the curve skeleton consider the shape's interior. Our approach falls into this category. To obtain meaningful components, these methods require a mapping from the curve skeleton to the object boundary. Various mappings have been proposed. In [14] for example, a combination of planar cross-sections and space sweeping is used. The approach in [4] uses force-following of boundary particles onto the curve skeleton. A comparative study of some of the latest segmentation methods can be found in [1].

Our segmentation approach is similar to that of Li et al. [14], in which the segmentation is also based on curve skeletons. However, there are some important differences. Our curve skeleton algorithm has a more fundamental underpinning (see [20] for details), is more noise resistant, and is connected by default. Li et al. use a planar cross-section sweeping along the curve skeleton between critical points to obtain components. The definition of our curve skeleton is such that it provides a natural skeleton-to-boundary mapping. Instead of taking the planar cross-section as borders between components, we use the actual curve skeleton definition and take the shortest geodesics between feature points as borders. This makes for more natural borders between segments (compare e.g. the Hand object in Fig. 11 with [14, Fig.13]). Furthermore, our framework provides a hierarchical segmentation.

3 Definitions

The surface skeleton $\mathscr{S}$ of a 3D object Ω with boundary $\partial\Omega$ is defined as those points in Ω having at least two boundary points at minimum distance:

$$\begin{aligned}\mathscr{S} = &\{p \in \Omega \mid \exists a, b \in \partial\Omega, a \neq b,\\ &dist(p,a) = dist(p,b) = \min_{k \in \partial\Omega} dist(p,k)\},\end{aligned} \tag{1}$$

where *dist* is the Euclidean distance in $\mathbb{R}^3$. Points a, b are the points at minimum distance of p and are called the *feature points* of p. Let $F : \Omega \to \mathscr{P}(\partial\Omega)$ be the *feature transform* which assigns to each object point its set of feature points, where $\mathscr{P}$ denotes the power set. The surface skeleton consists of 2D manifolds, called *sheets*, and of 1D curves [7]. The sheets intersect in *sheet-intersection curves*. Points on these intersection curves have at least three feature points. The algorithms we present in this paper all work in binary voxel space [12]. When used in a discrete-space context, Ω denotes the set of foreground, or object voxels.

4 Curve skeleton computation

4.1 Definition

First, we define the curve skeleton $\mathscr{C}$ for the generic case, namely when the curve skeleton is not (partly) incident with a sheet-intersection curve, so that there are exactly two feature points for each point $p \in \mathscr{C}$. The curve skeleton is defined as those points having at least two shortest geodesics γ_i, γ_j between their two feature points $F(p)$:

$$p \in \mathscr{C} \Leftrightarrow \exists \text{ two shortest geodesics } \gamma_i \neq \gamma_j \text{ between } F(p) \tag{2}$$

A similar definition was first presented in [6], and in [20]. This definition ensures that the curve skeleton is centered in two ways. First, the curve skeleton is included in the surface skeleton, as the surface skeleton is defined as those points having at least two feature points. Second, it is also centered within the surface skeleton structure, as the two shortest geodesics γ_i, γ_j necessarily have the same length.

In the non-generic case, the curve skeleton is partly incident with a sheet-intersection curve, and there are more than two feature points for a point $p \in \mathscr{C}$. For example, Figure 2a depicts the surface skeleton of a box with a vertical ridge. Figure 2b depicts the curve skeleton of the same object. Point p lies on the sheet-intersection curve of the box's sheet and the ridge's sheet, and has three feature points. Point p clearly is a curve skeleton point, but the definition Eq. 2 does not detect this. Consequently, this definition cannot be used to detect all curve skeleton points, so we modify it as follows. With each intersecting sheet a feature-point pair a, b is associated and thus a shortest geodesic γ_i between a, b. For example, for three intersecting sheets there are three shortest geodesics (Fig. 2b), not necessarily of the same length. We call the combination of these shortest geodesics for a surface skeleton point $p \in \mathscr{S}$ the *shortest-geodesic set* $\Gamma(p)$:

$$\Gamma(p) = \{\gamma_i\}_i, \tag{3}$$

where γ_i is a shortest geodesic between the feature-point pair $a, b \in F(p)$ associated with a sheet.

Both in the generic and non-generic case, the union of shortest geodesics form a Jordan curve on the object surface $\partial\Omega$. Hence, detecting a curve skeleton point p comes down to detecting whether $\Gamma(p)$ contains a ring, i.e. computing the genus of $\Gamma(p)$. By the genus of Γ, we mean the genus of the surface that is obtained by taking a infinitesimal dilation of Γ on $\partial\Omega$. The new definition of the curve skeleton replaces Eq. 2 and becomes:

$$p \in \mathscr{C} \Leftrightarrow \text{genus}\big(\Gamma(p)\big) \geq 1 \tag{4}$$

Additionally, the genus of Γ can be used to differentiate between junction and non-junction points, called *regular* points. A junction is a point on the curve skeleton where at least three branches come together. The shortest-geodesic set of such a junction is the union of the Jordan curves of the neighboring regular points. Hence, if the genus of $\Gamma(p)$ is larger than 1, p is a junction:

$$p \in \text{junctions}(\mathscr{C}) \Leftrightarrow \text{genus}\big(\Gamma(p)\big) \geq 2 \tag{5}$$

As mentioned, a similar curve skeleton definition was first presented by Dey and Sun [6], namely as the set of singular points of the the so-called medial geodesic function, which assigns to each surface skeleton point the length of the geodesic between its two feature points, and they show that this curve skeleton is homotopy equivalent to the original shape. In contrast, we detect the curve skeleton by performing a topological analysis of the shortest geodesic set. This definition allows us to compute the curve skeleton without first detecting the surface skeleton, and readily allows junction detection. At an implementation level, whereas [6] uses polygonal representations, our implementation of the curve skeleton detection is voxel-based, as detailed in the next section.

4.2 Algorithm

Based on the above definitions, we now present our algorithm for computing the curve skeleton of a voxelized object Ω. The algorithm consists of several steps that can be executed in parallel for each object voxel. The different stages are depicted in Figure 3. In the first step, we compute the feature transform F (using [15]) and extended feature transform $\overline{F}$ of Ω. The *extended feature transform* merges the feature set of each voxel p with that of its first octant 26-neighbors: $\overline{F} = \bigcup_{x,y,z \in \{0,1\}} F(p_x + x, p_y + y, p_z + z)$. The result is that a regular curve skeleton point has two (compact) groups of feature voxels (Fig. 3a), one on each side of the surface skeleton. The extended feature transform solves two problems. First, it solves the well-known problem that in discrete space the feature set of a surface skeleton voxel might contain only one feature voxel, since there might be no voxel exactly in the center of the voxelized boundary [5,19]. Second, it solves the related problem that on a voxelized object surface, there is typically only one shortest path between the two feature voxels of a curve skeleton voxel, although there are two on the underlying continuous surface.

After computing $\overline{F}$, we compute the shortest-geodesic set $\Gamma(p)$. Because we perform our computations in discrete voxel space, we compute the shortest geodesics as shortest paths in the *boundary graph* in which the surface voxels are the nodes and their neighborhood relations represent the edges. The shortest paths are represented as 3D chain codes [11] in the boundary graph. For computing a shortest path we use the A* algorithm [8], a modification of Dijkstra's algorithm, with Euclidean distance as the search heuristic. We compute a shortest path between each two feature voxels in $\overline{F}(p)$. Because $\overline{F}(p)$ contains groups of voxels, we will find numerous shortest paths for a regular point having only two feature points. Together, they form a discrete (noisy) Jordan curve on the object surface (Fig. 3b).

