EUROGRAPHICS 2007 / D. Cohen-Or and P. Slavík
(Guest Editors)

Volume 26 (2007), Number 3

On-the-fly Curve-skeleton Computation for 3D Shapes

Andrei Sharf[1] and Thomas Lewiner[2] and Ariel Shamir[3] and Leif Kobbelt[4]

[1] School of Computer Science, Tel Aviv University
[2] Departament of Mathematics, PUC—Rio de Janiero
[3] Efi Arazi School of Computer Science, The Interdisciplinary Center, Herzliya
[4] Computer Graphics Group, RWTH Aachen

Abstract

The curve-skeleton of a 3D object is an abstract geometrical and topological representation of its 3D shape. It maps the spatial relation of geometrically meaningful parts to a graph structure. Each arc of this graph represents a part of the object with roughly constant diameter or thickness, and approximates its centerline. This makes the curve-skeleton suitable to describe and handle articulated objects such as characters for animation. We present an algorithm to extract such a skeleton on-the-fly, both from point clouds and polygonal meshes. The algorithm is based on a deformable model evolution that captures the object's volumetric shape. The deformable model involves multiple competing fronts which evolve inside the object in a coarse-to-fine manner. We first track these fronts' centers, and then merge and filter the resulting arcs to obtain a curve-skeleton of the object. The process inherits the robustness of the reconstruction technique, being able to cope with noisy input, intricate geometry and complex topology. It creates a natural segmentation of the object and computes a center curve for each segment while maintaining a full correspondence between the skeleton and the boundary of the object.

Categories and Subject Descriptors (according to ACM CCS): I.3.3 [Computer Graphics]: Curve Generation

1. Introduction

Computer graphics applications handle huge models representing complex 3D objects. The most common representations for such objects are boundary meshes or point-sets. However, applications such as editing, animation, morphing or shape matching often need a higher level understanding of the shape and its structure. Such an understanding can be conveyed through the use of an inner curve-skeleton for the object. This geometrical and topological abstraction represents the 3D shape on an object, by mapping the relation of geometrically meaningful parts to a graph structure with univariate arcs. Each arc represents one of these meaningful parts by its centerline, assuming it has roughly constant diameter around it. The curve-skeleton is both concise and expressive enough to represent an abstraction of the 3D object. Moreover, it is easier to handle than the medial axis since it avoids surface elements, while its topological structure still represents the shape efficiently.

The main difficulty in computing curve-skeletons for complex objects is to correctly interpret their shape. Objects may include complex local topology, large missing parts and noise, and this requires a robust and accurate interpretation mechanism. In this work we present an algorithm to extract the curve skeleton from 3D objects represented as boundary meshes as well as point-sets. The algorithm relies on a surface reconstruction technique that can handle both meshes and point-sets surfaces, while robustly interpreting their shape.

The key idea is to track the reconstruction of a given object using a deformable model similar to [SLS*06]. It uses competing fronts that evolve inside the object until they achieve a sufficient approximation of the shape. Since the deformable model reconstructs the original object, we compute the center curve of this model to define the center skeleton of the object itself. To do so, during the model deformation we follow the fronts' centers, generating a first approximation for the curves of the skeleton. These are filtered on-the-fly according to their geometry, their branching structure and the front competition performance, obtaining the final curve-skeleton (see Figure 1).

Furthermore, using the reconstructed model we can maintain a full correspondence between the curve-skeleton and

