Segmenting Simplified Surface Skeletons

Dennie Reniers[1] and Alexandru Telea[2]

[1] Department of Mathematics and Computer Science
Eindhoven University of Technology, Eindhoven, The Netherlands
d.reniers@tue.nl
[2] Institute for Mathematics and Computing Science
University of Groningen, Groningen, The Netherlands
a.c.telea@rug.nl

Abstract. A novel method for segmenting simplified skeletons of 3D shapes is presented. The so-called simplified Y-network is computed, defining boundaries between 2D sheets of the simplified 3D skeleton, which we take as our skeleton segments. We compute the simplified Y-network using a robust importance measure which has been proved useful for simplifying complex 3D skeleton manifolds. We present a voxel-based algorithm and show results on complex real-world objects, including ones containing large amounts of boundary noise.

1 Introduction

The skeleton, also known as medial axis, is a compact shape descriptor. The skeleton is of a lower dimensionality than the shape it describes, and makes the shape's structure, such as its topology and articulation, more explicit. Hence, it is a useful tool in a wide range of computer vision and visualization applications, such as shape analysis, recognition, shape alignment, motion planning, and collision detection. One way of analyzing the structure of the skeleton is by segmenting it in its logical parts. To give just one of many examples, Zhang *et al.* use such a skeleton segmentation as a step in their 3D model retrieval pipeline [1]. Besides the explicit capture of a shape's logical components, skeletons offer pose invariance under many types of shape deformation. However, skeletons are notoriously unstable to small boundary perturbations, which affects the segmentation robustness. What is needed is a robust segmentation of a simplified skeleton, i.e., a skeleton from which spurious, small-scale, parts are removed.

In this paper, we present a voxel-based algorithm for segmenting simplified skeletons of 3D shapes into disjoint segments. A 3D skeleton consists of a set of compact 2D manifolds, called *sheets*. A sheet boundary consists of 1D curves which are either part of the 3D skeleton's boundary, or curves where at least three sheets intersect. Given this property, the sheet intersection curves are also called *Y-curves* [2]. Our aim is to robustly segment a noisy 3D skeleton in its sheets.

We compute a robust simplified skeleton using [3], and extend this method to compute the simplified network of Y-curves by combining information from two different simplification levels. Our segmentation, based on this simplified Y-network, can handle objects with large amounts of noise on the boundary, without having to prune the segmentation afterward. Besides the segmentation itself, the simplified Y-network is

also a useful result of our approach, which can be used in shape analysis tasks. We demonstrate our voxel-based algorithm on several real-world examples, and show that our approach can handle objects containing large amounts of noise on the boundary.

As far as we know, this is the first segmentation method specifically designed for simplified 3D skeletons. Existing surface-segmentation approaches, such as the well-known topological segmentation for discrete surfaces of Malandain et al. [4], fail on our simplified skeletons, as we show in Sec. 5. To achieve a better segmentation, we consider not only the skeleton itself, but also its relation to the object boundary.

The outline of this paper is as follows. Section 2 introduces the definition of the skeleton and details on its structure. Section 3 briefly outlines our previous work on simplified skeletons necessary for a good understanding of the following material. Section 4 details our novel method for computing the simplified Y-network Y_τ, at simplification level τ, and the decomposition of Y_τ into its constituent Y-curves. In Section 5, we extend Y_τ using information from a less-simplified Y-network Y_υ to correctly and robustly segment the simplified skeleton at level of detail τ. Section 6 presents and discusses results. Section 7 concludes this paper.

2 Preliminaries

2.1 Skeleton Definition

Let Ω^d be a d-dimensional shape with boundary $\partial\Omega$. Let $D : \Omega \to \mathbb{R}^+$ be the distance transform, assigning to each object point the minimum Euclidean distance to the object's boundary. Let $F : \Omega \to \mathcal{P}(\partial\Omega)$ be the feature transform [5] (where $\mathcal{P}$ is the power set), assigning to each object point the set of boundary points at minimum distance:

$$F(p \in \Omega) = \left\{ q \in \partial\Omega \;\middle|\; \|p - q\| = D(p) \right\}, \tag{1}$$

where $\|\cdot\|$ denotes Euclidean distance. The *skeleton* $\mathcal{S}$ of Ω can be defined in multiple equivalent ways: as the locus of centers of maximally inscribed balls, as the singularities of D, or as those object points having at least two feature points. We use the latter definition:

$$\mathcal{S}(\Omega) = \left\{ p \in \Omega \;\middle|\; |F(p)| \geq 2 \right\}. \tag{2}$$

This definition can be used both in 2D and 3D. In 3D, $\mathcal{S}$ is also called the medial surface, or *surface skeleton*, to distinguish it from the curve skeleton, or centerline [6]. It has been proved that the skeleton is homotopic to the original shape in any dimension [7].

