EG UK Theory and Practice of Computer Graphics (2012)
Hamish Carr and Silvester Czanner (Editors)

Computing Curve Skeletons from Medial Surfaces of 3D Shapes

A. Telea[1] and A. C. Jalba[2]

[1]University of Groningen, the Netherlands
[2]Eindhoven University of Technology, the Netherlands

Abstract

Skeletons are powerful shape descriptors with many applications in shape processing, reconstruction and matching. In this paper we show that in 3D, curve skeletons can be extracted from surface skeletons in the same manner as surface skeletons can be computed from 3D object representations. Thus, the curve skeleton is conceptually the result of a recursion applied twice to a given 3D shape. To compute them, we propose an explicit advection of the surface skeleton in the implicitly-computed gradient of its distance-transform field. Through this process, surface skeleton points collapse into the sought curve skeleton. As a side result, we show how to reconstruct accurate and smooth surface skeletons from point-cloud representations thereof. Finally, we compare our method to existing state-of-the-art approaches.

Categories and Subject Descriptors (according to ACM CCS): I.3.5 [Computer Graphics]: Computational Geometry and Object Modeling—Curve, surface, solid, and object representations

1. Introduction

Skeletons, or medial axes, are shape descriptors used in virtual navigation, shape matching, shape reconstruction, and shape processing [SP09]. 3D shapes admit two types of skeletons. *Surface skeletons*, or S-skeletons, are 2D manifolds which contain the loci of maximally-inscribed balls within a shape [SP09, PSS*03]. *Curve skeletons*, or C-skeletons, are 1D curves which are locally centered in the shape [CSYB05]. Surface-skeleton points, together with their distance to the shape and closest-shape points, define the medial surface transform (MST), which is used for shape animation, smoothing, and matching [CC00].

While S-skeletons have a formal definition, several definitions for C-skeletons exist. Different methods use these definitions to extract C-skeletons directly from the input 3D shape. As such, the relation between surface and curve skeletons is still largely unexplored.

In this paper, we show that C-skeletons can be defined based on (and extracted from) S-skeletons analogously to the definition (and extraction) of S-skeletons from 3D shapes. For this, we extend a recent surface skeletonization method [MBC12] that extracts point-cloud S-skeletons to reconstruct manifold S-skeleton representations in a noise-resistant manner. Next, we use this representation to extract C-skeletons and compare these with results of related methods that directly extract C-skeletons from 3D shapes.

To our knowledge, the proposed approach is the first one which shows that surface and curve skeletons can be defined within a single, unitary and dimension-independent framework, rather than using separate definitions for the two. This conceptually reduces curve skeleton computation to a 'recursive' skeletonization operation applied twice on 3D shapes. From a practical viewpoint, our approach offers a robust and simple alternative for noise-resistant C-skeleton extraction from complex 3D shapes.

This paper is structured as follows. Section 2 reviews related work on curve and surface skeleton extraction. Section 3 shows how we extract manifold S-skeleton representations from (noisy) point-cloud S-skeletons. Section 4 shows how to extract C-skeletons from S-skeleton manifolds and presents results on several 3D shapes. Section 5 compares our results with two related curve skeleton extractors and discusses our method. Section 6 concludes the paper.

DOI: 10.2312/LocalChapterEvents/TPCG/TPCG12/099-106

2. Related Work

Given a shape $\Omega \subset \mathbb{R}^3$ with boundary $\partial\Omega$, we first define its distance transform $DT_{\partial\Omega} : \Omega \to \mathbb{R}^+$

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

The S-skeleton of Ω is next defined as

$$S(\Omega) = \{\mathbf{x} \in \Omega \,|\, \exists \mathbf{y}, \mathbf{z} \in \partial\Omega, \mathbf{y} \neq \mathbf{z}, \\ \|\mathbf{x} - \mathbf{y}\| = \|\mathbf{x} - \mathbf{z}\| = DT_{\partial\Omega}(\mathbf{x})\}, \tag{2}$$

where $\|\cdot\|$ is the Euclidean distance metric in $\mathbb{R}^3$, and $\mathbf{y}$ and $\mathbf{z}$ are the contact points with $\partial\Omega$ of the maximally-inscribed ball in Ω centered at $\mathbf{x}$, also called *feature points* [ST04] or *spoke vectors* [SWS09]. $S(\Omega)$ is a set of manifolds which meet along a set of Y-intersection curves [Dam06] or medial scaffold [LK07, CLK09].

Surface skeletons: $S(\Omega)$ is computed by voxel or mesh-based methods. Voxel-based methods include thinning, distance-field, and general-field methods. Thinning removes voxels of $\partial\Omega$ while preserving connectivity [PK99, Pud98]. Distance-field methods find $S(\Omega)$ along singularities of $DT_{\partial\Omega}$ [RT02, TvW02, WDK01, SBTZ02, HR08] and can be efficiently done on GPUs [ST04, SFM05, vDvdWT06, CTMT10]. General-field methods use fields with less singularities than distance transforms [AC97, CSYB05, HF09], so are more robust for noisy shapes. Stolpner *et al.* find skeleton voxels where the gradient of the shape's distance transform is multi-valued [SWS09, SWS11]. *Mesh-based methods* often use Voronoi diagrams to compute polygonal skeletons [DZ03]. Amenta *et al.* compute the Power Crust, an approximate S-skeleton, by a carefully-chosen subset of Voronoi points [ACK01]. Miklos *et al.* approximate shapes by a union of balls (UoB) and use UoB medial properties [GMPW09] to simplify S-skeletons [MGP10]. Mesh-based methods compute S-skeletons very precisely, can handle non-uniformly sampled surfaces, and use much less memory – typically $O(N^2)$ as compared to $O(N^3)$ needed for a N^3 voxel volume [Sud06].