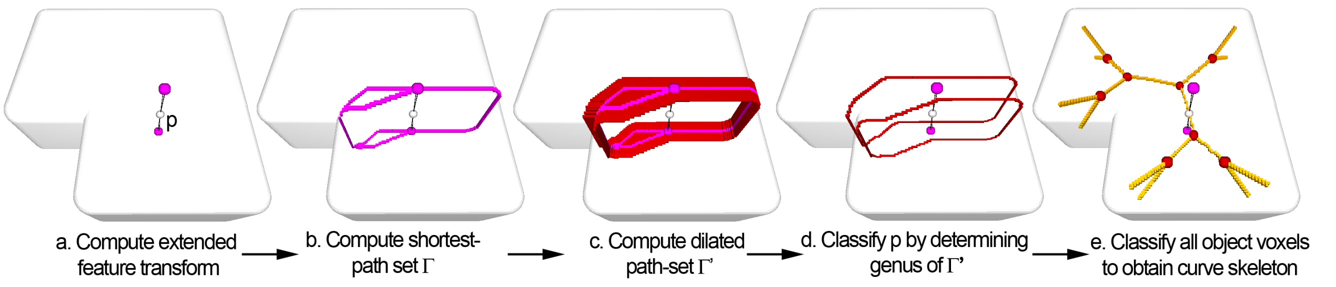

p
a. Compute extended feature transform
b. Compute shortest-path set Γ
c. Compute dilated path-set Γ'
d. Classify p by determining genus of Γ'
e. Classify all object voxels to obtain curve skeleton

Fig. 3 Curve skeleton computation and junction detection

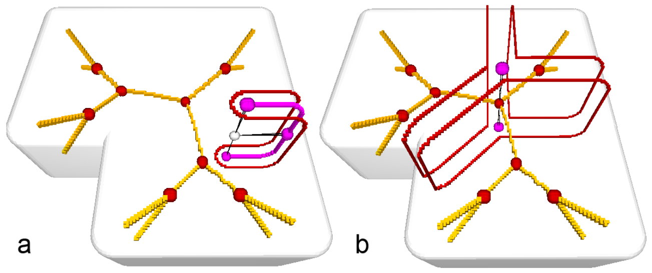


Fig. 4 The boundaries of two dilated path-sets

After computing the path set $\Gamma(p)$, we determine its genus on $\partial\Omega$, as follows. First we dilate $\Gamma(p)$ so that we obtain a surface band $\Gamma'(p)$ centered at the 1D structure $\Gamma(p)$ (Fig. 3c). Conveniently, small holes from the path set that are caused by the discrete nature of the shortest paths (Fig. 3b) are removed by the dilation. The dilation is performed by propagating the voxels in $\Gamma(p)$ a short distance outward using a distance-ordered flood fill. Next, we determine the number of compact boundary pieces that the area $\Gamma'(p)$ has. If two or more boundaries are found, p is concluded to be a curve skeleton voxel (Fig. 3d). When only one boundary is found, the voxel is concluded to be a non-curve skeleton voxel (Figure 4a). Empirical studies on an extensive family of real-world 3D shapes show that a (geodesic) dilation distance of 5.0 is enough to obtain two connected boundaries if $\Gamma(p)$ is the discrete representation of a Jordan curve. In principle, when three or more boundaries for $\Gamma'(p)$ are found, p is a junction. In Figure 4b for example, three boundaries are found for the junction voxel. However, due to the discrete nature of the shortest paths, curve skeleton voxels might be incorrectly classified as junctions, i.e., junction detection by analyzing the genus of Γ is conservative. This is problematic if we want to use the junctions in segmentation (Sec. 6). We address this problem in Section 5.4.

The steps described above (Fig. 3a-d) can be executed for all object voxels in parallel, resulting in the curve skeleton $\mathscr{C}$ (Fig. 3e). Figure 5 shows four curve skeletons as computed by our algorithm. We stress that these results are obtained without any post-processing, and that our algorithm does not perform any thinning or erosion step. The curve skeletons, and consequently their junctions, are up to 2 voxels thick, due to the extended feature transform. We observe that the computed curve-skeleton is homotopic to the object, and is connected, although some noise is present which represents insignificant parts of the shape. In Section 5.3 we present an importance measure for $\mathscr{C}$ that can be used to obtain noise-resistant, simplified skeletons. This measure is based on the component sets that are associated with each point $p \in \mathscr{C}$, as explained in Sec. 5.

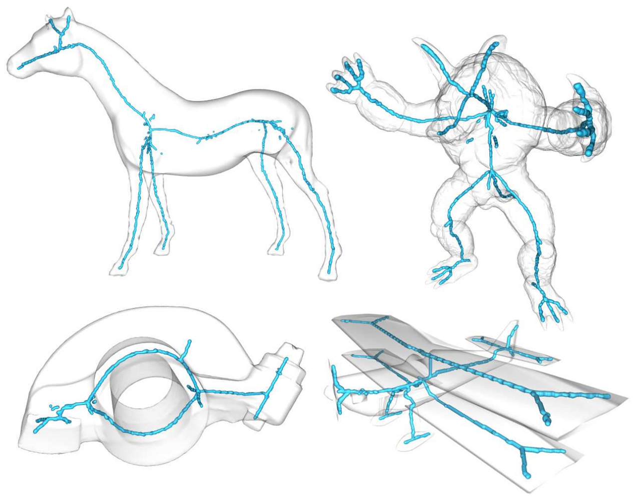


Fig. 5 The curve skeleton $\mathscr{C}$ of several objects

4.3 Skeleton computation speed-up

The fact that the curve skeleton is a connected structure allows us to make the following optimization in its computation. Instead of parallel processing of *all* object voxels, we stop when we encounter a curve-skeleton voxel p, called the *seed* voxel. We classify the 26-neighboring voxels of p, and recursively continue the curve-skeleton detection only for those neighbors classified as curve-skeleton voxels. Because we have seen in Fig. 5 that there may be some small parts of the curve skeleton that are disconnected from the main part, we search for a new seed voxel if the number of curve-skeleton voxels detected so far is smaller than a threshold, which we set to an empirically determined value of 10 voxels for all shapes shown in this paper. This simple modification

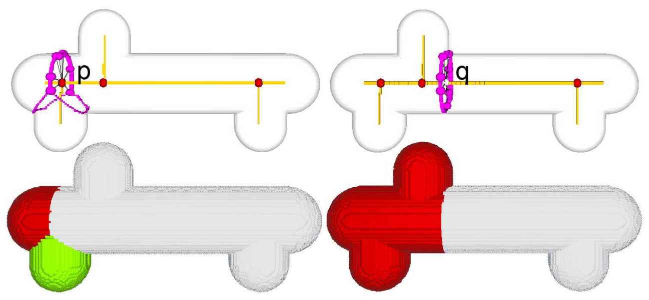


Fig. 6 The component sets (bottom) of two selected points p,q (top)

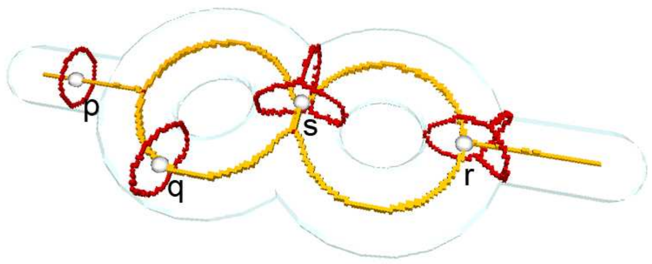


Fig. 7 An object with two tunnels and Γ of four selected points

of the algorithm means that we no longer have to visit all object voxels, typically giving a speed-up of a factor 8. For example, the time needed for computing the curve skeletons for the Horse and Armadillo objects from Fig. 5 goes down from 200 and 86 seconds to 25 and 9 seconds respectively. An overview of the performance obtained with our speed-up is given in Table 1.