2.2 Skeleton Structure and Segment Definition

It is well known that the skeleton of a 3D object consists of manifolds, called *sheets*, that intersect in curves, called *Y-curves* [2]. A sheet's boundary consists of Y-curves and/or skeleton boundary curves. In addition to the above sheet boundary curves, a 3D skeleton can also contain isolated curves in some degenerate cases, such as a cylinder. We assume for the time being that the skeleton contains no such degeneracies. The set of Y-curves is called the *Y-network*, denoted $Y = \{y_1, ..., y_n\}$. In the following,

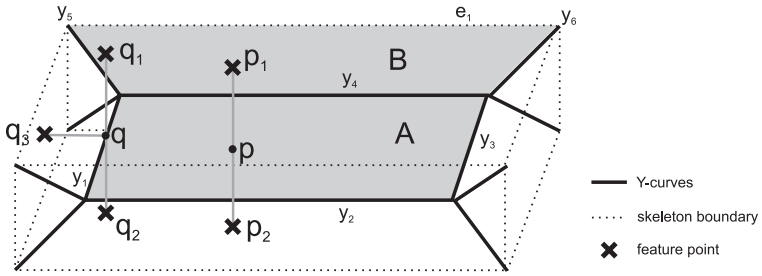


Fig. 1. Skeleton structure of a box. Point p is a sheet point having two feature points p_1, p_2, point q is a Y-curve point having three feature points q_1, q_2, q_3.

a *skeleton segment* is equivalent to a sheet. For example, Fig. 1 shows the skeleton of a box. This contains 12 skeleton boundary curves (dotted lines), and a Y-network containing 12 Y-curves (thick lines), yielding 13 sheets, or segments. For illustration, segment A is bounded by Y-curves $y_1 ... y_4$, whereas B is bounded by Y-curves y_4, y_5, y_6 and skeleton boundary curve e_1.

Skeleton points can be classified by the number of feature points they have [8]. A skeleton boundary-curve point has an infinite amount of feature points, because the inscribed ball at such a point has a contiguous arc of contact points with $\partial\Omega$. A sheet point p has two feature points, $|F(p)| = 2$. Within a sheet, the associated feature points evolve smoothly over the object boundary. That is, for a point $p + \Delta p$ close to p, the feature points $F(p + \Delta p)$ neighbor the feature points $F(p)$ of p. Finally, a Y-curve point q, where three sheets intersect, has three feature points, $|F(q)| = 3$. Each sheet contributes two feature points, but each feature point is shared with one of the two other sheets. Fig. 1 illustrates such a sheet point p and Y-curve point q and their respective feature points (crosses).

3 Simplified Skeletons

Following from Eq. 2, skeletons are inherently sensitive to small boundary perturbations [9]. Boundary noise, for example due to sampling or acquisition artifacts, produces spurious skeleton parts. To handle this, some skeletonization methods define an importance measure $\rho : \mathcal{S} \rightarrow \mathbb{R}^+$ indicating the importance for each skeleton point to the shape representation. Together with a subsequent pruning strategy, this delivers a simplified skeleton [10]. Well-known measures are the distance and angle between feature points [11]. The problem with these measures is that they are non-monotonic, which means that a non-trivial, often non-intuitive, pruning strategy is needed to enforce connectedness of the simplified skeleton, which is a desirable property as the skeleton should be connected (if the original object is connected). A frequent step of such a pruning strategy is an erosion step (e.g. [12]). To eliminate these complications, a monotonic importance measure can be used, so that simply thresholding the measure delivers a connected skeleton. In 2D, this is achieved by the monotonic boundary-distance measure, which takes as importance for a point the smaller distance along the boundary

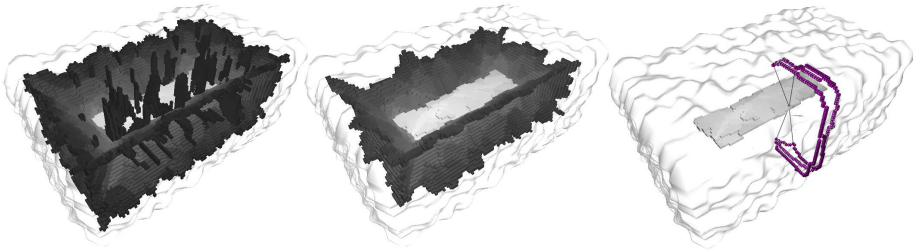


Fig. 2. Simplified skeletons $\mathcal{S}_{\tau=10}, \mathcal{S}_{\tau=20}, \mathcal{S}_{\tau=70}$ (from left to right) of a noisy box. The intensity encodes the importance measure ρ. In the right image a voxel was selected which has 4 feature voxels, resulting in 6 paths.

between its two feature points [13–15]. A robust, progressively simplified, connected skeleton is obtained simply by thresholding this measure with increasing values.