Curve skeletons: C-skeletons have widely different definitions [CSM07, CTK00]. Among recent advances, Au *et al.* compute C-skeletons by collapsing the input shape via Laplacian smoothing [ATC*08]. ROSA computes C-skeletons as centers of point-cloud projections on a cut plane found by optimizing for circularity [TZCO09]. Hassouna *et al.* [HF09] compute C-skeletons as the extrema of a cost function that captures centrality with respect to the input surface, and find these extrema using variational methods in a voxel setting.

Dey and Sun present the medial geodesic function (MGF), which defines the C-skeleton as the locus of S-skeleton points having two or more different shortest-geodesics between their feature points [DS06, PH02]. Reniers *et al.* [RvWT08] extend the MGF to compute both surface and curve skeletons using geodesic lengths and surface areas between geodesics, respectively. Unlike these methods, our approach does not rely on a volumetric representation of the input 3D object. As such, computing the C-skeleton by selecting those points of the S-skeleton which admit two or more, different shortest geodesics, would result in a very sparse (and disconnected) C-skeleton. Therefore, our approach does not rely on this criterion for selecting C-skeleton points and avoids the sparsity problem by computing the C-skeleton through collapsing the (connected) S-skeleton mesh.

3. Surface Skeleton Extraction

Recently, Ma *et al.* proposed arguably the fastest existing method for extracting S-skeletons from meshed shapes [MBC12]. Given a mesh representation $M = (\{(\mathbf{p}_i, \mathbf{n}_i)\}, \{t_j\})$ of an input surface $\partial\Omega$ consisting of points $\mathbf{p}_i$ with (unit) normals $\mathbf{n}_i$ and triangles t_j, for each point $\mathbf{p} \in M$, a (large) ball $B(\mathbf{s}, r_L) = -r_L\ \mathbf{n} + \mathbf{p}$ is created, with center $\mathbf{s}$ and tangent at $\mathbf{p}$. Then, the algorithm iteratively shrinks B until it becomes maximally inscribed, at which instance its center $\mathbf{s}'$ yields the final skeleton point. By definition, $\mathbf{f}_1^{\mathbf{s}'} \equiv \mathbf{p}$ is the first feature point of S-skeleton point $\mathbf{s}'$, whereas the second contact point of the maximal ball yields the second feature point $\mathbf{f}_2^{\mathbf{s}'}$ of $\mathbf{s}'$. However, as the authors of [MBC12] point out, this method produces a *point cloud* S-skeleton, S_C, which is of limited use in applications.

We create a mesh representation M_S from S_C in a three-step process, as follows.

Importance computation: Given S_C, we compute the importance $\rho : S_C \to \mathbb{R}^+$ as

$$\rho(\mathbf{s} \in S_C) = \min_{\gamma = (\mathbf{f}_1^{\mathbf{s}} \rightsquigarrow \mathbf{f}_2^{\mathbf{s}}) \subset \partial\Omega} \|\gamma\| \tag{3}$$

i.e. the length of the shortest geodesic path γ on the mesh M between the two feature points $\mathbf{f}_1^{\mathbf{s}}$ and $\mathbf{f}_2^{\mathbf{s}}$ of $\mathbf{s}$. For this, we compute the geodesic distance $DT_M(\mathbf{f}_1^{\mathbf{s}} \to \mathbf{f}_2^{\mathbf{s}})$ on M between $\mathbf{f}_1^{\mathbf{s}}$ and $\mathbf{f}_2^{\mathbf{s}}$ using the Fast Marching Method, and next trace γ in $-\nabla DT_M(\mathbf{f}_1^{\mathbf{s}} \to \mathbf{f}_2^{\mathbf{s}})$ from $\mathbf{f}_2^{\mathbf{s}}$ to $\mathbf{f}_1^{\mathbf{s}}$ [PC05]. This is essentially the S-skeleton importance metric of Reniers *et al.* [RvWT08], implemented in our case on mesh, rather than voxel, surfaces. Other methods can be used for the same purpose, *e.g.* [SSK*05, VS09].

S-skeleton reconstruction: Given ρ, we now iterate over all triangles $t_i = \{\mathbf{p}_j^i\}, 1 \leq j \leq 3$ of the mesh M and select, for each triangle vertex $\mathbf{p}_j^i$, the corresponding skeleton point $\mathbf{s}(\mathbf{p}_j^i) \in S$ having the maximal importance, *i.e.*

$$\mathbf{s}(\mathbf{p}_j^i) = \text{argmax}_{\mathbf{s} \in S \,|\, \mathbf{f}_1^{\mathbf{s}} = \mathbf{s}(\mathbf{p}_j^i) \vee \mathbf{f}_2^{\mathbf{s}} = \mathbf{s}(\mathbf{p}_j^i)}\, \rho(\mathbf{s}) \tag{4}$$

Next, we create a S-skeleton triangle $t_i^S = \{\mathbf{s}(\mathbf{p}_j^i)\}, 1 \leq j \leq 3$ for t_i. The S-skeleton mesh M_S is then simply the collection of these triangles, $\{t_i^S\}$. Let us observe that M_S is homotopic to M. In particular, M_S preserves the manifold properties of M. This is due to the fact that the vectors $\mathbf{s}(\mathbf{p}_j^i) - \mathbf{p}_j^i$ do not intersect with each other for any triangle t_i, as these vectors are parallel to the gradient vector of $DT_{\partial\Omega}$. Further, these vectors span a vector field which is divergence-free everywhere outside $S(\Omega)$ [SP09]. Hence, Eqn. 4 effectively 'maps' M onto M_S as if M were advected in $\nabla DT_{\partial\Omega}$ until it reached the S-skeleton.