5 Component sets

In the previous section we described the detection of a curve skeleton point p by analyzing the genus of its associated geodesic set $\Gamma(p)$. In this definition of the curve skeleton lies a natural skeleton-to-boundary mapping. For objects of genus 0, the Jordan curve theorem states that the Jordan curve Γ associated with a regular point divides the object surface $\partial\Omega$ into two connected components. For a junction point, Γ is the union of Jordan curves of neighboring regular points, dividing the boundary into multiple components. The *component set* of a point p is denoted $C(p)$. Let the k components in C be ordered by their areas: $\forall_{1\leq i<k}\ |C_i| \leq |C_{i+1}|$. The area of a component $C_i(p)$ is determined simply by counting its voxels: the cardinality of the set $C_i(p)$. Although we could use a more sophisticated surface estimator, the cardinality is sufficient for our purposes. In our voxel space representation, the component set $C(p)$ for a voxel p is computed by labeling the connected components in the boundary graph from which the voxels from the path set $\Gamma(p)$ are removed. The labeling is sped up using a simple spatial-subdivision scheme.

Figure 6 shows the component sets for two selected curve skeleton points p,q, where p and q are a junction and regular point respectively. The component sets of the junctions are used to obtain a segmentation in Section 6.

5.1 Topological Properties

It is important to note that the number of components $|C(p)|$ for a point $p \in \mathscr{C}$ is related to whether the shape Ω has tunnels, as this will affect the final segmentation. For shapes without tunnels or holes (genus 0), the curve skeleton has a tree structure. For objects with tunnels (genus $\geq$ 1), the curve skeleton is a graph that contains a *loop* around each tunnel. Since there is a skeleton-to-boundary mapping through the use of Γ, whether $\Gamma(p)$ divides the boundary into multiple components is equivalent to whether p divides the graph into multiple components. In graph theory, a point p dividing the graph into multiple connected components is called a *cut vertex* [24]. The number of components $|C(p)|$ is related to the genus of $\Gamma(p)$ and the amount of loops $L(p)$ that p is on:

$$|C(p)| = \max\left(\text{genus}(\Gamma(p)) + 1 - L(p), 1\right). \tag{6}$$

Figure 7 shows an example shape with two tunnels and four selected points. Points p,q are regular (non-junction) points (genus$(\Gamma(p)) = 1$, genus$(\Gamma(q)) = 1$), whereas r,s are junctions of three branches (genus$(\Gamma(r)) = 2$, genus$(\Gamma(s)) = 2$). Point p is not on a loop, points q,r are both on one loop, and s is on two loops. Indeed, we verify that the cardinality of the component sets generated by p,q,r,s is $2,1,2,1$ respectively. Points p,r are cut vertices in the curve skeleton graph, whereas q,s are not.

5.2 Inclusion Property

An important property of the component sets for the purpose of hierarchical segmentation is that any two components $C_i(p), C_j(q)$ for two points $p,q \in \mathscr{C}$ cannot partly overlap:

$$C_i(p) \cap C_j(q) \in \{\emptyset, C_i(p)\} \qquad \text{if } |C_i(p)| \leq |C_j(q)| \tag{7}$$

In order to prove Eq. 7 for two regular points p,q, we have to prove that the Jordan curves $\Gamma(p)$ and $\Gamma(q)$ cannot intersect. The proof can be roughly sketched as follows. Take that subset $\mathscr{C}_{pq} \subset \mathscr{C}$ that is between p and q. It is reasonable to assume that the feature points $F(\mathscr{C}_{pq})$ generate two curves L_1, L_2 on $\partial\Omega$, one on each side of the surface skeleton. Each geodesic $\gamma \in \Gamma$ has one endpoint on L_1 and the other endpoint on L_2. Such a geodesic γ does not intersect L_1 or L_2 in any other point as it would no longer be the shortest geodesic. Now, if a geodesic $\gamma_p \in \Gamma(p)$ intersects $\gamma_q \in \Gamma(q)$, it needs to have (a multiple of) two intersection points. But if γ_p intersects γ_q in two points a,b, the shortest geodesic between a,b would not be unique, which is impossible.

5.3 Importance Measure

The component sets can be used to define an importance measure for computing simplified curve skeletons that are robust to noise, as follows. The *importance measure* $\rho(p)$ for a point $p \in \mathscr{C}$ is defined as the importance of p in representing the original shape. One way of expressing $\rho(p)$ is in terms of the component areas associated with p. Recall that the components in C are ordered by their areas, or voxel count: $\forall_{1 \le i < k}\ |C_i| \le |C_{i+1}|$. We call the largest component C_k the *background component*. The others are called *foreground components*. The importance measure ρ for a point p is defined as the total area of the foreground components:

$$\rho(p) = \frac{1}{|\partial\Omega|}\Big|\bigcup_{1 \le i < k} C_i(p)\Big| \tag{8}$$

We normalize ρ by the total object surface area $|\partial\Omega|$. Note that this importance measure can only be computed for non-loop points. A regular loop-point has only one component and no foreground components (e.g. Fig. 7, point q), so that ρ is undefined. *Simplified curve skeletons* can be computed from a noisy curve skeleton $\mathscr{C}$ by discarding all points with an importance below a certain threshold τ:

$$\mathscr{C}_\tau = \{p \in \mathscr{C} \mid \rho(p) \ge \tau\} \tag{9}$$

The threshold τ is a user-parameter which controls what is considered noise. The meaning of this parameter is intuitive: all $\mathscr{C}$-points representing a smaller surface area than τ are discarded. Furthermore, $\mathscr{C}_\tau$ can be considered a multiscale representation of the curve skeleton. The simplified skeletons are connected by default at all scales, because ρ is monotonic, which follows from Eq. 7. The simplified curve skeletons $\mathscr{C}_\tau$ of several shapes are shown in Figure 11, with $\tau = 0.01$. Thus, curve skeleton points representing an area less than 1% of the object surface are discarded.

This definition of ρ has been presented in [20]. There, the curve skeleton detection was integrated with the computation of ρ. In contrast, the skeletonization method presented here computes a curve skeleton by analyzing the genus of Γ, making it much faster to compute, and making it possible to handle objects with tunnels. Furthermore, in this paper we also present a method to robustly detect curve-skeleton junctions, as detailed in the next section.

5.4 Robust Junction Detection

As indicated in Section 4.2, the genus of $\Gamma(p)$ is a conservative criterion for detecting junctions. The computed genus may be higher than in the original object, due to boundary noise or discretization artifacts. This is problematic if we want to use the detected junctions for segmentation. Inspired by the importance measure from the previous section, we want to discard junctions which have small components among their components, as they likely result from noise. One could simply filter out small components in $C(p)$ and count the remaining components: If at least three components remain, point p is a junction. However, junctions on $\mathscr{C}$-loops cannot be detected in this manner, as these junctions have only two components (Sec. 5.1).