In [3], we extended the boundary-distance measure to 3D, and obtained the first monotonic importance measure for 3D skeletons. We assign to each point $p \in \mathcal{S}$ the length of the shortest geodesic on the surface $\partial\Omega$ between the two feature points $F(p)$. This measure is continuous on sheets, over which the feature points evolve smoothly, may contain jumps at Y-curves (cf. Fig. 2), and has a local maximum ridge that forms a 1D connected structure on $\mathcal{S}$. This last property has been shown in [16] and was used to formally define a curve skeleton.

The next section explains how we compute our measure in a discrete setting, a step which is needed prior to computing the simplified Y-network (Sec. 4).

3.1 Computing Simplified Skeletons

Most skeletonization methods work on discretized objects, using e.g. a boundary sampling [13] or a volumetric sampling on a regular voxel grid $\mathbb{Z}^3$. We take the latter, voxel-based, approach. Def. 1 and 2 for the continuum have to be adapted accordingly.

The object Ω, its boundary $\partial\Omega$, and skeleton $\mathcal{S}$ are represented as sets of voxels. We compute the feature transform of Ω by the raster-scanning approach of [17]. Voxelization introduces the problem that when placing a discretized inscribed ball at a skeleton voxel, this ball does not always touch the boundary exactly in two voxels: It might touch it only in one [5]. Thus, we *extend* the feature transform by merging the feature set of a voxel $p = (p_x, p_y, p_z)$ with the feature sets of p's 26-neighbors in the first octant:

$$\overline{F}(p \in \Omega) = \bigcup_{\Delta x, \Delta y, \Delta z \in \{0,1\}} F(p_x + \Delta x, p_y + \Delta y, p_z + \Delta z)\,. \tag{3}$$

In the rest of this paper, the extended feature transform will simply be referred to as feature transform. Hence, skeletons based on $\overline{F}$ can be up to two voxels thick.

In order to compute our importance measure ρ for a voxel $p \in \Omega$, we compute the *shortest-path set* Γ, by computing between each two feature voxels a, b the shortest

path $\gamma(a,b)$, as discrete equivalent to the shortest geodesic, on the boundary $\partial\Omega$:

$$\Gamma(p) = \bigcup_{a,b \in \overline{F}(p)} \gamma(a,b) \; . \tag{4}$$

Shortest paths are computed as 26-connected voxel chains using Dijkstra's algorithm on the boundary graph, in which the voxels $\partial\Omega$ are the nodes, and the 26-neighborhood relations represent the edges. By caching computed paths, we avoid computing the same path twice, and thereby accelerate the method, as detailed in [3].

The importance measure ρ is now defined for each object voxel p as the maximum shortest-path length in $\Gamma(p)$:

$$\rho(p \in \Omega) = \max_{\gamma \in \Gamma(p)} \|\gamma\|, \tag{5}$$

where $\|\gamma\|$ denotes the length of path γ, computed using the geodesic length estimator of [18]. In Figure 2 (right), the shortest-path set Γ containing 6 paths for a voxel with 4 feature voxels is shown. The resulting ρ is robust as it maximizes path length.

Finally, the simplified skeleton $\mathcal{S}_\tau$ is defined by imposing a threshold τ on ρ:

$$\mathcal{S}_\tau(\Omega) = \{p \in \Omega \mid \rho(p) > \tau\} \; . \tag{6}$$

The threshold τ functions as a continuous scale-parameter controlling the simplification level. Small τ values eliminate less important skeleton parts that are due to small-scale surface features such as noise. Larger τ values can be used to retain the most salient parts of the skeleton. Thresholding ρ combines both skeleton detection *and* simplification in a single step. Experimental studies show that for $\tau \geq 5$ all non-skeleton voxels are pruned. Figure 2 illustrates the use of τ for a noisy box. Whereas $\mathcal{S}_{\tau=10}$ (left) contains some spurious sheets. $\mathcal{S}_{\tau=20}$ (middle) can be considered robust, as it contains 13 sheets like a non-noisy box. In $\mathcal{S}_{\tau=70}$ (right), only the center sheet is retained, which can be seen as a coarse scale representation of the box.

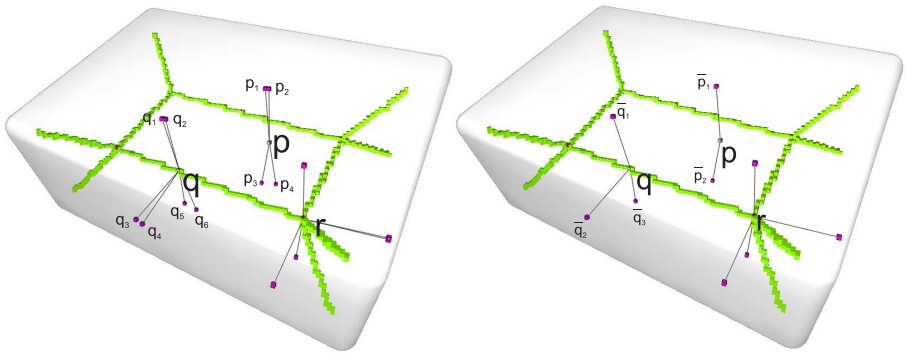
The simplified skeleton is homotopic to the original object as long as the local maximum ridge in the thresholded ρ forms a connected structure. For completeness, we note that an alternative definition of ρ (Eq. 5) can be formulated on this ridge in order to obtain a monotonic measure for *all* skeleton points [3, 19], so that the simplified skeleton is homotopic for every τ. However, this is outside this paper's scope.