S-skeleton smoothing: As noted in [MBC12], if the input mesh M is noisy, S_C will be a noisy skeleton point-cloud, due to the well-known fact that small surface perturbations create many spurious skeleton sheets [PSS*03, Dam06]. In turn, this means that our skeleton mesh M_S can contain many spike-like triangles. We easily regularize M_S by performing $2 \ldots 5$ (constrained) Laplacian smoothing iterations on all its points whose importance ρ exceeds a (small) user-specified threshold τ. This effectively removes spikes created by low-importance skeleton points, by pulling these points towards the high-importance skeleton points to which they are connected in M_S. This process is different from the skeleton regularization of Reniers *et al.* [RvWT08]: While Reniers *et al.* *remove* low-importance skeleton points to obtain a clean skeleton, we *move* these points onto the planes of triangles formed by high-importance points. While point removal would imply a complicated re-meshing process, our solution is straightforward.

Figure 1 (a1-i1) show several skeleton meshes M_S constructed by our method. The corresponding skeleton clouds S_C are shown in Fig. 1 (a2-i2), colored by importance ρ via a blue-to-red colormap. The reconstructed meshes capture accurately fine-scale skeleton details even for complex shapes.

4. Curve Skeleton Extraction

4.1. Definition

Given a surface skeleton $S(\Omega)$ of a 3D shape, defined as in Eqn. 2, we define the corresponding curve skeleton by using an analogous approach to Eqn. 2, as follows.

Let us first examine the continuous case. Denote by ∂S the *boundary* of the S-skeleton, *i.e.*, the set of 1D curves that form the so-called external borders of the skeletal manifolds [LK07, CLK09]. Then, by analogy to Eqn. 1, we first define the distance transform $DT_{\partial S} : S \to \mathbb{R}^+$ of ∂S as

$$DT_{\partial S}(\mathbf{x} \in S) = \min_{\mathbf{y} \in \partial S} \|\mathbf{x} - \mathbf{y}\|_S \tag{5}$$

where $\|\cdot\|_S$ is the geodesic distance metric on S.

Given Eqn. 5, we now define the curve skeleton $C(\Omega)$ as

$$\begin{aligned} C(\Omega) = \{\mathbf{x} \in S \,|\, \exists \mathbf{y}, \mathbf{z} \in \partial S, \mathbf{y} \neq \mathbf{z}, \\ \|\mathbf{x} - \mathbf{y}\|_S = \|\mathbf{x} - \mathbf{z}\|_S = DT_{\partial S}(\mathbf{x})\}. \end{aligned} \tag{6}$$

In words, C is the locus of points on the S-skeleton at equal geodesic distance on S from at least two points on ∂S.

Note the similarity of the C-skeleton and S-skeleton definitions, *i.e.* Eqns. 1 and 5, and Eqns. 2 and 6 respectively. The curve skeleton is nothing but the 'skeleton of the surface skeleton', where we replace the Euclidean distance in Ω (used for the surface skeleton) by the geodesic distance in $S(\Omega)$ (used for the curve skeleton). In this sense, the C-skeleton is conceptually the result of a recursive skeletonization operation applied twice on a given 3D shape.

4.2. Computation

Directly solving Eqns. 5 and 6 that define the C-skeleton is, however, much more involved than solving Eqns. 1 and 2 that define the S-skeleton. First, we would need to accurately locate the boundary ∂S of the S-skeleton, using the mesh representation M_S computed as outlined in Sec. 3. This is already challenging, given the very complex structure of M_S (see Fig. 1 a1-i1). Secondly, we would need to compute geodesic paths on M_S, which is possible, given the manifold structure of M_S, but relatively expensive.

We take here a different approach. Given the identical structure of the C-skeleton and S-skeleton definitions, we infer that the C-skeleton $C(\Omega)$ is the locus of singularities of $DT_{\partial S}$, by analogy with the identical well-known observation that holds for the S-skeleton. Hence, we can compute $C(\Omega)$ by advecting all points of $S(\Omega)$ in $\nabla DT_{\partial S}$, until these points reach the aforementioned singularities. The problem is now reduced to computing $\nabla DT_{\partial S}$.

For the above, we proceed as follows. Consider a S-skeleton like the one of a 3D parallelepiped shown in Fig. 2. Consider a S-skeleton point $\mathbf{s}$ and its two feature points $\mathbf{f}_1^{\mathbf{s}}$ and $\mathbf{f}_2^{\mathbf{s}}$. Each such feature-point-pair is connected by a shortest geodesic γ_s on the input surface $\partial\Omega$, as described in Sec. 3. Consider now the tangent vectors to γ_s at its feature endpoints, oriented as shown in Fig. 2, *i.e.* $\mathbf{t}_1^{\mathbf{s}}, \mathbf{t}_2^{\mathbf{s}}$. Following the observations of Reniers *et al.* [RvWT08], the sum vectors $(\mathbf{t}_1^{\mathbf{s}} + \mathbf{t}_2^{\mathbf{s}})/2$ are tangent to $S(\Omega)$ and oriented in the direction of $\nabla DT_{\partial S}(\mathbf{s})$. The above hold for any S-skeleton point, see *e.g.* also point $\mathbf{q}$ in Fig. 2. Since we have already computed the shortest-geodesics for all S-skeleton points as part of the regularized S-skeleton mesh construction (Sec. 3), we obtain our desired gradient field virtually for free, *i.e.*, without having to explicitly compute $\nabla DT_{\partial S}$.

