An Unified Multiscale Framework for Planar, Surface, and Curve Skeletonization

Jalba, Andrei C.; Sobiecki, Andre; Telea, Alexandru C.

Published in:
Ieee transactions on pattern analysis and machine intelligence

DOI:
10.1109/TPAMI.2015.2414420

Publication date:
2016

An Unified Multiscale Framework for Planar, Surface, and Curve Skeletonization

Andrei C. Jalba, Andre Sobiecki, and Alexandru C. Telea

Abstract—Computing skeletons of 2D shapes, and medial surface and curve skeletons of 3D shapes, is a challenging task. In particular, there is no unified framework that detects all types of skeletons using a single model, and also produces a multiscale representation which allows to progressively simplify, or regularize, all skeleton types. In this paper, we present such a framework. We model skeleton detection and regularization by a conservative mass transport process from a shape's boundary to its surface skeleton, next to its curve skeleton, and finally to the shape center. The resulting density field can be thresholded to obtain a multiscale representation of progressively simplified surface, or curve, skeletons. We detail a numerical implementation of our framework which is demonstrably stable and has high computational efficiency. We demonstrate our framework on several complex 2D and 3D shapes.

Index Terms—Medial axes, skeleton regularization, physically-based shape processing

1 Introduction

Skeletons, or medial axes, are shape descriptors used in virtual navigation, shape matching, shape reconstruction, and shape processing [60]. 3D shapes admit two types of skeletons. *Surface skeletons* are 2D manifolds which contain the loci of maximally-inscribed balls in a shape [50], [60]. *Curve skeletons* are 1D curves which are locally centered in the shape [16]. Surface-skeleton points, with their distance to the shape and closest-shape points, define the medial surface transform (MST), used for animation, smoothing, and matching [4], [7], [20].

Many methods exist for computing 2D skeletons [20], [47], [69], 3D surface skeletons [27], [53], [59], [64], and 3D curve skeletons [6], [18], [26], [68]. Although recent methods demonstrate high accuracy, insensitivity to noise, and computational efficiency, several challenges remain open. We focus here on two modeling challenges, as follows. First, 2D skeletons, 3D surface skeletons, and 3D curve skeletons are typically extracted, and next simplified, using *different* methods and metrics. This makes the comparison and the formal reasoning about the properties of the extracted skeletons difficult. Second, few (if any) methods offer a continuous multiscale representation that addresses *all* skeleton types, i.e., a model which encodes both the geometric *importance* of any skeleton point (useful for simplifying, or regularizing, noisy skeletons) and the *type* of skeleton point (non-skeleton, surface skeleton, or curve skeleton).

In this paper, we present a framework for 2D and 3D curve-and-surface skeletonization that addresses the above two goals. We model both the skeleton detection and its importance using an advection principle that collapses mass from a shape boundary to its skeleton and next to the skeleton center (in 2D); and from the boundary to the surface skeleton, next to the curve skeleton, and finally to the latter's center (in 3D). This allows us to detect all types of mentioned skeletons, and also to regularize them, e.g., to remove detail branches, via a single model and a simple thresholding operation. We propose a single algorithm that unifies skeleton detection and regularization in 2D and 3D, and also establishes a formal connection between surface and curve skeletons. Our method is simple to implement, computationally efficient, and easy to use. We show that our results are very similar to the ones produced by several existing 2D and 3D skeletonization methods on a set of complex shapes.

The structure of this paper is as follows. Section 2 reviews related work. Section 3 presents our skeletonization method. Section 4 details our method's implementation. Section 5 compares our results with one 2D, six 3D surface, and 11 curve skeletonization methods. Section 6 discusses our results. Section 7 concludes the paper.

2 Related Work

Given a shape $\Omega \subset \mathbb{R}^n$, $n \in \{2, 3\}$ with boundary $\partial\Omega$, we first define its euclidean distance transform $DT_{\partial\Omega} : \mathbb{R}^n \rightarrow \mathbb{R}^+$

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

The skeleton, or medial axis, of Ω is next defined as

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

where $\mathbf{f}_1$ and $\mathbf{f}_2$ are the contact points with $\partial\Omega$ of the maximally inscribed balls in Ω centered at $\mathbf{x}$ [23], [53]. The points $\mathbf{f}_1$ and $\mathbf{f}_2$ are called *feature transform* (FT) points [65]. The vectors $\mathbf{f} - \mathbf{x}$ are called *spoke vectors* [63]. For $n = 2$, S_Ω is a set of curves which meet at the so-called skeleton junction

- *A.C. Jalba is with the Institute for Mathematics and Computing Science, Eindhoven University of Technology, Eindhoven, The Netherlands. E-mail: a.c.jalba@tue.nl.*
- *A. Sobiecki and A.C. Telea are with the Johann Bernoulli Institute for Mathematics and Computer Science, University of Groningen, Groningen, The Netherlands. E-mail: {a.sobiecki, a.c.telea}@rug.nl.*

Manuscript received 3 Apr. 2014; revised 6 Feb. 2015; accepted 10 Mar. 2015. Date of publication 23 Mar. 2015; date of current version 9 Dec. 2015. Recommended for acceptance by C. Sminchisescu.
For information on obtaining reprints of this article, please send e-mail to: reprints@ieee.org, and reference the Digital Object Identifier below.
Digital Object Identifier no. 10.1109/TPAMI.2015.2414420

points [20]. For $n = 3$, S_Ω is a set of manifolds with boundaries which meet along a set of so called Y-intersection curves [14], [17], [36].