4 The Simplified Y-network

In this section, we show how we extend the previous work on simplified skeletons to compute a simplified Y-network, on which our skeleton segmentation is based.

4.1 Computing the Y-network

In order to find the Y-network, i.e. the sheet-intersection curves, of a simplified skeleton $\mathcal{S}_\tau$ we must check if a voxel is on a Y-curve or not. In the continuous $\mathbb{R}^3$ space, an intersection curve point has (at least) three feature points: $|F| \geq 3$ (Sec. 2.2). However,



(a) The feature transform $\overline{F}$. (b) The simplified feature transform $\overline{F}_\tau$.

Fig. 3. A (non-axis aligned) box with its detected network $Y_{\tau=10}$. Three selected points p, q, r: a sheet point p, a Y-curve point q, and a Y-curve intersection r.

as indicated in Section 3, the feature transform $\overline{F}(p)$ for a skeleton voxel p consists of several voxels. For example, $\overline{F}(p)$ in Fig. 3(a) contains 4 feature voxels $p_1 ... p_4$ and $\overline{F}(q)$ contains 6 voxels: $q_1..q_6$. If we naively use the cardinality of $\overline{F}$ to detect the Y-curves, then both p and q would be selected, which is wrong for p. To solve this problem, we group each two feature points that have a small geodesic distance (shortest-path length) on the boundary. More formally, we define an equivalence relation $a \sim b$ on $\overline{F}$:

$$a \sim b \Leftrightarrow \|\gamma(a,b)\| < \tau \ , \tag{7}$$

where τ is the same threshold we used to simplify the skeletons (Eq. 6). This relation gives rise to a number equivalence classes. We now replace $\overline{F}$ by the *simplified feature transform* $\overline{F}_\tau$, defined as a set of class representatives following Eq. 7, i.e. a subset of $\overline{F}$ containing one point from each equivalence class. Which particular point we choose in a class is not important, as we are only interested in the cardinality of $\overline{F}_\tau$.

Using the simplified feature transform $\overline{F}_\tau$, we replace the definition of $\mathcal{S}_\tau$ in Eq. 6 with:

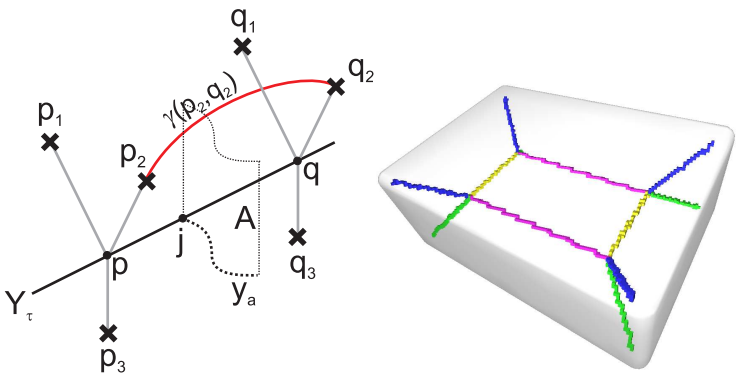
$$\overline{\mathcal{S}}_\tau(\Omega) = \left\{ p \in \Omega \;\middle|\; |\overline{F}_\tau(p)| \geq 2 \right\}, \tag{8}$$

which more closely parallels Eq. 2. In $\overline{\mathcal{S}}_\tau$ all sheet points p that have a shortest path between their two feature points that is shorter than τ, or in other words, that have a lower importance than τ, are pruned.

Now, the *simplified Y-network* is straightforwardly defined as:

$$Y_\tau = \left\{ p \in \Omega \;\middle|\; |\overline{F}_\tau(p)| \geq 3 \right\} , \tag{9}$$

which is a subset of $\overline{\mathcal{S}}_\tau$. For a Y-curve point q, where three sheets meet in q's neighborhood, this means that if one of the sheets in its neighborhood is pruned because its importance is lower than τ, the point is not considered a Y-curve point in Y_τ. It is important to note that the simplified Y-network is not simply a post-processing of the simplified skeleton. Instead, the Y-network is computed directly out of the shape, using



(a) Two points p, q on a Y-network. (b) The detected Y-curves of a box.

Fig. 4. Y-network decomposition into Y-curves.