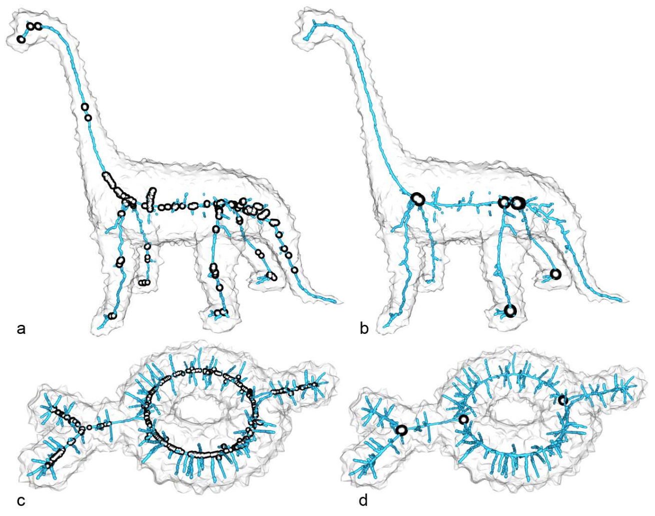


Fig. 8 Conservative junctions J_0 (a,c). Robust junctions J_τ (b,d)

The key idea here is to associate each connected boundary piece of the dilated path-set $\Gamma'(p)$ (Sec. 4.2) with the component in $C(p)$ it bounds, and then filter and count the boundary pieces instead of the components. Concisely put, a point p is a junction-point if and only if its dilated path-set Γ' has at least three connected boundary pieces, that bound large enough components. We denote the set of robust junctions J_τ:

$$J_\tau = \big\{p \in J_0 \;\big|\; |\Gamma'_\tau(p)| \ge 3\big\}, \tag{10}$$

where J_0 is the set of conservatively detected junctions from Section 4.2, and $\Gamma'_\tau(p)$ are the filtered boundary pieces in the dilated path-set $\Gamma'(p)$:

$$\Gamma'_\tau(p) = \Big\{b \in \Gamma'(p) \;\Big|\; c \in C(p) \wedge b \subseteq \partial c \Rightarrow \frac{c}{|\partial\Omega|} > \tau\Big\}, \tag{11}$$

where b is a connected boundary piece from $\Gamma'(p)$ and ∂c is the boundary of component c with respect to $\partial\Omega$. Although related to the computation of ρ, we do not have to compute ρ to compute J_τ: We only have to compute the component sets for the conservative junctions J_0. Equivalent to the computation of $\mathscr{C}_\tau$, the scale τ is a user parameter used to distinguish between small-scale noise and signal.

Figure 8 shows the effect of using Eq. 10. Figs. 8a and c show the noisy Dino and Genus1 objects with conservatively detected junctions J_0 depicted as black balls. Figs. 8b and d show the robust junctions computed by Eq. 10 with $\tau = 0.05$.

6 Segmentation

We now present two methods for computing a shape segmentation from the critical points. By segmentation we mean the final segmentation S of the object into disjoint and connected segments S_i. In order to obtain a segmentation into meaningful components, the junctions of the curve skeleton are detected and used as critical points. Alternatively, we can pick critical points by some other (semi) automatic process. The first segmentation method (Sec. 6.1) we present is the most straightforward. It is based on the geodesic sets of the critical points, and produces a flat (non-hierarchical) segmentation at the finest scale. The second method (Sec. 6.2) is based on the component sets generated by the critical points, and produces a hierarchical, level-of-detail segmentation.

6.1 Flat Segmentation

We produce a simple, flat segmentation by considering the shortest-path sets Γ of all critical points P at once. The connected components in the boundary graph due to all these shortest paths are labeled. A segmentation that is obtained in this way is shown in Figure 10b, using the shortest-path sets of the critical points shown in Fig. 10a. One property of this segmentation is that it splits tunnel-parts of the object into multiple segments.

6.2 Hierarchical Segmentation

In order to produce a hierarchical, level-of-detail segmentation, we consider the component sets C of the critical points P. In Section 5.3 we distinguished between foreground and background components. In a component set C consisting of k components $C_{1..k}$, the largest component C_k is called the background component, the remaining ones are called foreground components. The foreground components are those that one would consider meaningful and intuitively associate with the critical point. The background component is merely the remaining surface area. In Figure 6 for example, the background components are light gray. Furthermore, we observe that the foreground components combined also form a meaningful component. We denote this compound component by $C'(p) = \cup_{1 \le i < k} C_i(p)$.

Let F be the set of meaningful components of all critical points combined. The segmentation S should be based on F, but the components in F are not disjoint. We now present an algorithm for creating a segmentation S from F consisting of disjoint segments S_i at a certain scale τ. The pseudo code of the algorithm is shown in Figure 9.

The idea is that we consider all components $f \in F$ in ascending order of their area (line 5), but only those components that are larger than the specified scale τ (line 6). The potential segment s is computed as the set difference between f and the existing segments in S: $s = f \setminus S$ (line 7). Before adding the potential segment s to S, we check whether f overlaps any existing segments from S. If not, s is added without restriction. If f overlaps, we only add s if it contributes enough to the segmentation, that is, if it adds at least 10% of the area that it overlaps (line 8). This is to prevent tiny segments due to different junctions having similar components among their component sets. This occurs for example due to the fact that junctions computed by the algorithm in Section 4 may consist of multiple voxels, having almost the same component sets. After processing all components in F, because the background components have been left out in the segmentation, the object surface might not be fully covered. Therefore, we add to F the whole boundary as the largest component (line 2), which ends up as a single segment filling up the remaining part.

```
1:  $F \leftarrow \{C_i(p) | p \in P, 1 \le i < k\} \cup \{C'(p) | p \in P\}$
2:  $F \leftarrow F \cup \{\partial\Omega\}$
3:  **procedure** computeS($\tau$)
4:  $S \leftarrow \emptyset$
5:  **for** each $f \in F$ in ascending order of $|f|$ **do**
6:      **if** $\frac{1}{|\partial\Omega|}|f| \ge \tau$ **then**
7:          $s \leftarrow f \setminus S$
8:          **if** $f \cap S \ne \emptyset \Rightarrow |s| > 0.1 \cdot |f \cap S|$ **then**
9:              $S \leftarrow S \cup \{s\}$
10:         **end if**
11:     **end if**
12: **end for**
13: **end procedure**
14: **procedure** computeLevelsOfDetail()
15: **for** each $f_i, f_{i+1}$ in $F$ in ascending order of $|f|$ **do**
16:     **if** $|f_{i+1}| - |f_i| > 0.1 \cdot |f_i|$ **then**
17:         computeS($\frac{1}{|\partial\Omega|}|f_i|$)
18:     **end if**
19: **end for**
20: **end procedure**
```

Fig. 9 Algorithm for computing a level-of-detail segmentation

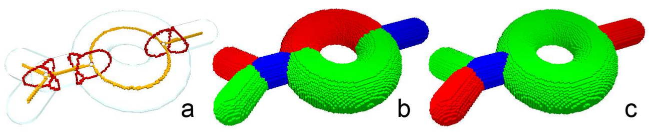


Fig. 10 Critical points and their path sets (a). Flat segmentation (b). Finest level of the hierarchical segmentation (c)

In order to compute multiple segmentations from fine to coarse (line 14), we can simply consider all components from F in ascending order of area, and produce a segmentation for each of those areas. To limit the amount of generated hierarchy levels, in our implementation we only compute a hierarchy if two consecutive areas differ by at least 10% from the smaller area f_i (line 16). The different segmentations produced at the various scales actually form a hierarchy, because every segment is included in a segment from a coarser scale, by Eq. 7. Besides being hierarchical, another difference between this segmentation method and the flat one is that tunnel-parts are not segmented and are left intact (see Fig. 10c).