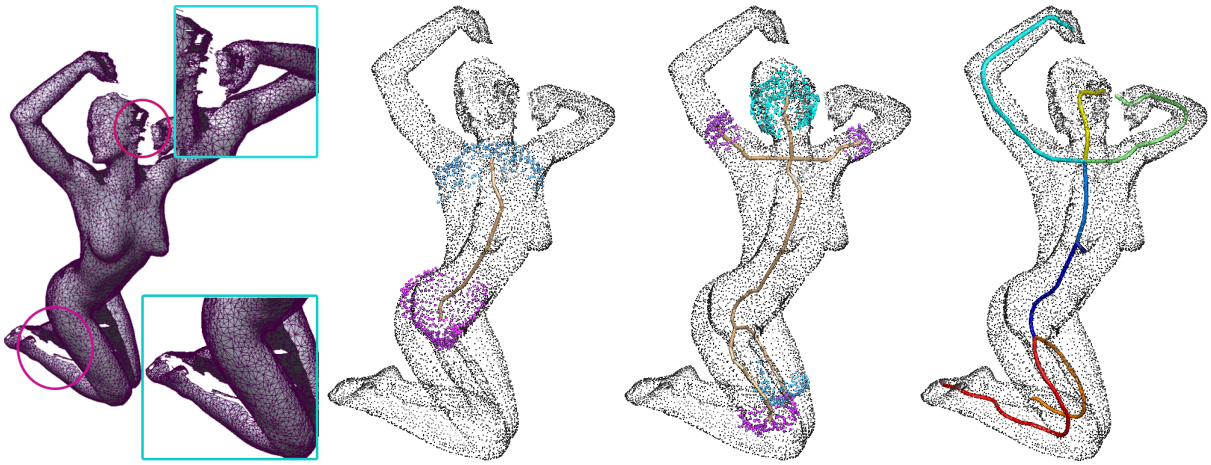

Figure 1: *On-the-fly curve-skeleton computation for a real scan of a woman model. Even in the presence large missing data (left), the deformable model accurately interprets the shape to generate a thin curve skeleton with a meaningful segmentation.*

the object boundary. For curve-skeleton of point sets, this mapping is explicit on the reconstructed mesh, while for 3D meshes it is defined by a simple closest-point projection. Such curve-skeleton computation inherits the robustness of the deformable model technique, while performing on-the-fly during the reconstruction process.

2. Related Works

Techniques to compute curve-skeletons of 3D models can be classified into three categories, according to [CSYB05b]: voxel topology, computational geometry and continuous implicit. Some of the most relevant works of each category are summarized next. We refer the reader to [CSYB05b] for a more complete survey.

Voxel topology. The computation of the curve-skeleton can be derived from computer vision techniques, such as topological thinning [BND99, GS99]. These methods iteratively remove *simple points* from the boundary of a voxel set. They differ mainly by the definition of these simple points and the priority for their removal.

Geometry. Curve-skeleton is a natural structure for 2D shapes, where it is usually computed as a subset of the medial axis [SB98]. For discrete shapes, this medial axis can be extracted from the Voronoï diagram of some points on or near the shape [DZ04, ABE]. However, for 3D shapes, the medial axis may contain surface elements. The curve skeleton can be extracted from these 2D elements using their own medial axis [AM97, ACK01], a distance field [WML*03] or a more refined geodesic field [DS06]. This set of techniques allows a definition of a curve-skeleton, but requires some delicate sampling condition to reach the correct interpretation. Our technique correctly interprets the data in various situations including both noise and large missing pieces.

Implicit. Another set of techniques compute the curve-skeleton from the ridge points of a 3D field [SZSH98, PK99, BKS01]. These techniques generally detect and track the ridge points using implicit methods such as fast marching [ZT99, HF05, CSYB05a] or active contours [GG00]. The uniform integration and complex geometry tracking used in these works typically assume that the shape thickness and complexity decreases during tracking, which may lead to shape misinterpretation in complex topology and noisy case. Moreover, this implicit process does not maintain the correspondence between the curve-skeleton and the shape. Our technique shares some similarity with these approaches. However, we do not extract the curve skeleton directly from a field, but rather interpret it in a coarse-to-fine manner using explicit evolving fronts.