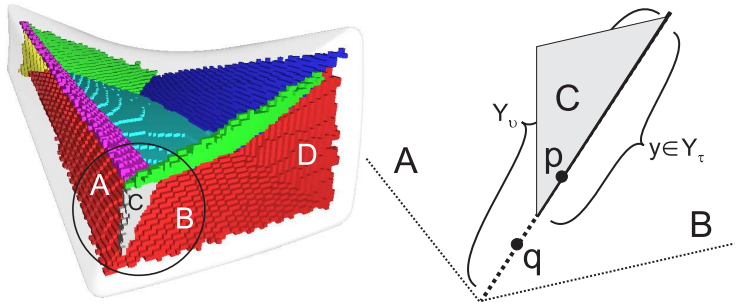
the integral quantity of geodesic distance on the object boundary. This is more stable than extracting the sheet-intersection curves from the voxelized skeleton, and also offers a natural scale parameter.

Fig. 3 shows a non-axis aligned box with its simplified network Y_τ, $\tau = 10$. Like in the continuous case, the discrete Y-network forms a connected structure. Three points p, q, r are selected: a sheet point p, a Y-curve point q, and a Y-curve intersection point r. Their feature voxels are connected to the corresponding points by line segments. Fig. 3(a) uses the non-simplified feature transform $\overline{F}$ for the selected points, while Fig. 3(b) uses the simplified feature transform $\overline{F}_\tau$. We see the merit of $\overline{F}_\tau$: p and q can be classified as a sheet and a Y-curve voxel respectively based on the cardinality $|\overline{F}_\tau|$, but not on $|\overline{F}|$. For point q for instance, $\overline{F}_\tau$ gives us exactly three feature points $\overline{q}_1, \overline{q}_2, \overline{q}_3$, i.e. the representations of the classes $\{q_1, q_2\}, \{q_3, q_4\}, \{q_5, q_6\}$.

4.2 Y-network Decomposition

Although Eq. 9 enables the detection of the Y-network voxels, it does not make explicit the structure of the Y-network as a collection of Y-curves. This section presents how to compute such a decomposition. The next section shows how to use the decomposition to compute a skeleton segmentation.

To decompose the network Y_τ into its n Y-curves $\{y_1, ..., y_n\}$, we define two points $p, q \in Y_\tau$ to be on the same Y-curve when there is no junction in the Y-network between them. For illustration, Figure 4(a) sketches (part) of a Y-network. Let p, q be two points on Y_τ, each one having three feature points, with the indexes p_1, p_2, p_3 and q_1, q_2, q_3 chosen in such a way that they are "aligned", that is, the sum of the geodesic distances $\sum_i \|\gamma(p_i, q_i)\|$ is minimal. Suppose now there is a junction j between p, q, due to a Y-curve y_a as shown (dotted) in the figure. Such a Y-curve y_a results from a sheet A with a local importance higher than τ, otherwise y_a would have been pruned and would not be present in Y_τ. Thus, we can detect whether there is a junction j between p and q by looking at the geodesic distance $\|\gamma\|$ between the feature points of p and q. Because



(a) Sheets A, B and D are wrongly in the same segment.

(b) Close-up of (a).

Fig. 5. Segmented simplified skeleton of the Deformed Box using Y_τ.

the feature points evolve smoothly within a skeletal sheet, we argue that if there is a Y-curve y_a between p, q due to sheet A, the geodesic distance between the feature points p_2 and q_2 must be larger than τ, because A's importance is at least τ. Generalizing, two points $p, q \in Y_\tau$ belong to the same Y-curve y if and only if all three geodesic distances between each pair of feature points (p_i, q_i) are smaller than τ:

$$p, q \in y \in Y_\tau \Leftrightarrow \forall_{i \in \{1,2,3\}} \|\gamma(p_i, q_i)\| \leq \tau \; . \tag{10}$$

One remark is due. Voxels where several Y-curves come together have more than three feature points (see e.g. Fig. 3(b), point r). To handle these cases, Eq. 10 is applied for all subsets of both p and q, having 3 feature points. One might ask why we do not just detect the Y-curve intersection points and use these to separate the Y-curves, which is more trivial. The first reason is that our approach is of sub-voxel precision: a single voxel may contain multiple Y-curves. Second, our method does not use a topological analysis of the Y-network voxels and is thus more accurate in cases where Y-curves meet under small angles. Figure 4(b) shows the segmentation of the Y-network of a box obtained using Eq. 10. The Y-curves are distinctively colored[3].

5 Skeleton Segmentation

Although it seems natural to use the simplified network Y_τ to segment the skeleton $\overline{\mathcal{S}}_\tau$, both computed at the same scale τ, this can sometimes deliver unexpected results. Figure 5(a) shows the segmentation produced for the skeleton $\overline{\mathcal{S}}_\tau$ ($\tau = 10$) of a twisted box. We expect 13 segments, similar to the skeleton segmentation of a non-deformed box (Sec. 2.2). Yet, for the deformed box some segments get merged inadvertently. For example, the skeleton sheets A, B and D are incorrectly merged, whereas they should be distinct segments as for a non-deformed box.