7 Results

We have implemented our framework in C++ and have run it on a Pentium 4, 3GHZ with 1GB of RAM. In computing the shortest paths we used a cache to prevent computing the same shortest path twice. As input we used several polygonal meshes voxelized using binvox (http://www.cs.princeton.edu/~min/binvox/), for various resolutions ranging up to 384 voxels. We used both organic and geometric objects. Figure 11 shows for four objects the simplified curve skeleton $\mathscr{C}_\tau$ and robust junctions J_τ, both with $\tau = 0.01$, and three selected levels from the hierarchical segmentation. Figure 12 shows the segmentations of the Tree object for all levels. Figure 13 shows for several objects the segmentation at the finest scale. We observe that our method is able to extract fine details, such as the toes and fingers of the Armadillo and Dinopet objects, and it does not suffer from over-segmentation. The use of geodesics as borders of the surface segments has the very beneficial effect of producing sharp, non-jagged separations, since geodesics are minimally twisting curves on the surface. Finally, our approach can handle objects with tunnels, as exemplified by the Genus1 object.

Table 1 shows performance measurements on our framework for several objects that are shown in this paper. Columns "object", "dim", "$|\Omega|$", "$|\partial\Omega|$" denote object name, voxel-grid dimensions, number of object voxels, and number of boundary voxels respectively. Column "init t." denotes the time needed for initialization, including loading the object and computing the spatial-subdivision datastructure needed for efficient computation of component sets. Column "$\mathscr{C}$ t." denotes the time in seconds to compute the non-simplified curve skeleton $\mathscr{C}$ (Sec. 4.2). Column "seg t." denotes the time for computing all levels in the hierarchical segmentation. This time strongly depends on the amount of levels generated, and is non-optimized. Column "ρ t." denotes the time required to compute ρ on $\mathscr{C}$ (Sec. 5.3), that is, to obtain simplified curve skeletons. Note that computing ρ is not needed for obtaining a segmentation. Column "mem" gives an indication of the peak memory usage. The time needed for computing the curve skeleton can be attributed to the computation of the shortest paths and dilations. It takes relatively long for the Plane, which can be explained by the fact that the Plane has a high surface area to volume ratio, meaning that the average shortest-path between feature voxels is relatively long. The time needed to compute the robust junctions J_τ is not shown as it is negligible: up to 5 seconds for the considered objects.

8 Discussion

8.1 Properties

The segmentations produced by our approach have the following desirable properties. Following from the curve-skeleton definition, the borders of the segments are piecewise geodesic. They do not necessarily follow local surface-concavities, but instead find the minimal-length path between feature points. Hence, the borders exhibit minimal twist on the surface and look natural.

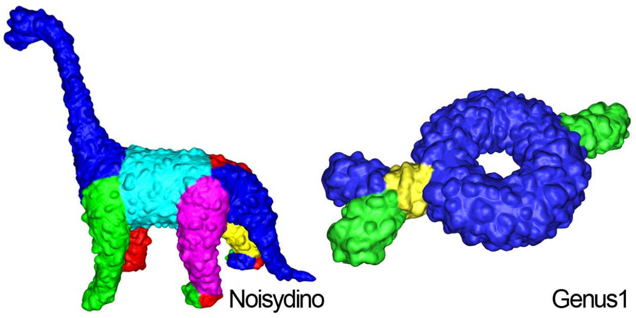


Fig. 14 Robust segmentation of noisy shapes

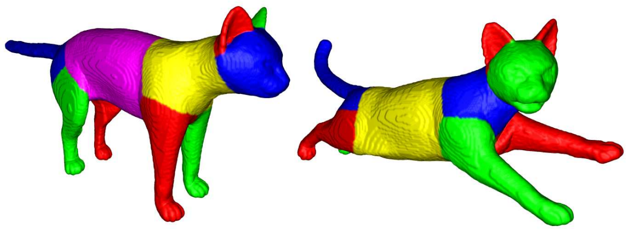


Fig. 15 Pose-invariance of the segmentations

Using geodesics for segment borders has the additional advantage that it yields stable and robust segments for very noisy shapes, as shown by the Noisydino and Genus1 objects (Fig. 14). For segmentation methods based on surface curvature (e.g. [2]), this noise could be problematic.

Segment borders are not based on boundary features but on the curve skeleton, capturing global geometrical (e.g. symmetrical) and topological properties. This has a number of advantages. The segmentations respect the object's circular symmetry and are invariant for different poses of the same object. This is because the curve skeleton structure in general remains stable under deformations of the object. Figure 15 shows the segmentations for two poses of the Cat (poses from [23]).

Borders are not necessarily found at curvature extrema. In the H-shape object for example, borders are found on flat parts of the surface, segmenting the tips of the H-shape. These segments are ascribed to the fact that the tips have sharp convex corners, which generate branches and junctions in the curve skeleton. A consequence of choosing only the junctions as critical points is that we do not find segment borders for curvature extrema in tubular-like parts of the object. In Figure 13 for example, the Dinopet object has no segment borders on its knees. In order to consider more the object's geometry in addition to its topology, we could choose extra critical points on the curve skeleton at curvature extrema, computed either on the curve skeleton or original surface.