Figure 2: *The deformable model with competing fronts. The fronts move in a coarse to fine manner, and may split to form sub-fronts, inducing the branching structure of our skeleton.*

3. Curve-skeleton by Reconstruction

Complex local topology and large missing parts of an object require a robust and accurate interpretation of its shape. Fol-

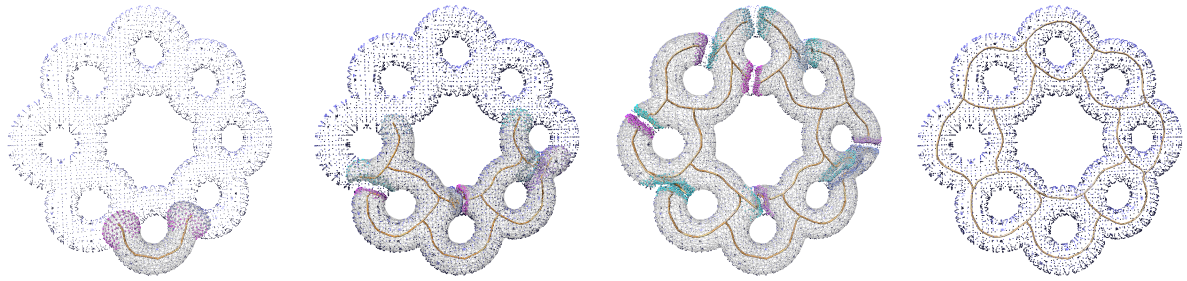

Figure 3: *Tracking the center of fronts during reconstruction defines the inner curve-skeleton of a multi-torus object. The topological events on the deformable model are mapped as loops in the curve-skeleton.*

lowing this observation, our technique for curve-skeleton extraction uses strategies similar to the reconstruction method of [SLS*06]. In particular, we utilize a deformable model to interpret the shape. We first extract from the deformation of the model a set of center curves. We filter this set during the evolution to generate a clean skeleton, while maintaining a full mapping between the model and the skeleton. This whole process is performed on-the-fly using inexpensive operations (see Algorithm 1 for pseudo-code of the whole process). In particular, the front center's tracking relies on an inexpensive marking technique, avoiding delicate geometry calculus or discrete integration.

The deformable model. Similarly to [SLS*06], our deformable model includes multiple evolving fronts that reconstruct the fine features of the shape only after the coarse ones are done. This coarse-to-fine interpretation of the shape is achieved through a competition mechanism between the fronts on an explicit, dynamic mesh representation. Each front is a connected set of mesh vertices, which moves in outward normal direction of the mesh (see Figure 2). This movement is proportional to the distance to the surface, and constrained by a Laplacian system that enforces the smoothness of the mesh. This system is weighted in order to control the tension of the mesh. A small weight provides more elasticity, allowing it to evolve fast, while a large tension slows its evolution. The competition mechanism sets the tension of a front according to its geometry and previous evolutions.

Algorithm 1 On-the-fly curve-skeleton.

```
1:  while evolving fronts do
2:      count++
3:      for all evolving front f do
4:          create skeleton node s_f
5:          s_f.arc_id ← (count, f)
6:          for all vertex v in front f do
7:              let (old_count, old_f) = v.arc_id
8:              Arcs+ = Arc[(old_count, old_f), (count, f)]
9:              v.arc_id ← (count, f)
10:     filter(Arcs) ; remap vertices to skeleton nodes
```

The vertices move while *active*, and are deactivated when they are close enough to the target shape. Deactivating vertices eventually disconnects a front. In such case, the front is split into connected sub-fronts, and each sub-front continues to evolve separately. Collisions between fronts are prevented during the evolution. At the end of the deformation, fronts collisions are interpreted as topological events and are explicitly applied as handle attachments by connecting the fronts that still collide. Even for complex topology (see Figure 3), this reconstruction strategy leads to a robust high level interpretation of the shape which we utilize for our skeleton extraction

Skeleton nodes computation. We use the same deformation technique for both point-sets and meshes. The evolving fronts algorithm is used to interpret the shape from coarse to fine, while we follow the centers of the fronts to define the centerline of the shape. Each front evolution is tracked independently. At each iteration we insert a new skeleton node for every front at the barycenter of its mesh vertices. For instance in Figure 4, the whole palm is conquered with a single front, whose skeleton node position is at the palm's center.