Given the above implicitly-computed gradient field $\nabla DT_{\partial S}$, we now advect all S-skeleton points $\mathbf{s} \in M_S$ iteratively in the normalized gradient field by

$$\mathbf{s}^{i+1} = P_{T(\mathbf{s}^i)}\left(\mathbf{s}^i + \frac{\nabla DT_{\partial S}(\tilde{\mathbf{s}}^i)}{\|\nabla DT_{\partial S}(\tilde{\mathbf{s}}^i)\|}\delta\right). \tag{7}$$

Here, $\tilde{\mathbf{s}}$ is the closest skeleton point in the original skeleton cloud S_C to the currently advected point $\mathbf{s}^i$ at the i^{th} iteration, found by searching for the closest nearest-neighbor of $\mathbf{s}^i$ in

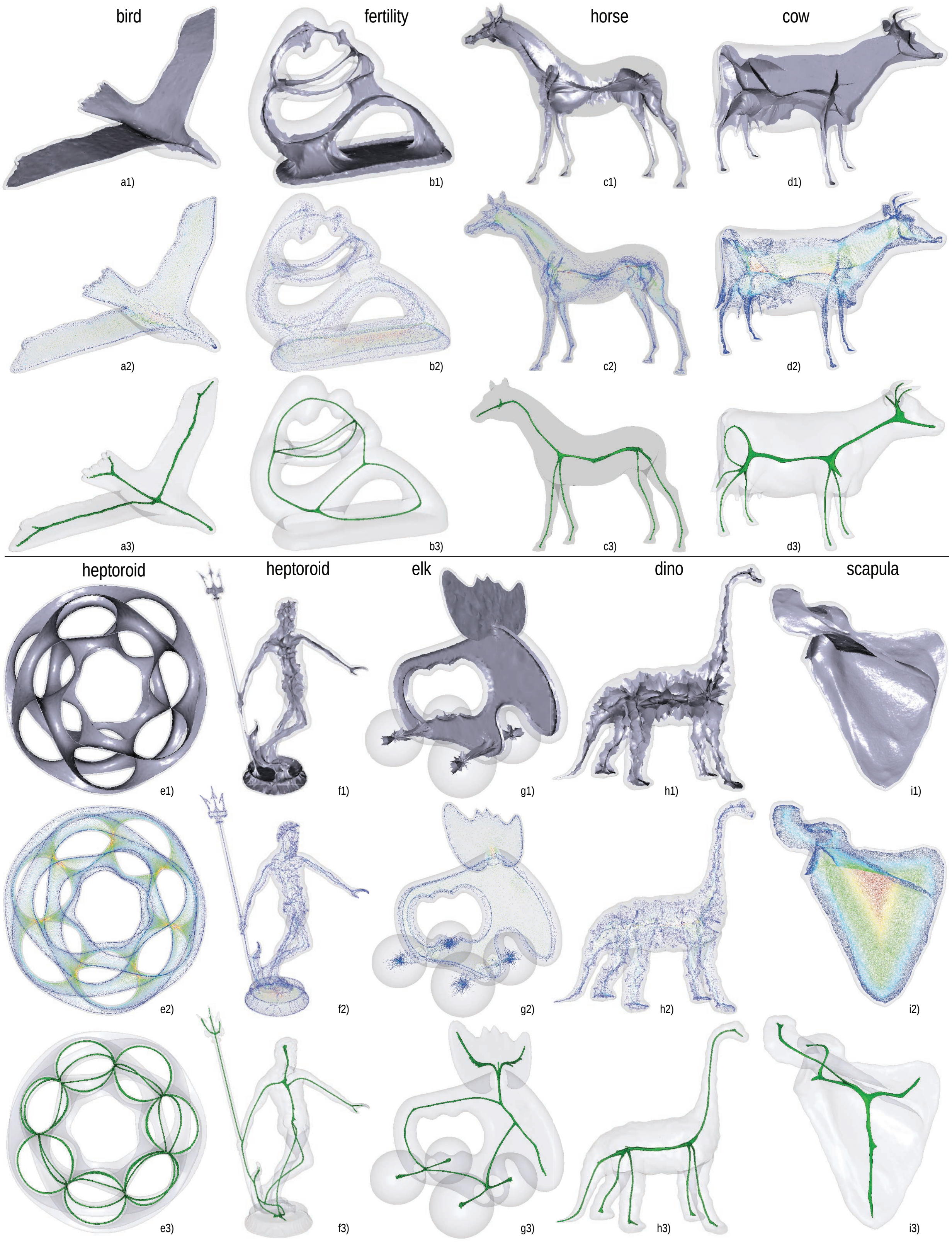

Figure 1: *Examples: Medial surfaces (a1-i1); medial point clouds colored by importance (a2-i2); curve skeletons (a3-i3).*

S_C. This effectively samples our implicit gradient field at the locations of the advected skeleton points during the advection. Gradient normalization produces a field whose slope is everywhere 1, *i.e.* the same as the gradient of a distance transform. δ is set to half of the average edge-size of the triangle fan in M_S around $\mathbf{s}^i$. This ensures that advection steps are proportional to the local skeleton-mesh density. The operator $P_{T(\mathbf{s}^i)}$ projects the advected point on the triangle fan $T(\mathbf{s}^i)$ in the skeleton mesh around $\mathbf{s}^i$ to constrain advection on the S-skeleton surface.