Figure 5(b) is a schematic close-up of the situation. The problem is that the Y-curve $y \in Y_\tau$ does not extend all the way to the skeleton boundary. The missing part is

[3] Please view the images in color

indicated by the dotted line. A narrow tunnel connects the areas A and B on the skeleton manifold, so that they end up in the same segment. The reason that y is too short is that sheet C is simplified for lower values of τ than A and B at the dotted line segment. The cardinality of the simplified feature transform for these points is 2, so that point q, for example, is not detected as a Y-curve point by Eq. 9. In other words, only two sheets are found to come together at q in the *simplified* skeleton, namely A and B. For the same reason, other surface segmentation approaches based on local topology, e.g. [4], would fail segmenting this *simplified* skeleton. Note that it would also fail because our skeletons are two voxels thick.

We solve the issue noticing that curve y in Fig. 5(b) would be longer for less simplified skeletons. For non-simplified skeletons, we would not have the problem at all. Hence, to segment a skeleton $\overline{\mathcal{S}}_\tau$ we use a less simplified Y-network Y_υ, $\upsilon < \tau$, which contains longer, extended, versions of the Y-curves which ended prematurely in Y_τ. However, we must be careful to only extend Y-curves from Y_τ, and not to incorporate *any* Y-curves that only occur in Y_υ. Hence, we only consider those Y-curves $y \in Y_\upsilon$ for which there is at least one point $p \in y$ in Y_τ. We call this set the extension of Y_τ using Y_υ, denoted as $Y_{\tau,\upsilon}$:

$$Y_{\tau,\upsilon} = \left\{ y \in Y_\upsilon \;\middle|\; \exists_{p \in y}\, p \in Y_\tau \right\}. \tag{11}$$

Finally, we compute the decomposition of $Y_{\tau,\upsilon}$ into its respective Y-curves by taking both scales τ and υ into account and adapting Eq. 10 accordingly:

$$p, q \in y \in Y_{\tau,\upsilon} \Leftrightarrow \forall_{i \in \{1,2,3\}} \|\gamma(p_i, q_i)\| \leq \begin{cases} \tau & \text{if } p \in Y_\tau \wedge q \in Y_\tau \\ \upsilon & \text{otherwise.} \end{cases} \tag{12}$$

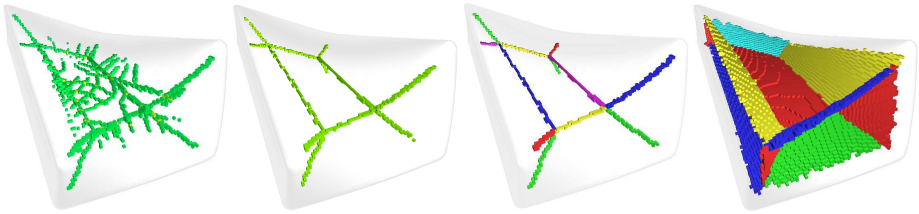
We have not yet specified the value of υ. For all objects we tested, a fixed value of $\upsilon = 5$ gives very good results. As explained in Sec. 3, this value essentially robustly yields the entire skeleton of a 3D shape.

After computing $Y_{\tau,\upsilon}$ we can segment $\overline{\mathcal{S}}_\tau$ as follows. We consider the skeleton $\overline{\mathcal{S}}_\tau$ as a 26-connected graph, from which we remove the voxels occupied by $Y_{\tau,\upsilon}$, and then determine the connected components in the remaining skeleton graph using e.g. a flood-fill. Since $\overline{\mathcal{S}}_\tau$ can be up to two voxels thick [3] and the Y-network is only one voxel thick, we first dilate the Y-network by 1 voxel in each 26-direction, before removing them from the skeleton graph. Hereafter we erode the dilated Y-network so that these voxels are also part of a segment. Clearly, many other alternative implementations are possible once we have a robust Y-network.

Figure 6 shows Y_υ, Y_τ, $Y_{\tau,\upsilon}$ (decomposed in its Y-curves), and the segmentation based on $Y_{\tau,\upsilon}$ respectively, which is a correct segmentation of the deformed box skeleton as opposed to Fig. 5(a).

6 Results and Discussion

We have tested our method on shapes of varying complexity and amount of boundary noise. As input objects we used 3D triangle meshes, voxelized using binvox (http://www.cs.princeton.edu/~min/binvox/) in various resolutions ranging up to two million



(a) $Y_\upsilon, \upsilon = 5$ (b) $Y_\tau, \tau = 10$ (c) $Y_{\tau,\upsilon}$ (d) segmentation

Fig. 6. Correctly segmenting the simplified skeleton of a deformed box.

object voxels. For all images in this paper, the original object mesh representation is rendered instead of its voxel representation for nicer display.