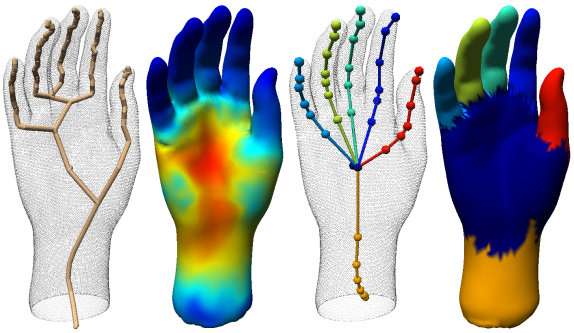

Figure 4: *Color mapping of the evolution tension parameter (left). The initial skeleton structure (left) is filtered using the evolution tension parameter (center left), simplifying the skeleton (center right) while preserving the skeleton/model correspondence and segmentation (right).*

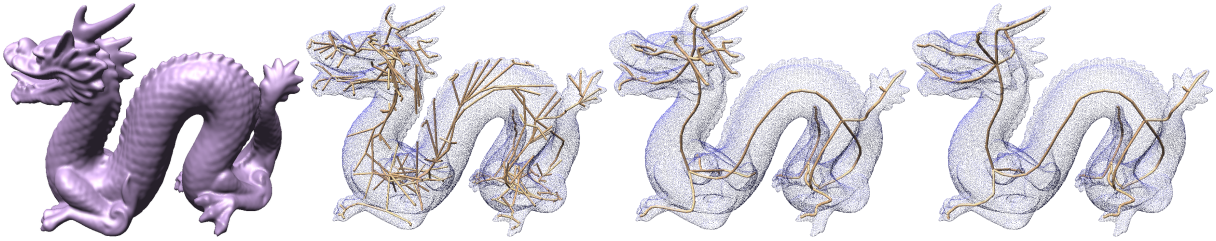

Figure 5: *Filtering of the curve-skeleton of a dragon mesh (left) from an initial curve set (center left). Filtering with loose (center right) or tight (right) threshold removes the skeleton's noise (in the dragon's front leg) and fine details (spine's bumps), while preserving the main shape elements.*

Model to skeleton correspondence. During the evolution, both skeleton nodes (positioned at the front barycenter) and fronts' mesh vertices are assigned an *arc identifier*, which is a pair composed of an iteration counter and the front identifier (see Algorithm 1: lines 5 and 9). This identifier links mesh vertices directly to curve-skeleton nodes having the same arc identifier. When a vertex is deactivated during the model deformation, it keeps its last valid arc identifier. Since the model guarantees a close reconstruction of the original shape, we simply project the model onto the shape to obtain the shape-to-skeleton correspondence (Figure 4).

Curve-skeleton structure. The skeleton nodes are connected according to the fronts' evolution that generates an initial curve set (see Figure 5). We compute these connections based on an inexpensive marking technique using arc identifiers in a union-find structure. At each iteration, the arc identifier of some front vertices is updated. Skeleton nodes corresponding to the old and new arc identifiers are connected (see Algorithm 1: lines 7-9). Straight connections without branches belong to the same curve. Note that the front identifier does not need to be coherent from one iteration to the next one, since the vertices of the mesh both define the movement and the skeleton connections. Furthermore, since evolution iterations are constrained by a tension parameter, arc links are guaranteed to lie inside the 3D shape.

Branching and topological events. During the evolution, fronts eventually split and merge. Whenever a front splits, we generate a branching node for the corresponding arc and continue to track the different sub-fronts separately. Front splits do not alter the topology of the model, and neither does the branching in our skeleton. For example on the hand of Figure 4, the splitting of the initial front at entering the fingers generates a branch for each finger and separates a sub-front for the wrist. Fronts can merge either due to topological events or to the dynamic remeshing of the deformable model. When fronts merge due to a topological event, this is explicitly detected by the deformable model. We connect the corresponding arcs to close a loop in the curve graph which ensures that the skeleton has the same homotopy as the deformable model (see Figure 3). However, for local merging of close fronts due to remeshing we do not link the corresponding branches together in order to preserve the correct topology of the skeleton. Instead, we merge the two branches into the one branch of smallest curvature.