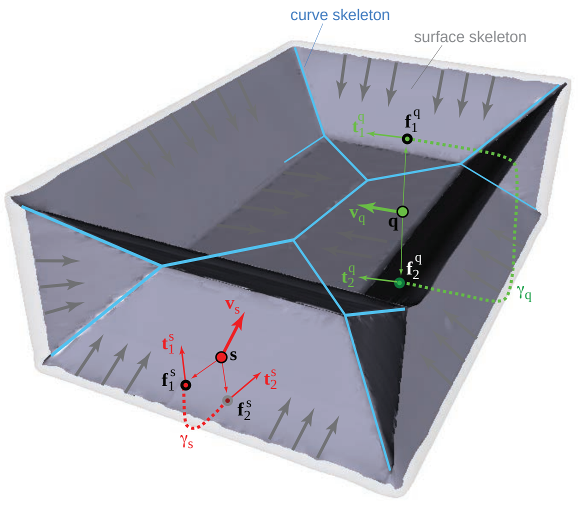

Figure 2: *Advection principle for curve skeleton extraction*

As explained in [DS06, RvWT08], certain S-skeleton points admit more than one shortest-geodesic. The above-mentioned authors use this property to define the locus of the C-skeleton. In our model, such points are located precisely on the singularities of $\nabla DT_{\partial S}$, *i.e.* are points where this gradient abruptly switches directions. In other words, our C-skeleton definition and computation proposal produces precisely the same C-skeletons as [DS06, RvWT08].

Several implementation details are important for stability and convergence speed, as follows. We advect S-skeleton points in decreasing order of their importance ρ. This ensures an "upwind strategy" such that points can be updated in place, in decreasing distance from the S-skeleton boundary (since ρ monotonically increases on S from ∂S inwards [RvWT08]). In practical terms, this reduces numerical problems such as skeleton-mesh self intersections. To further reduce such problems, which may create badly shaped triangles as the skeleton mesh collapses towards its C-skeleton, we perform one Laplacian smoothing iteration of the advected mesh after each advection iteration.

After advecting each skeleton point, we evaluate the area of its triangle fan $T(\mathbf{s}^i)$, and block the point for any further advection if this area falls below a very small value $\varepsilon = 10^{-5}$. This stopping criterion determines whether S-skeleton points have reached the location of the C-skeleton. A similar approach is used in [ATC*08]. Blocking advection of such points has two desirable effects: First, convergence is sped up as points reaching the C-skeleton do not require further update. More importantly, without such blocking, end-points of C-skeleton branches would be advected along these branches inwards, which would needlessly shrink the C-skeleton.

Figure 3 shows several iteration steps of the advection of the S-skeleton of a hand model. The top row shows the corresponding S-skeleton meshes. We see how the S-skeleton shrinks inwards from its (implicit) boundary with equal speed. During this process, the mesh remains of high quality due to the constrained advection to the skeleton surface and Laplacian smoothing. Also, we see how points that reach the (implicit) location of the C-skeleton stay blocked, see *e.g.* points along the finger centerlines. This is due to blocking the advection of small-area triangles. The implicit C-skeleton structure starts becoming visible already from the tenth iteration, see Fig. 3 2b, in the middle of the palm. After 35 iterations, the entire S-skeleton has collapsed to its C-skeleton (Fig. 3 5b).

In contrast to other methods *e.g.* [ATC*08], we do not compute the C-skeleton as an explicit 1D curve, or polyline. Our C-skeleton is essentially a mesh having the same topology as the S-skeleton but shrunk to a dense 1D geometric structure. To convert such C-skeleton meshes into 1D curve skeletons, one could remove all collapsed mesh faces through a mesh decimation technique, similar to that used by Au *et al.* [ATC*08]. Finally, we render the C-skeleton by drawing small fixed-size balls centered at the advected points (Fig. 3 a5).

5. Results and Comparison

We implemented the S-skeleton reconstruction (Sec. 3) and C-skeleton extraction (Sec. 4.2) in C++ using the ANN library [MA11] for nearest-neighbor point searches (Eqn. 7). The S-skeleton reconstruction is $O(N)$ for a skeleton cloud of N points. On a 2.8 GHz PC, this takes under 3 seconds for skeleton clouds of up to 500K points. C-skeleton extraction takes 30...40 iterations to converge to a thin structure for all meshes presented in this paper. C-skeleton extraction is $O(kN\ logN)$ for k collapse iterations on a N point S-skeleton points. The entire method requires $O(N)$ memory for meshes of N points, *i.e.*, the costs of storing the S-skeleton mesh and 2 feature points per skeleton point. Table 1 shows timings for the models in Figs. 1 and 3.

Figure 4 shows several curve skeletons extracted by two methods related to our approach: Reniers *et al.* [RvWT08] extract C-skeletons from voxel models using the MGF-based geodesic criterion [DS06]. We use geodesics differently, *i.e.* to determine the collapse direction of S-skeleton points,