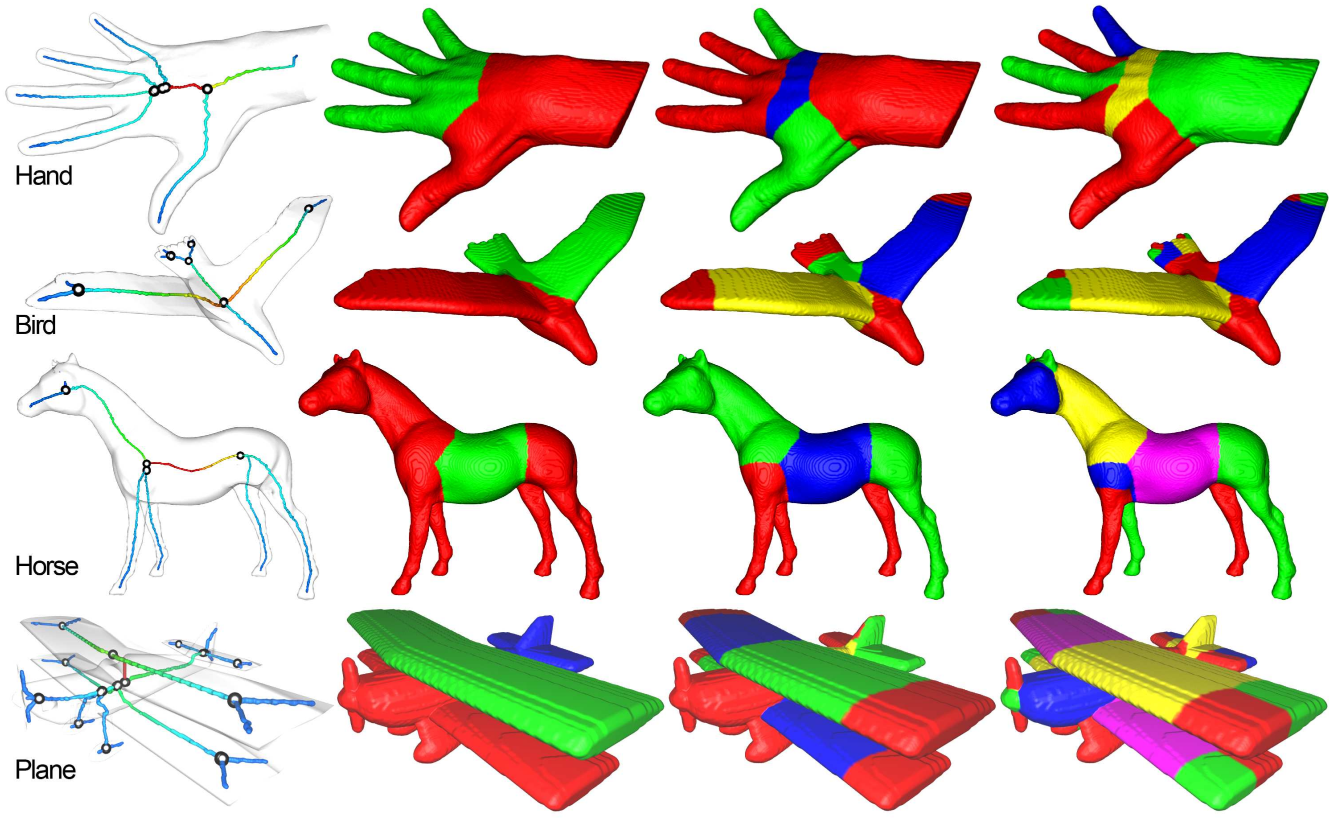


Fig. 11 Simplified curve skeletons $\mathscr{C}_\tau$ with detected junctions (first column), and three levels of the hierarchical segmentation (other columns)

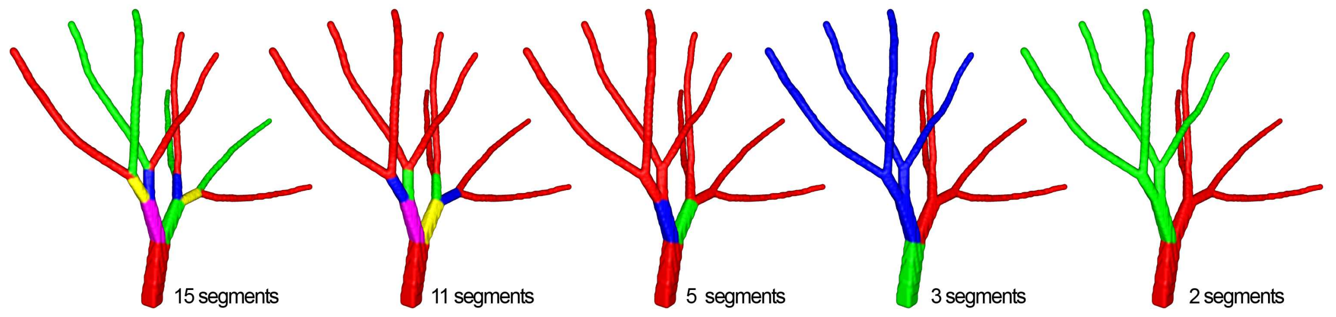


Fig. 12 All segmentation levels for the Tree object

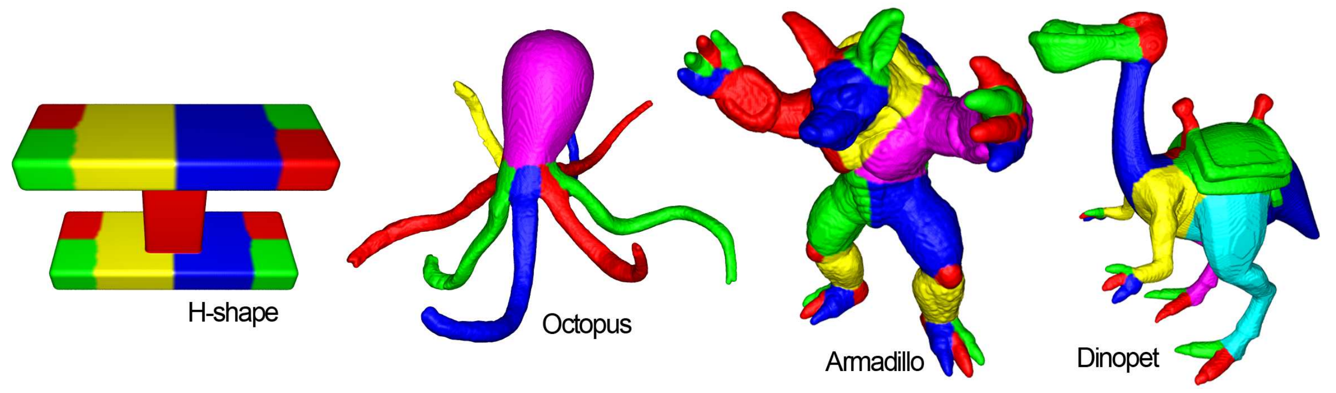


Fig. 13 Segmentations at the finest scale

Table 1 Table with measurements. See the text for details

object	dim	$\|\Omega\|$	$\|\partial\Omega\|$	init t. (s)	$\mathscr{C}$ t. (s)	seg t. (s)	ρ t. (s)	mem (MB)
Armadillo	188x245x207	905k	80k	23	8.6	41	7.5	447
Dinopet	334x366x180	1,810k	136k	49	15	54	40	707
Hand	366x154x257	1,300k	94k	36	30	21	5.6	540
Horse	366x316x171	2,038k	119k	48	25	34	10	660
Noisydino	125x346x365	1,421k	114k	41	16	19	14	552
Plane	217x304x98	545k	110k	20	53	21	39	492
Octopus	366x259x335	1,860k	154k	53	14	9.5	11	600

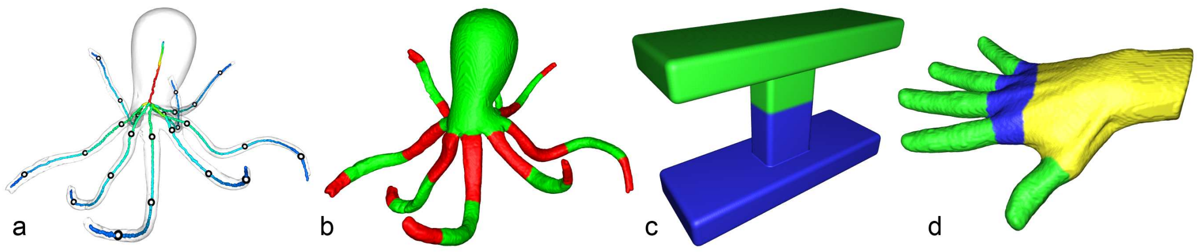


Fig. 16 Selected critical points (a). Manual segmentations (b-d)

As mentioned, our framework supports manual selection of critical points. In Figure 16a, we manually selected three such points on each tentacle of the Octopus. The resulting segmentation containing three segments per tentacle is shown in Fig. 16b. Another interesting possibility is to pick the "root" of the curve-skeleton as a critical point. The root is defined as that curve-skeleton point for which ρ reaches its maximum. In case the root r is a non-junction point, $\rho(r)$ is half the object-surface area. A segmentation based on the root thus divides the object in two in a *natural* manner, as exemplified in Fig. 16c for the H-shape. In future work, this special case could be extended to segmenting the object in n equally sized segments, where n is a user-parameter.