In contrast to 2D and surface skeletons (Eqn. (2)), 3D curve skeletons CS_Ω admit many definitions [16], implemented by a wide variety of methods (discussed further below). As such, a formal relationship between S_Ω and CS_Ω is still not unanimously accepted. For instance, although it is commonly accepted that CS_Ω should be centered within S_Ω, only few skeletonization methods use and/or enforce this property [30], [53].

Skeletons can be computed by various methods, as follows.

Thinning. Thinning removes $\partial\Omega$ voxels (or pixels in 2D) while preserving connectivity [8], [48], [52]. Although simple and fast, thinning can be sensitive to euclidean transformations.

Field methods. These methods find S_Ω along singularities of $DT_{\partial\Omega}$ or related fields [22], [27], [33], [37], [55], [69], [72] and can be efficiently done on GPUs [13], [65], [66]. General-field methods use fields smoother (with fewer singularities) than distance transforms [1], [4], [16], [26], and thus are more robust for noisy shapes. Siddiqi et al. find the skeleton as the non-zero divergence locus of $\nabla DT_{\partial\Omega}$ [59]. However, $\nabla \cdot (\nabla DT_{\partial\Omega})$, with $\nabla\cdot$ the divergence operator, can be non-zero also at non-skeletal points. Torsello and Hancock (TH) correct this for a more accurate 2D skeleton detection by a momentum conservation principle $\nabla \cdot (\rho \nabla DT_{\partial\Omega}) = 0$, where ρ is the mass density on the evolving boundary $\partial\Omega$ [5]. Rossi and Torsello extend this idea to compute 3D surface skeletons [54]. However, this method does not compute curve skeletons and does not model the curve-surface skeleton relationship.

Mesh-based methods. Field methods volumetrically sample Ω, which can be expensive memory-wise. Mesh-based methods use a surface sampling of $\partial\Omega$, which allows processing higher-resolution shapes. Mesh methods include Voronoi diagrams to compute polygonal skeletons [19]. Amenta et al. compute the Power Crust, an approximation of a surface and its medial axis by a subset of Voronoi points [2]. Other methods use edge collapses [38], starting from a mesh segmentation [32]. Surface skeletons can be extracted from oriented point clouds [29], [41] or polygon meshes [36], [45] by searching for maximally inscribed balls tangent at at least two shape points. Curve skeletons can be extracted from point clouds as centers of cloud projections on a cut plane which optimizes for circularity [68]. *Contraction* techniques are a separate subclass of mesh methods. Like field techniques, they evolve $\partial\Omega$ under various types of normal flows, effectively collapsing it onto the surface-or-curve skeleton. Methods using a (constrained) Laplacian contraction by mean curvature flow deliver high-quality curve skeletons [6], [12], [15], or even 'meso skeletons' mixes of surface and curve skeletons [67]. A different approach is taken by Jalba and Telea who contract the surface skeleton to compute its curve skeleton counterpart [30]. A recent review of contraction methods is given in [62].

Multiscale skeletons (MS). Clean skeletons are extracted from noisy shapes by thresholding *importance measures* $\rho : \Omega \rightarrow \mathbb{R}^+$. This prunes skeletal branches caused by small details [17], [58]. We distinguish between local and global measures [44], [53]. *Local measures* cannot separate locally-identical, yet globally-different, contexts (see e.g., [53], Fig. 1). Thresholding local measures can disconnect skeletons. Reconnection needs extra work [42], [50], [59], [66], and makes pruning less intuitive [58]. Local measures include the angle between the feature points and distance-to-boundary [2], [21], [66], divergence-based [11], [59], first-order moments [55], and points where $\nabla DT_{\partial\Omega}$ is multi-valued [63], [64]. Leymarie and Kimia topologically simplify point-cloud skeletons to capture Y-intersection curves and skeleton sheet boundaries in medial scaffolds [36]. A good survey of such methods is given in [60].

Global measures. monotonically increase from the skeleton boundary ∂S_Ω inwards. Thresholding them yields connected skeletons which capture skeleton details at a user-given scale. Miklos et al. approximate shapes by unions of balls (UoB) and use UoB medial properties [24] to simplify skeletons [44]. Dey and Sun introduce the medial geodesic function (MGF), equal to the shortest-geodesic length between feature points [18], [51]. Reniers et al. [53] extend the MGF for surface and curve skeletons using geodesic lengths and surface areas between geodesics, respectively, inspired by the so-called collapse metric used to extract multiscale 2D skeletons [20], [47], [69]. A fast GPU implementation of this extended MGF is given in [29].