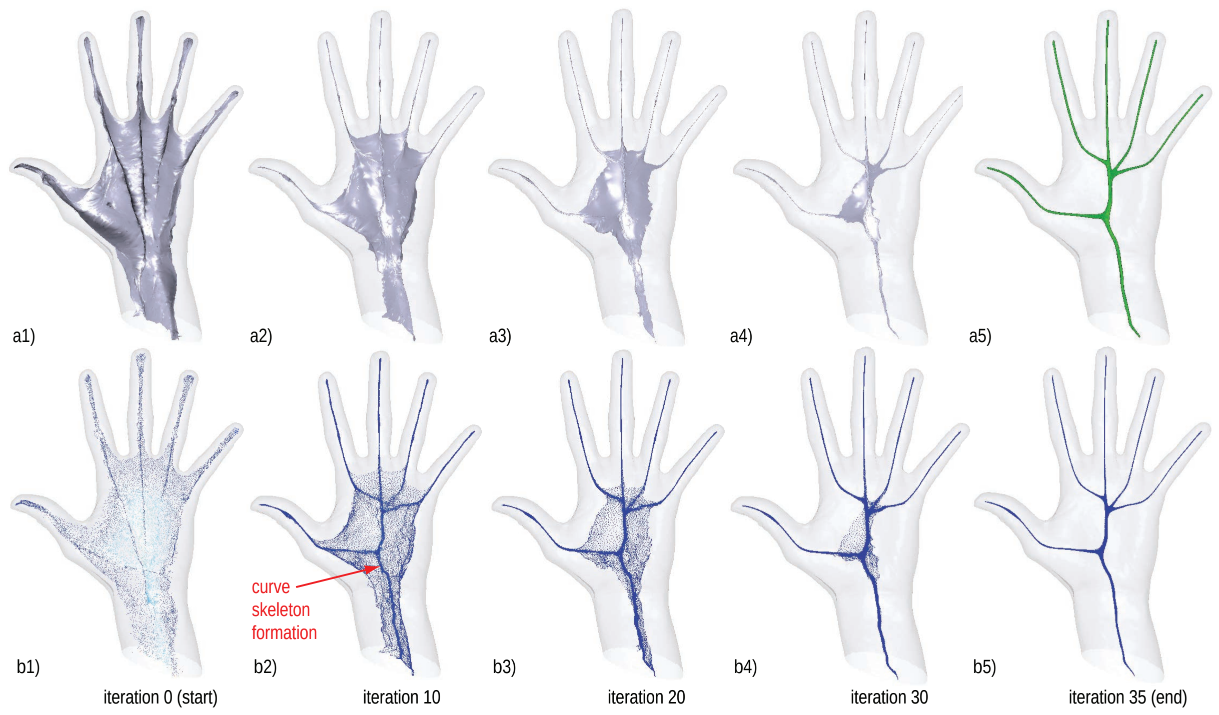

Figure 3: *Advection steps for curve skeleton extraction. (a1-a4) Surface skeleton mesh. (b1-b5) Medial point clouds. (a5) Curve skeleton rendering.*

Table 1: *Timings for C-skeleton computation, Section 4.2.*

Model	Points	Triangles	Extraction time (sec.)
Bird	11718	23432	2.1
Fertility	24994	50000	8.7
Horse	193934	387864	246
Cow	185882	371456	232
Heptoroid	79056	158196	41
Neptune	28052	56112	8.6
Elk	35062	70124	9.1
Dino	23255	46600	8.2
Scapula	116930	233856	134
Hand	197245	393984	253

rather than to select C-skeleton points from the S-skeleton. Au *et al.* extract C-skeletons from mesh models by collapsing the *mesh* along its normals. In contrast, we collapse the *S-skeleton* along its distance-transform gradient, rather than the input mesh. Our area-based collapse stopping criterion (Sec. 4.2) is however similar to the one of Au *et al.*

Comparing Fig. 4 with the corresponding results of our method in Fig. 1, we notice several differences. Even though our surface point count is smaller than, or at most equal to, the surface voxel count of Reniers *et al.*, our C-skeleton has more branches, see *e.g.* the cow udder and horns (Fig. 4 c *vs* Fig. 1 d3). This is expected: Both we and Reniers *et al.* simplify the S-skeleton prior to C-skeleton computation, to eliminate spurious branches. However, while Reniers *et al.* do this by *thresholding* ρ, which cuts off peripheral skeleton parts, we just smooth such areas (Sec. 3). This keeps more of the peripheral skeleton details.

Compared to Au *et al.*, our skeletons do not have artificial straight-line branches and sharp bends (see Fig. 4 e-l, green markers). Au *et al.* added these in a so-called "surgery" step to connect disjoint skeleton parts. Our C-skeletons are inherently smoother since our entire extraction process is continuous (advection in the continuous $\nabla DT_{\partial S}$ and Laplacian smoothing of S). However, this built-in continuity also makes our C-skeletons slightly thicker close to junctions. Thinner C-skeletons can be obtained by decreasing the minimal collapsed area constraint ε (Sec. 4.2), at the expense of more iterations. In contrast, the method of Au *et al.* has several built-in constraints. While these ensure C-skeleton centeredness and thinness, smoothness is suboptimal.

Performance-wise, our method is slightly faster, on the average, than Au *et al.* and Reniers *et al.*, see Tab. 1 *vs* Fig. 4. Although these are positive findings, we stress that we do not consider our speed results to be an important value point for our method: Our main novelty, and added-value, is that we showed that C-skeletons can be computed directly, and only, from S-skeletons, using an identical skeleton definition for both skeleton types. If desired, significant speed-ups of the advection process can be obtained by parallel implementation of the advection algorithm *e.g.* using CUDA.