Figure 7 shows a selection of the results. For each object, τ is empirically chosen based on the noise level of the object, and we show both the extended simplified network $Y_{\tau,\upsilon}$, decomposed into its Y-curves (top image), and the segmentation of the simplified skeleton (bottom image) using $Y_{\tau,\upsilon}$. In Figures 7(a) and 7(e), we added 5-10% noise to show the robustness of our approach.

We highlight some of the results. For the Noisybox, we see that our method correctly detects 13 segments and 12 Y-curves. For the Dinosaur and Noisydino, the four legs and feet are all assigned different segments. Our method correctly segments the skeleton of the Dodecahedron (Fig. 7(f)). This is a difficult skeleton to segment as it contains degeneracies, in the sense that each Y-curve actually separates five sheets instead of the usual three, which is why the Y-network is thicker than for the other objects.

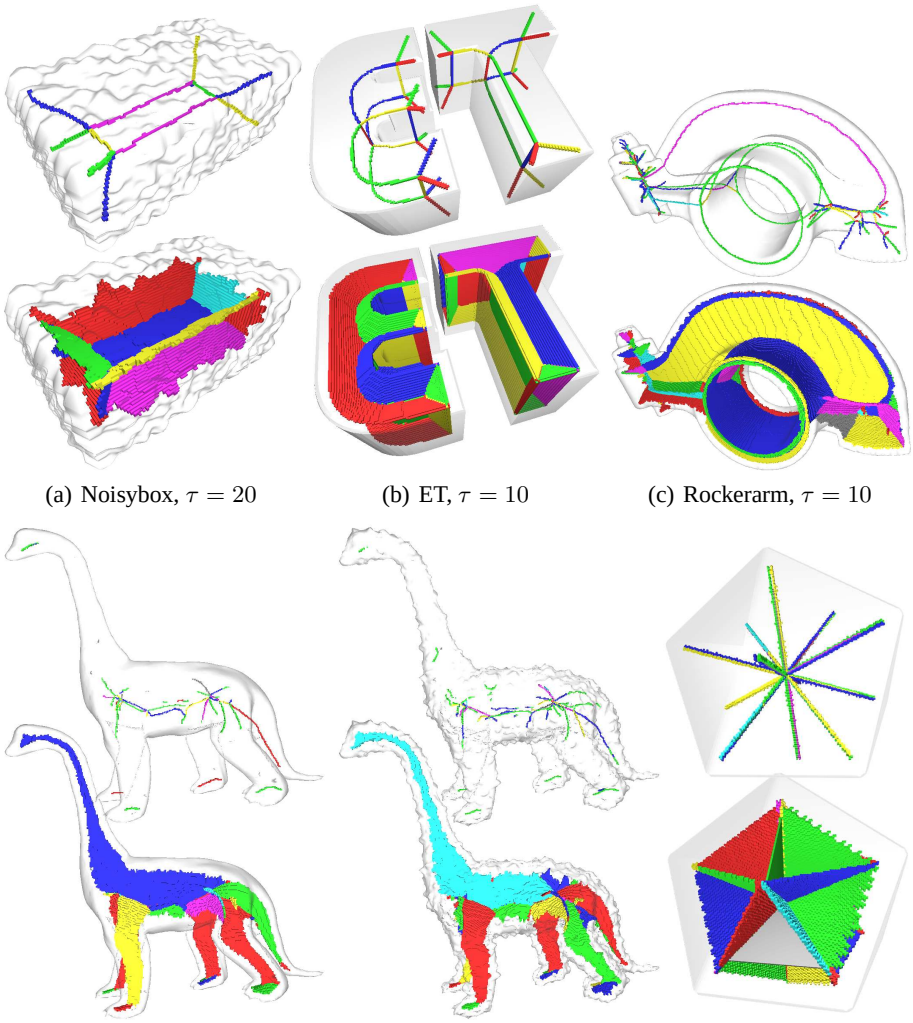
Table 1 shows timing measurements performed on a Pentium IV 3 GHz, with 1 GB of RAM for all objects in this paper. Columns "dim", "$|\Omega|$", and $|\partial\Omega|$ denote the grid resolution, and the amount of object and boundary voxels respectively. Columns "$\overline{\mathcal{S}}$ time" and "segm. time" denote the wall-clock time for computing the skeleton and segmentation respectively. The most time is spent computing shortest paths in the boundary graph, needed in both stages.