Filtering. The final curve-skeleton is a subset of the initial curve set. Nevertheless, very similar to the medial axis, it can become very noisy and contain spurious branching especially near the object's boundary. Hence, we allow to simplify the initial curve graph connectivity by pruning and merging; Depending on a user specified parameter, we remove from the curve skeleton arcs whose length or corresponding front tension are smaller than the given threshold. Removing end arcs with no sons is a simple pruning operation, while removing intermediate arcs merge between branchings. In both cases, we re-map corresponding vertices to the new nodes using the union-find structure. Note that these operations do not alter the skeleton topology. Next, the tree (or graph due to topological events loops) geometry can be further smoothed to produce our curve-skeleton (see Figure 4) using a spline filter but keeping the branching nodes fixed. This filtering process is performed on-the-fly from the root node, which prevents the very first root branch of the skeleton to be filtered (see Figure 6). Note that both the filtered curve-skeleton and the model-skeleton mapping can be computed on-the-fly during the evolution without complex geometrical calculus.

Figure 6: *The result of our algorithm is a thin, 1D curve-skeleton even in degenerated cases such as a 3D box. When the initial model is not exactly centered (left), the skeleton quickly recovers the symmetry of the input (center). The filtering preserves the symmetry (right).*

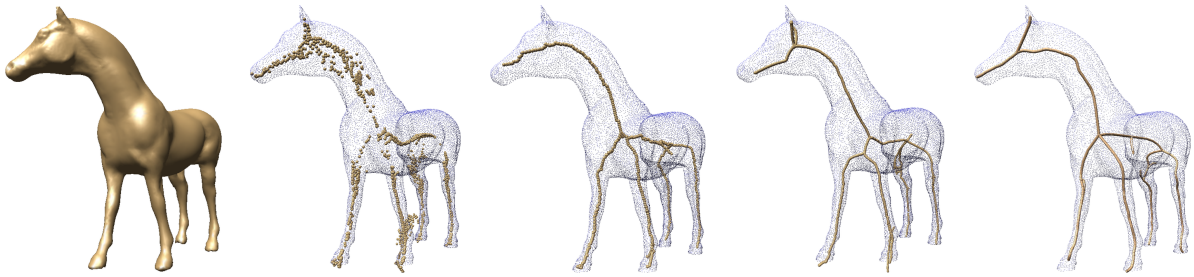

Figure 7: *Comparing our technique on a horse model. From left to right: The horse mesh. The curve-skeleton computed using the distance field of [GS99] (16 seconds). Using the topological thinning method of [PK99] (92 seconds). Using the potential field method of [CSYB05a] (16 minutes). Our result, computed in 3.1 seconds.*

4. Results

We have tested our algorithms on various inputs ranging from manifold meshes (Figures 6 and 7) to noisy points-sets with holes (Figure 1) including objects with complex topology (Figures 3 and 8) and geometry (Figure 5). Table 1 gives some timing results for the computation. We also demonstrate how using our method, the skeleton can assist in interpreting the shape and structure of an object. Figures 4 and 9 show a mapping of the skeleton curves to the boundary mesh using the skeleton mapping to the boundary. They also show how the skeleton segments the shape into meaningful parts. This method can also support a hierarchical decomposition following major and minor skeleton arcs (Figure 5). In the illustrations of this work, the curve-skeleton is drawn using cylinders for visualization purposes.

We compare our method with implicit techniques using the code provided in [CSYB05b]. In Figure 7, we use the same volumetric field for guiding our model and for the other technique. The distance field of [GS99] performs in 16 seconds, the topological thinning method of [PK99] in 92 seconds and the potential field method of [CSYB05a] in 16 minutes, while our method last 3.1 seconds.