Finally, our method has the limitation that segment borders do no always tightly wrap around attached object parts, which might be undesirable for some applications. In the Hand object (Fig. 11) for example, the segment borders do not wrap tightly around the thumb and fingers. The reason for this is that the associated junctions lie deep within the palm of the hand, so that the feature points, and thus the ends of the geodesics that are computed between the feature points, are far from the attachment. One direction to overcome this limitation would be to move the segment border by moving the corresponding critical point away from the junction point along the curve-skeleton branch, until the segment border shows less strain. A manual segmentation exemplifying this method is shown in Fig. 16d.

8.2 Part-type and patch-type segmentations using the skeleton: a comparison

Our curve-skeleton-based segmentation produces good results for a wide range of organic, articulated, shapes, as shown in Fig. 13. However, for faceted shapes, such as polyhedral models, the obtained segmentations are less natural. For example, one would naturally expect the H-shape in Fig. 13 to be segmented in its planar faces. The question arises for which shapes the proposed curve-skeleton-based segmentation is appropriate, and for which not. Can we still use skeletons, possibly in a different way, to segment those shapes where our current method fails to deliver natural results?

To answer the above questions, let us recall the classification of segmentation methods into two classes: part-type and patch-type [21, 1]. Part-type segmentation methods decompose an object into its "meaningful components", such as the fingers and palm of a hand, or limbs, head and torso of a body. A part is not necessarily (quasi)flat, but is of a volumetric nature and is for the larger part separated from the other parts of an object by concavities. Patch-type segmentation methods on the other hand, decompose an object into quasi-flat regions, such as the six faces of a box. Parts are separated by regions of high positive or negative curvature, i.e. edges.

Our curve-skeleton based method is a *part-type* segmentation method by excellence. This method works well for objects which have a representative curve skeleton. By this, we mean a curve skeleton whose branches correspond logically to the parts, or segments, of the shape. Box-like objects, e.g. the H-shape in Fig. 13, do not fall in this category. Hence, we obtain the unexpected segmentation of the corners.

We can use the skeleton to produce a more precise classification of objects into part-segmentable and patch-segmentable. For this, we consider not only the curve-skeleton, but also the *surface skeleton* of a shape (for an explanation of the differences see Eqs. 1 and 2). Figure 17 shows a comparison between two shapes. The left column shows an organic shape, the perfect candidate for the part-type segmentation. We see that its curve skeleton is quite similar to its surface skeleton, i.e., the curve skeleton is just a thin version of the surface skeleton. The right column shows a polyhedral shape, the perfect candidate for a patch-type segmentation. Here we see that its curve

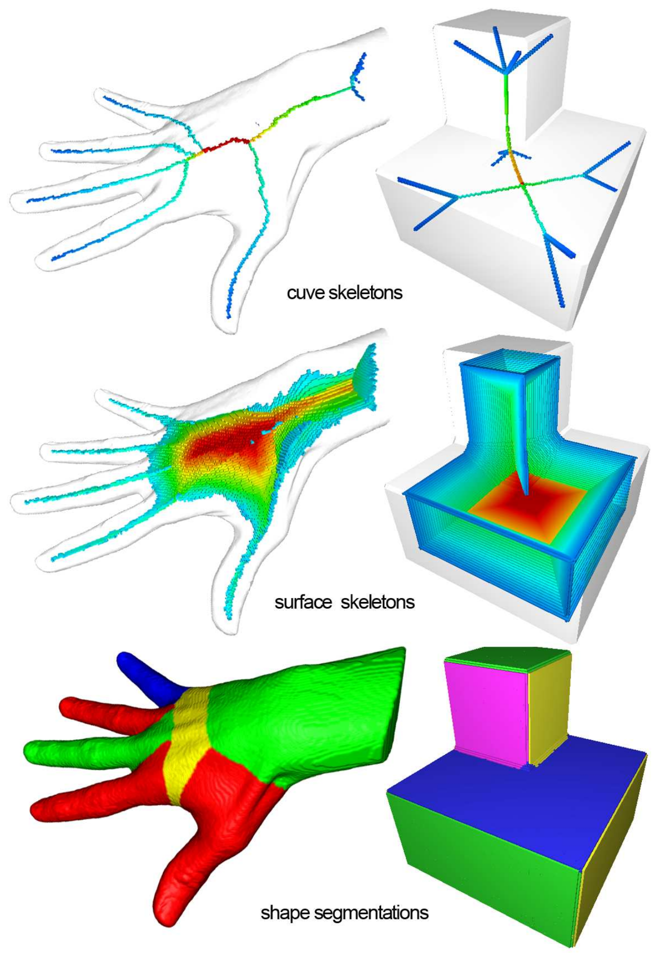


Fig. 17 A part-segmentable object (left column) versus a patch-segmentable object (right). Rows show curve skeletons, surface skeletons, and the corresponding segmentations respectively

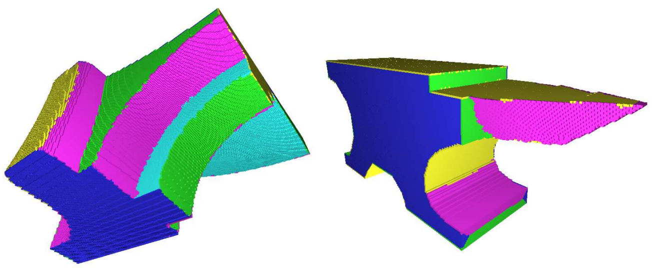


Fig. 18 Patch-type segmentations using the surface skeleton

skeleton differs strongly from its surface skeleton: It has a different structure. In other words, shapes whose curve and surface skeletons do not differ significantly are well described by their curve skeleton only, and are good candidates for part-type segmentations. Shapes where the two skeletons differ significantly have extra information which cannot be captured in the curve skeleton. For these shapes, a curve-skeleton-based segmentation might not characterize the object adequately for the application at hand.

Table 2 Comparison of skeleton uses for shape segmentation

	curve skeleton	surface skeleton
segmentation type	part-type	patch-type
segment correspondence	skeleton branch	skeleton sheet
segment type	volume	surface
multiscale	yes	yes
noise robust	yes	yes

In view of patch-segmentable objects, remark that the surface skeleton consists of a set of medial sheets [7]. The boundary of the surface skeleton corresponds and reaches into curvature maxima, i.e. edges, of the shape [16]. Hence, the skeleton sheets which share these borders correspond to quasi-flat shape regions separated by an edge. This observation permits us to use the surface skeleton as a possible tool to produce patch-type segmentations of shapes. One possible approach is to segment the surface skeleton into its sheets, using e.g. [18], and to segment the shape into areas corresponding to these sheets. Preliminary results of this approach yield good patch-type segmentations, see e.g. Fig. 18. Using surface skeletons to produce patch-type segmentations preserves several advantages of the curve-skeleton-based method presented in this work: It produces segmentations which reflect the object's symmetry, is robust to small-scale surface noise, and delivers multiscale segmentations by changing the skeleton simplification level [20].