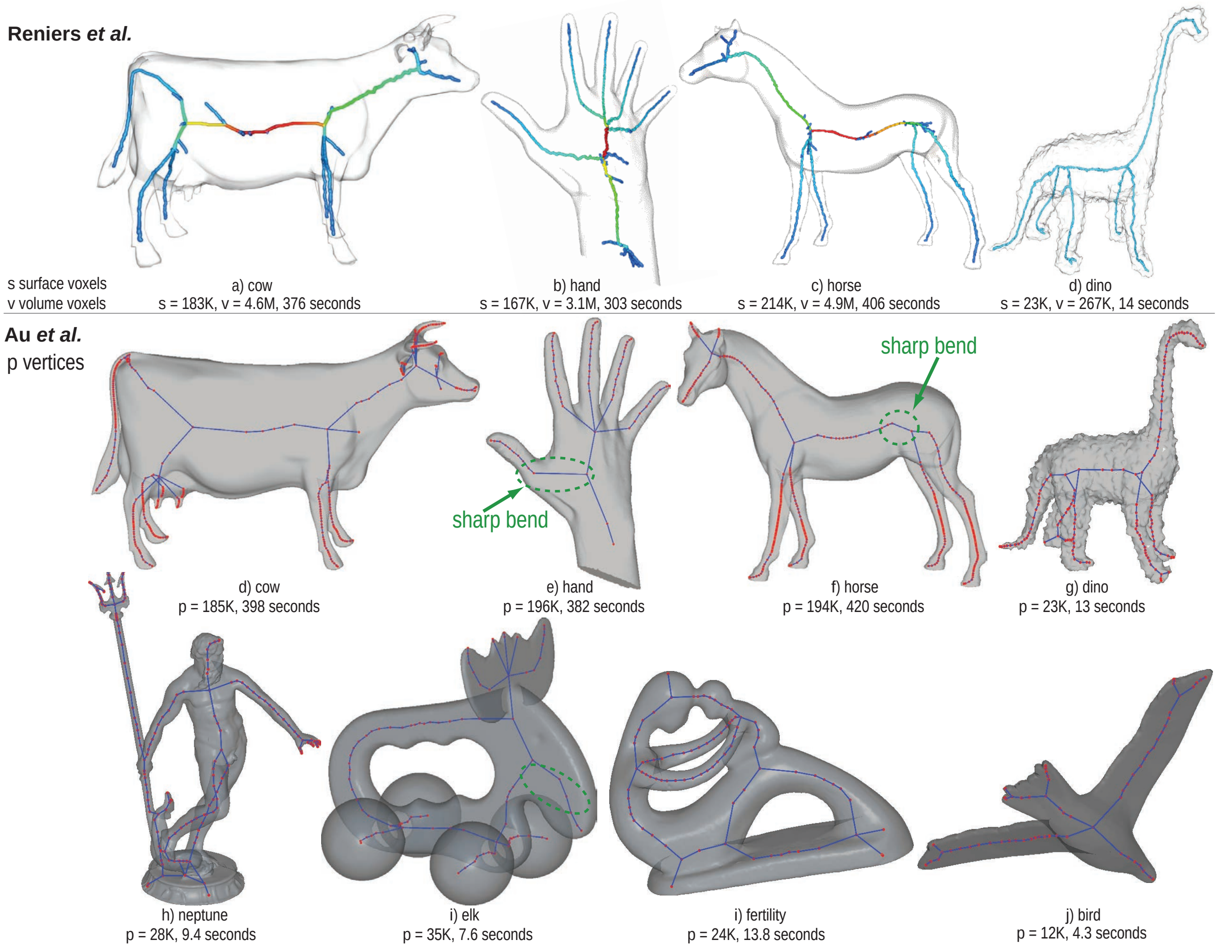

Figure 4: *Curve skeletons extracted by related methods. Compare results with Fig. 1 and timings in Tab. 1.*

6. Conclusions

In this paper, we have introduced a new definition for curve skeletons for 3D shapes. We generalized the well-known definition of medial surfaces (in 3D) or medial axes (in 2D) to define curve skeletons as the loci of points on the medial surface situated at maximal *geodesic* distance from at least two medial surface boundary points. We showed that our definition delivers detailed and centered curve skeletons which resemble results produced by other curve-skeletonization methods. Next, we have presented a simple method to compute curve skeletons based on an explicit advection of the surface skeleton in the implicitly-computed gradient of its distance-transform field. As a side result, we have showed how to reconstruct accurate and smooth surface skeletons from point-cloud representations thereof.

Our results open several interesting follow-up directions. Given our unified surface and curve-skeleton definitions, new theoretical insights on quantitative and qualitative relations of the two skeletons can be researched. We plan to investigate these relations to use our surface and curve skeletons for shape segmentation, feature extraction, and shape compression applications.

On the practical side, a current shortcoming of our method, which we plan to eliminate in a future work, is that the C-skeleton is represented by a shrunk mesh instead of a polyline. Secondly, we plan to implement fast GPU-based numerical methods to perform the advection process which, combined with the already-existing fast GPU surface-skeleton extraction [MBC12], should lead to a competitive curve skeletonization pipeline.

References

[AC97] AHUJA N., CHUANG J.: Shape representation using a generalized potential field model. *IEEE TPAMI 19*, 2 (1997), 169–176. 2

[ACK01] AMENTA N., CHOI S., KOLLURI R.: The power crust. In *Proc. SMA* (2001), ACM, pp. 65–73. 2

[ATC*08] AU O. K. C., TAI C., CHU H., COHEN-OR D., LEE

T.: Skeleton extraction by mesh contraction. In *Proc. ACM SIGGRAPH* (2008), pp. 441–449. 2, 5

[CC00] Costa L., Cesar R.: *Shape analysis and classification*. CRC Press, 2000. 1

[CLK09] Chang M., Leymarie F., Kimia B.: Surface reconstruction from point clouds by transforming the medial scaffold. *CVIU*, 113 (2009), 1130–1146. 2, 3

[CSM07] Cornea N., Silver D., Min P.: Curve-skeleton properties, applications, and algorithms. *IEEE TVCG 13*, 3 (2007), 87–95. 2

[CSYB05] Cornea N., Silver D., Yuan X., Balasubramanian R.: Computing hierarchical curve-skeletons of 3D objects. *Visual Computer 21*, 11 (2005), 945–955. 1, 2

[CTK00] Chuang J., Tsai C., Ko M.: Skeletonization of three-dimensional object using generalized potential field. *IEEE TPAMI 22*, 11 (2000), 1241–1251. 2

[CTMT10] Cao T., Tang K., Mohamed A., Tan T.: Parallel banding algorithm to compute exact distance transform with the GPU. In *Proc. SIGGRAPH I3D Symp.* (2010), pp. 134–141. 2