Our method guarantees the following properties (stated in [CSYB05b] as curve-skeleton qualitative criteria); **Homotopic:** Since we close a loop in the skeleton for each topological event of the deformable model, the curve-skeleton has the same homotopy as the deformable model (Figures 3 and 8). **Thin:** As opposed to many topological thinning and distance transform methods, our skeleton is computed directly as a set of 1D curves, leading to a thin skeleton (Figure 6). **Robust component-wise differentiating:** The deformable model performs a robust shape interpretation of the data (Figures 3 and 8), which generates a meaningful decomposition of both the surface and the curve-skeleton (Figures 4 and 9). **Efficient:** Our technique performs on-the-fly in only a few seconds and is among the fastest curve-skeleton extraction algorithms (see Table 1). **Hierarchic:** As many of the distance field techniques, we use a filter to remove spurious branches. This filter can be tuned to generate a hierarchical skeleton (Figure 5).

	Deformation	Skeleton	Filtering
foot	0.7 secs.	0.4 secs.	0.1 secs.
woman	4.9 secs.	3.5 secs.	0.4 secs.
horse	1.7 secs.	1.2 secs.	0.2 secs.
multi-torus	1.0 secs.	0.6 secs.	0.3 secs.
hand	0.5 secs.	0.1 secs.	0.2 secs.
CAD	1.9 secs.	1.2 secs.	0.1 secs.
dragon	8.6 secs.	6.1 secs.	2.3 secs.

Table 1: *Computation time on a Pentium IV 1GHz / 1Gb.*

Nevertheless our method does not fully guarantee the following two criteria: direct visibility of the shape from the skeleton and centeredness. The centeredness property is observed but not guaranteed in our method since we merely use a tracking heuristic for the centers. Another limitation of our on-the-fly approach is that the skeleton root node, which corresponds to the deformable model initial positioning , can be filtered only as a post-processing.

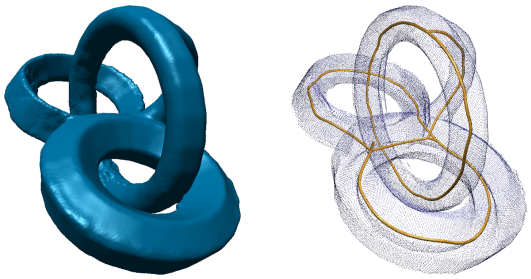

Figure 8: *Skeleton extraction for a complex CAD mesh.*

Conclusions

We propose an efficient and robust method to extract a curve-skeleton from either meshes or point-set objects. This work illustrates the similarity of shape interpretation for surface reconstruction and curve-skeleton computation. In both problems, correct interpretation and segmentation must be performed to achieve high quality results. Our method uses a

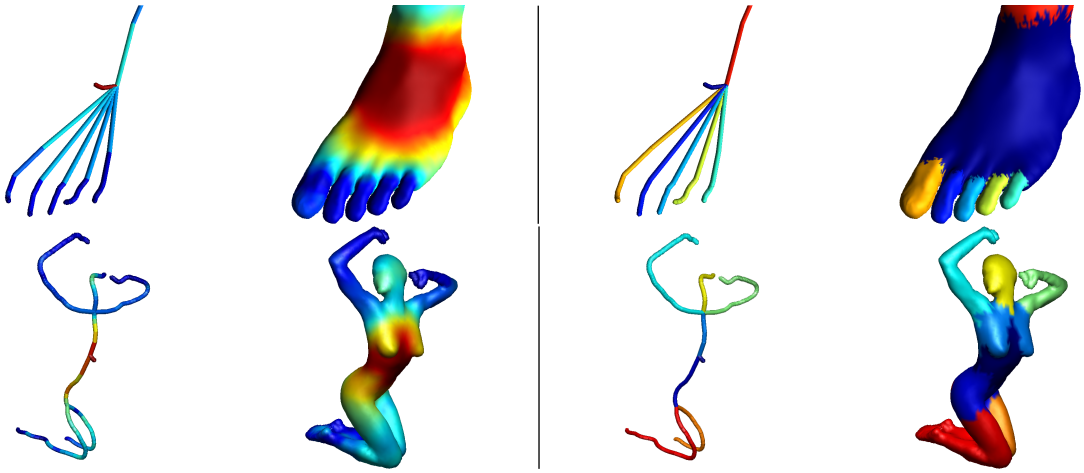

Figure 9: *The front tension parameter from the skeleton to the boundary can distinguish between coarse and fine parts of the object (color-mapped on left figures). The curve skeleton can also be used to interpret the shape: the different branches of the skeleton impose a meaningful segmentation (color-mapped on right figures).*

deformable model borrowed from reconstruction techniques to robustly interpret the shape and topology of the target object. The curve skeleton is then computed on-the-fly during the model deformation. Its robustness and speed make it attractive for many applications such as animation and matching, and we plan to utilize it to such problems in the future.