Concluding, both curve and surface skeletons can be used to produce shape segmentations with a number of properties, as summarized in Table 2.

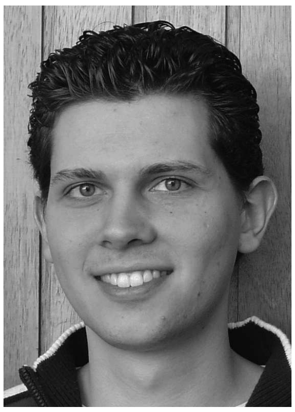
9 Conclusion

We have presented a novel framework for hierarchical segmentation of 3D shapes. Being based on the formally defined curve skeleton, our framework has a solid underpinning. The produced segmentations inherit several desirable properties, such as pose-invariance, from the curve skeleton and reflect the symmetry of the object. The definition of the curve skeleton in terms of shortest geodesics gives rise to a natural skeleton-to-boundary mapping. The meaningful components are found using the curve skeleton junctions and are combined into a hierarchical, level-of-detail segmentation. Being piecewise geodesic, the segment borders are smooth and non-twisting. Interestingly, because our method is based on the curve skeleton representing the object's interior, our method produces both a surface segmentation and a corresponding volumetric segmentation. The framework supports segmentations based on the critical points defined as the curve skeleton junctions, but also defined in other ways. A voxel-based implementation is provided. Besides the segmentations, it computes multiscale curve skeletons that are robust to boundary noise, and performs robust junction detection. We showed that our framework delivers good results on a wide range of (noisy) objects. Finally, we discuss the limitations of curve-skeleton-based segmentations for shapes which have a patch-type structure, and show how the surface skeleton can be used to produce patch-type segmentations for such objects. In future work, we would like to detect additional critical points to obtain segment borders at curvature extrema of the object surface.

Acknowledgements This work was supported by the Netherlands Organization for Scientific Research (NWO) under grant number 612. 065.414. We used shapes from the AIM@SHAPE shape repository (http://shapes.aim-at-shape.net) and the Princeton Shape Benchmark [22].

References

1. Attene, M., Katz, S., Mortara, M., Patane, G., Spagnuolo, M., Tal, A.: Mesh segmentation - a comparative study. In: IEEE Int. Conference on Shape Modeling and Applications (SMI06), p. 7 (2006)
2. Clarenz, U., Griebel, M., Schweitzer, M.A., Telea, A.: Feature sensitive multiscale editing on surfaces. The Visual Computer **20**(5), 329–343 (2004)
3. Cornea, N.D., Silver, D., Min, P.: Curve-skeleton properties, applications and algorithms. IEEE Transactions on Visualization and Computer Graphics **13**(3), 530–548 (2007)
4. Cornea, N.D., Silver, D., Yuan, X., Balasubramanian, R.: Computing hierarchical curve-skeletons of 3d objects. The Visual Computer **21**(11), 945–955 (2005)
5. Costa, L.F., Cesar Jr, R.M.: Shape analysis and classification. CRC Press (2001)
6. Dey, T.K., Sun, J.: Defining and computing curve-skeletons with medial geodesic function. In: Proc. of Eurographics Symposium on Geometry Processing, pp. 143–152 (2006)
7. Giblin, P., Kimia, B.B.: A formal classification of 3d medial axis points and their local geometry. IEEE Trans. on Pattern Analysis and Machine Intelligence **26**(2), 238–251 (2004)
8. Hart, P.E., Nilsson, N.J., Raphael, B.: A formal basis for the heuristic determination of minimum cost paths. IEEE Trans. on Systems Science and Cybernetics **4**(2), 100–107 (1968)
9. Katz, S., Leifman, G., Tal, A.: Mesh segmentation using feature point and core extraction. The Visual Computer **21**, 649–658 (2004)
10. Katz, S., Tal, A.: Hierarchical mesh decomposition using fuzzy clustering and cuts. ACM Transactions on Graphics **22**(3), 954–961 (2003)
11. Kiryati, N., Székely, G.: Estimating shortest paths and minimal distances on digitized three-dimensional surfaces. Pattern Recognition **26**, 1623–1637 (1993)
12. Kong, T.Y., Rosenfeld, A.: Digital topology: Introduction and survey. Computer Vision, Graphics, and Image Processing **48**, 357–393 (1989)
13. Lee, Y., Lee, S., Shamir, A., Cohen-Or, D., Seidel, H.P.: Mesh scissoring with minima rule and part salience. Computer Aided Geometric Design **22**, 444–465 (2005)
14. Li, X., Woon, T.W., Tan, T.S., Huang, Z.: Decomposing polygon meshes for interactive applications. In: Proc. of symposium on Interactive 3D graphics, pp. 35–42 (2001)
15. Mullikin, J.C.: The vector distance transform in two and three dimensions. CVGIP: Graphical Models and Image Processing **54**(6), 526–535 (1992)
16. Pizer, S.M., Siddiqi, K., Székely, G., Damon, J.N., Zucker, S.W.: Multiscale medial loci and their properties. Int. Journal of Computer Vision **55**(2-3), 155–179 (2003)
17. Reniers, D., Telea, A.: Skeleton-based hierarchical shape segmentation. In: Proc. of the IEEE Int. Conf. on Shape Modeling and Applications (SMI'07), pp. 179–188 (2007)
18. Reniers, D., Telea, A.: Segmenting simplified surface skeletons (2008). Accepted for the 14th IAPR Int. Conf. on Discrete Geometry for Computer Imagery
19. Reniers, D., Telea, A.: Tolerance-based feature transforms. In: Advances in Computer Graphics and Computer Vision, vol. 4, pp. 187–200. Springer Berlin Heidelberg (2008)
20. Reniers, D., Van Wijk, J.J., Telea, A.: Computing multiscale curve and surface skeletons of genus 0 shapes using a global importance measure. IEEE Transactions on Visualization and Computer Graphics **14**(2), 355–368 (2008)
21. Shamir, A.: A formulation of boundary mesh segmentation. In: Proc. of the Second Int. Symp. on 3D Data Processing, Visualization and Transmission (3DPVT'04), pp. 82–89 (2004)
22. Shilane, P., Min, P., Kazhdan, M., Funkhouser, T.: The Princeton shape benchmark. In: Shape Modeling International (2004)
23. Sumner, R.W., Popovic, J.: Deformation transfer for triangle meshes. ACM Transactions on Graphics **23**(3) (2004)
24. West, D.B.: Introduction to Graph Theory. Prentice Hall (1996)



Dennie Reniers received his MSc in Computer Science from the Eindhoven University of Technology (TU/e), the Netherlands, in 2004. Currently, he is a PhD student in the Department of Mathematics and Computer Science at TU/e. His research interests include shape analysis, representation and segmentation, and discrete geometry.

Alexandru Telea received his Master's Degree from the Polytechnics University of Bucharest, Romania (1996), and his PhD in Computer Science from the Eindhoven University of Technology, the Netherlands (2000). Between 2000-2007, he worked as an assistant professor in visualization at the Eindhoven University. He is currently an adjunct professor at the Scientific Visualization and Computer Graphics group of the University of Groningen. His research interests are in multiscale modeling and analysis for scientific and information visualization, shape analysis and representation, and component and object-oriented software architectures.