[Dam06] Damon J.: Global medial structure of regions in $\mathbf{R}^3$. *Geometry and Topology 10* (2006), 2385–2429. 2, 3

[DS06] Dey T., Sun J.: Defining and computing curve skeletons with medial geodesic functions. In *Proc. SGP* (2006), IEEE, pp. 143–152. 2, 5

[DZ03] Dey T., Zhao W.: Approximating the medial axis from the Voronoi diagram with a convergence guarantee. *Algorithmica 38* (2003), 179–200. 2

[GMPW09] Giesen J., Miklos B., Pauly M., Wormser C.: The scale axis transform. In *Proc. Annual Symp. Comp. Geom.* (2009), pp. 106–115. 2

[HF09] Hassouna M., Farag A.: Variational curve skeletons using gradient vector flow. *IEEE TPAMI 31*, 12 (2009), 2257–2274. 2

[HR08] Hesselink W., Roerdink J.: Euclidean skeletons of digiral image and volume data in linear time by the integer medial axis transform. *IEEE TPAMI 30*, 12 (2008), 2204–2217. 2

[LK07] Leymarie F., Kimia B.: The medial scaffold of 3d unorganized point clouds. *IEEE TVCG 29*, 2 (2007), 313–330. 2, 3

[MA11] Mount D., Arya S.: Approximate nearest neighbor search software. `www.cs.umd.edu/~mount/ANN`. 5

[MBC12] Ma J., Bae S., Choi S.: 3D medial axis point approximation using nearest neighbors and the normal field. *Vis. Comput. 28*, 1 (2012), 7–19. 1, 2, 3, 7

[MGP10] Miklos B., Giesen J., Pauly M.: Discrete scale axis representations for 3D geometry. In *Proc. ACM SIGGRAPH* (2010), pp. 394–493. 2

[PC05] Peyre G., Cohen L.: Geodesic computations for fast and accurate surface remeshing and parameterization. In *Progress in Nonlinear Differential Equations and Their Applications* (2005), vol. 63, Springer LNCS, pp. 151–171. `www.ceremade.dauphine.fr/~peyre`. 2

[PH02] Prohaska S., Hege H. C.: Fast visualization of planelike structures in voxel data. In *Proc. IEEE Visualization* (2002), p. 29Ð36. 2

[PK99] Palagyi K., Kuba A.: Directional 3D thinning using 8 subiterations. In *Proc. DGCI* (1999), vol. 1568, Springer LNCS, pp. 325–336. 2

[PSS*03] Pizer S., Siddiqi K., Szekely G., Damon J., Zucker S.: Multiscale medial loci and their properties. *IJCV 55*, 2-3 (2003), 155–179. 1, 3

[Pud98] Pudney C.: Distance-ordered homotopic thinning: A skeletonization algorithm for 3D digital images. *CVIU 72*, 3 (1998), 404–413. 2

[RT02] Rumpf M., Telea A.: A continuous skeletonization method based on level sets. In *Proc. VisSym* (2002), pp. 151–158. 2

[RvWT08] Reniers D., van Wijk J. J., Telea A.: Computing multiscale skeletons of genus 0 objects using a global importance measure. *IEEE TVCG 14*, 2 (2008), 355–368. 2, 3, 5

[SBTZ02] Siddiqi K., Bouix S., Tannenbaum A., Zucker S.: Hamilton-Jacobi skeletons. *IJCV 48*, 3 (2002), 215–231. 2

[SFM05] Sud A., Foskey M., Manocha D.: Homotopy-preserving medial axis simplification. In *Proc. SPM* (2005), pp. 103–110. 2

[SP09] Siddiqi K., Pizer S.: *Medial Representations: Mathematics, Algorithms and Applications*. Springer, 2009. 1, 3

[SSK*05] Surazhsky V., Surazshky T., Kirsanov D., Gortler S., Hoppe H.: Fast exact and approximate geodesics on meshes. In *Proc. ACM SIGGRAPH* (2005), pp. 130–138. 2

[ST04] Strzodka R., Telea A.: Generalized distance transforms and skeletons in graphics hardware. In *Proc. VisSym* (2004), pp. 221–230. 2

[Sud06] Sud A.: *Efficient computation of discrete Voronoi diagram and homotopy-preserving simplified medial axis of a 3D polyhedron*. PhD thesis, UNC Chapel Hill, 2006. 2

[SWS09] Stolpner S., Whitesides S., Siddiqi K.: Sampled medial loci and boundary differential geometry. In *Proc. IEEE 3DIM* (2009), pp. 87–95. 2

[SWS11] Stolpner S., Whitesides S., Siddiqi K.: Sampled medial loci for 3D shape representation. *CVIU 115*, 5 (2011), 695–706. 2

[TvW02] Telea A., van Wijk J. J.: An augmented fast marching method for computing skeletons and centerlines. In *Proc. VisSym* (2002), pp. 251–259. 2

[TZCO09] Tagliasacchi A., Zhang H., Cohen-Or D.: Curve skeleton extraction from incomplete point cloud. In *Proc. SIGGRAPH* (2009), pp. 541–550. 2

[vDvdWT06] van Dortmont M., van de Wetering H., Telea A.: Skeletonization and distance transforms of 3D volumes using graphics hardware. In *Proc. DGCI* (2006), Springer LNCS, pp. 617–629. 2

[VS09] Verma V., Snoeyink J.: Reducing the memory required to find a geodesic shortest path on a large mesh. In *Proc. ACM GIS* (2009), pp. 227–235. 2

[WDK01] Wan M., Dachille F., Kaufman A.: Distance-field based skeletons for virtual navigation. In *Proc. IEEE Visualization* (2001), pp. 239–246. 2