Acknowledgement This work was partly supported by the Israeli Ministry of Science, and the Brazilian Ministry of Science and Technology(MCT/CNPq02/2006).

References

[ABE] ATTALI D., BOISSONNAT J.-D., EDELSBRUNNER H.: Stability and computation of medial axes.

[ACK01] AMENTA N., CHOI S., KOLLURI R.: The Power Crust. *Comp. Geometry 19*, 2–3 (2001), 127–153.

[AM97] ATTALI D., MONTANVERT A.: Computing and simplifying 2D and 3D skeletons. *Comp. Vision and Image Underst. 67*, 3 (1997), 156–169.

[BKS01] BITTER I., KAUFMAN A., SATO M.: Penalized-distance volumetric skeleton algorithm. *TVCG 7*, 3 (2001), IEEE, 195–206.

[BND99] BORGEFORS G., NYSTROM I., DIBAJA G.: Computing skeletons in three dimensions. *Pattern Recognition 32*, 7 (1999), 1225–1236.

[CSYB05a] CORNEA N., SILVER D., YUAN X., BALASUBRAMANIAN R.: Computing hierarchical curve-skeletons of 3d objects. *The Visual Computer 21*, 11 (2005), 945–955.

[CSYB05b] CORNEA N., SILVER D., YUAN X., BALASUBRAMANIAN R.: Curve-skeleton applications. In *Visualization* (2005), IEEE, pp. 95–102.

[DS06] DEY T. K., SUN J.: Defining curve-skeletons with medial geodesic function. In *SGP* (2006), pp. 143–152.

[DZ04] DEY T. K., ZHAO W.: Approximate medial axis as a Voronoï subcomplex. *CAD 36*, 2 (2004), 195–202.

[GG00] GOLLAND P., GRIMSON W.: Fixed topology skeleton. In *Comp. Vision and Pattern Recog.* (2000), IEEE, pp. 1010–1017.

[GS99] GAGVANI N., SILVER D.: Parameter-controlled volume thinning. *Graph. Models and Image Proc. 61*, 3 (1999), 149–164.

[HF05] HASSOUNA M., FARAG A. A.: Robust centerline extraction framework using level sets. In *Comp. Vision and Pattern Recog.* (2005), IEEE, pp. 458–465.

[PK99] PALAGYI K., KUBA A.: A parallel 3D 12-subiteration thinning algorithm. *Graph. Models and Image Proc. 61*, 4 (1999), 199–221.

[SB98] SHAKED D., BRUCKSTEIN A.: Pruning medial axis. *Comp. Vision and Image Underst. 69*, 2 (1998), 156–169.

[SLS*06] SHARF A., LEWINER T., SHAMIR A., KOBBELT L., COHEN-OR D.: Competing fronts for coarse-to-fine surface reconstruction. *Computer Graphics Forum 25*, 3 (2006), 389–398.

[SZSH98] SCHIRMACHER H., ZOCKLER M., STALLING D., HEGE H.-C.: Boundary surface shrinking. In *Image and Multidim. Digital Signal Proc.* (1998), pp. 25–28.

[WML*03] WU F.-C., MA W.-C., LIOU P.-C., LAING R.-H., OUHYOUNG M.: Skeleton extraction of 3D objects with visible repulsive force. In *Pacific Graphics* (2003), pp. 409–413.

[ZT99] ZHOU Y., TOGA A.: Efficient skeletonization of volumetric objects. *TVCG 5*, 3 (1999), IEEE, 196–209.