7 Conclusion

We have presented a voxel-based approach for robustly segmenting simplified skeletons of 3D shapes, based on the simplified feature transform and simplified Y-network. The

Table 1. Timing measurements

Object	dim	$\lvert\Omega\rvert$	$\lvert\partial\Omega\rvert$	$\overline{\mathcal{S}}$ time (s)	segm. time (s)
Deformed Box	65x64x124	247k	24k	21s	11s
Dodecahedron	124x124x124	945k	39k	48s	17s
ET	125x78x173	1.045k	93k	304s	37s
Noisydino	99x325x365	1.128k	100k	62s	33s
Rockerarm	366x188x112	2.000k	151k	470s	62s



(a) Noisybox, $\tau = 20$ (b) ET, $\tau = 10$ (c) Rockerarm, $\tau = 10$

(d) Dino, $\tau = 16$ (e) Noisydino, τ=16 (f) Dodecahedron, τ=10

Fig. 7. Results. Y-network decomposition (top images), $\overline{\mathcal{S}}_\tau$ segmentation (bottom images).

Y-network is decomposed into its respective Y-curves. The simplified Y-network could prove to be more useful in certain shape analysis or retrieval tasks than the segmentation itself, because the Y-curves change more continuously under shape deformations than the segmentation does.

Our entire method relies upon the choice of a single parameter τ, which controls the simplification of the skeleton [3], but *not* the segmentation itself, thereby yielding a fully autonomous segmentation method. One limitation of our current implementation is that cylindrical object-parts having degenerate curve skeletons may locally result in

over-segmentation. Second, the two-voxel thickness of the skeletons might yield undesirable topological changes in the skeleton for too complex objects, when compared to thin skeletons.

In future work, we would like to investigate whether our skeleton segmentation can be used to achieve a robust patch-type segmentation of the object surface. As opposed to traditional methods acting only on the surface, a skeleton-based segmentation has the benefit that it captures a sense of symmetry, and is potentially more robust, being based on the object's volume, not on its surface.

References

1. Zhang, J., Siddiqi, K., Macrini, D., Shokoufandeh, A., Dickinson, S.: Retrieving articulated 3-d models using medial surfaces and their graph spectra. In: Int. Workshop On Energy Minimization Methods in Computer Vision and Pattern Recognition. (2005) 285–300
2. Damon, J.: Global medial structure of regions in R^3. Geometry and Topology **10** (2006) 2385–2429
3. Reniers, D., Van Wijk, J.J., Telea, A.: Computing multiscale curve and surface skeletons of genus 0 shapes using a global importance measure. IEEE TVCG **14**(2) (2008) 355–368
4. Malandain, G., Bertrand, G., Ayache, N.: Topological segmentation of discrete surfaces. International Journal of Computer Vision **10**(2) (1993) 183 – 197
5. Reniers, D., Telea, A.: Tolerance-based feature transforms. In: Advances in Computer Graphics and Computer Vision. Volume 4., Springer Berlin Heidelberg (2008) 187–200
6. Cornea, N., Silver, D., Min, P.: Curve-skeleton properties, applications and algorithms. IEEE Transactions on Visualization and Computer Graphics **13**(3) (2007) 530–548
7. Lieutier, A.: Any open bounded subset of $\mathbb{R}^3$ has the same homotopy type as its medial axis. In: Proc. of the 8th ACM symposium on Solid modeling and applications. (2003) 65–75
8. 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) (2004) 238–251
9. Chazala, F., Lieutier, A.: The λ-medial axis. Graphical Models **67**(5) (2005) 304–331
10. Shaked, D., Bruckstein, A.M.: Pruning medial axes. Computer Vision and Image Understanding **69**(2) (1998) 156–169
11. Malandain, G., Fernández-Vidal, S.: Euclidean skeletons. Image and Vision Computing **16**(5) (1998) 317 – 327
12. Siddiqi, K., Bouix, S., Tannenbaum, A., Zucker, S.W.: The hamilton-jacobi skeleton. In: Proc. of the Int. Conference on Computer Vision (ICCV'99). (1999) 828–834
13. Ogniewicz, R.L., Kübler, O.: Hierarchic voronoi skeletons. Pattern Recognition **28**(3) (1995) 343–359
14. Costa, L.F., Cesar, Jr, R.M. In: Shape analysis and classification. CRC Press (2001) 416–419
15. Telea, A., Van Wijk, J.J.: An augmented fast marching method for computing skeletons and centerlines. In: Proc. of the Symposium on Data Visualisation (VisSym'02). (2002) 251–259
16. Dey, T.K., Sun, J.: Defining and computing curve-skeletons with medial geodesic function. In: Proc. of Eurographics Symposium on Geometry Processing. (2006) 143–152
17. Mullikin, J.C.: The vector distance transform in two and three dimensions. CVGIP: Graphical Models and Image Processing **54**(6) (1992) 526–535
18. Kiryati, N., Székely, G.: Estimating shortest paths and minimal distances on digitized three-dimensional surfaces. Pattern Recognition **26** (1993) 1623–1637
19. Reniers, D., Telea, A.: Skeleton-based hierarchical shape segmentation. In: Proc. of the IEEE Int. Conf. on Shape Modeling and Applications (SMI'07). (2007) 179–188