The MGF and its 2D collapse metric counterpart have an intuitive geometric meaning: They assign to a skeleton point $\mathbf{p}$ the amount of shape boundary that corresponds, or 'collapses' to, $\mathbf{p}$ by some kind of boundary-to-skeleton mass transport. Skeleton points $\mathbf{p}$ with low metric values correspond to small-scale shape details or noise; points $\mathbf{p}$ with large metric values correspond to large-scale shape details. This allows an easy simplification of the skeleton: Thresholding by a value τ eliminates all skeleton points which encode less than τ boundary length or area units. If the collapse metric monotonically increases from the skeleton boundary to its center, thresholding delivers a set of connected and nested skeleton approximations, also called a multiscale skeleton [18], [20], [53], [69].

3 Proposed Framework

3.1 Preliminaries

Following the above, we aim to create a single model that

1) unifies the representation and detection of 2D skeletons, 3D surface skeletons, and 3D curve skeletons;
2) computes a monotonic, global importance metric for 2D and 3D skeleton regularization and simplification;

Conceptually, we aim to capture the desirable properties of the 2D and 3D boundary collapse metric [20], [47], [53], [69] in a single model, and also connect contraction-based and distance-field based methods in a single framework. Practically, we aim at a single, easy to implement and use, and computationally efficient method that extracts and regularizes all skeleton types.

To achieve this, we first introduce our unified skeleton definition: Given a shape $\Omega \in \mathbb{R}^{n\in\{2,3\}}$, we aim to compute an importance function $\lambda : \Omega \rightarrow \mathbb{R}^+$ so that the threshold sets $\lambda_\tau = \{\mathbf{x} \in \Omega | \lambda(\mathbf{x}) \geq \tau\}$ capture all existing skeleton types *and*

all their simplifications. Specifically, we want λ_0 to be the full input shape Ω; λ_ϵ (for a small $\epsilon > 0$) to be the full (unsimplified) surface skeleton S_Ω, which implies that $\lambda(\mathbf{x}) = 0$, $\forall \mathbf{x} \notin S_\Omega$. As τ increases, we want λ_τ to be progressively simplified surface skeletons, and as τ increases even further, progressively simplified curve skeletons. In the limit, when $\tau = \max_{\mathbf{x}\in\Omega}\lambda(\mathbf{x})$, we want λ_τ to be a single point (for genus 0 objects), which we next call the shape center C_Ω. This process can be seen as a 'recursive' skeletonization which computes the surface skeleton from the input shape, the curve skeleton from the surface skeleton, and the shape center from the curve skeleton. All skeletons λ_τ should satisfy the well-known desirable properties—centeredness, rotational invariance, homotopy to the input shape Ω, noise robustness, one-pixel (in 2D) and one-voxel (in 3D) thickness, inclusion of the curve skeleton in the surface skeleton, and computational efficiency [16], [61], [62].

3.2 Physically-Based Skeletonization Model

For a shape $\Omega \in \mathbb{R}^{n\in\{2,3\}}$, we model our unified skeletonization as a contraction of Ω on whose boundary mass is distributed with unit density. Contraction is described by three fields: $\phi(\mathbf{x}, t)$, $\rho(\mathbf{x}, t)$, and $\mathbf{u}(\mathbf{x}, t)$, with $\mathbf{x} \in \Omega$, and with $t \in \mathbb{R}^+$ being the time parameter, as follows. Similar to phase-field models [10], the field $\phi \to [-1, 1]$ is 1 inside Ω and -1 outside, so that the boundary of the contracting shape is implicitly given by $\Gamma_t = \{\mathbf{x} \in \Omega \,|\, \phi(\mathbf{x}, t) = 0\}$. For now, we assume that ϕ varies abruptly and monotonically over $[-1, 1]$ in a small vicinity around Γ_t. The field $\rho \to \mathbb{R}^+$ gives the mass density of Γ_t. Finally, $\mathbf{u} \to \mathbb{R}^n$ gives the contraction direction of Γ_t.

Our contraction is described by a system of three PDEs:

$$\frac{\partial \rho}{\partial t} + \nabla \cdot (\rho \mathbf{u}) = 0, \tag{3}$$

$$\frac{\partial \phi}{\partial t} + \mathbf{u} \cdot \nabla \phi = 0, \text{with } \Gamma_t \cong \Gamma_0, \tag{4}$$

$$\mathbf{u} = \frac{\nabla \phi}{\|\nabla \phi\|}. \tag{5}$$

Equation (3) imposes mass conservation on the shrinking boundary. Equation (4) models boundary contraction with the topological homeomorphic constraint $\Gamma_t \cong \Gamma_0$. This ensures that the computed skeletons are homotopic to the input shape $\partial\Omega = \Gamma_0$. Equation (5) imposes inwards contraction of our shape, with unit speed in normal direction to Γ_t.

Eliminating $\mathbf{u}$ from Eqns. (3)-(5), we obtain

$$\frac{\partial \rho}{\partial t} = -\nabla \rho \cdot \frac{\nabla \phi}{\|\nabla \phi\|} - \rho \nabla \cdot \frac{\nabla \phi}{\|\nabla \phi\|} = -\nabla \rho \cdot \frac{\nabla \phi}{\|\nabla \phi\|} - \rho\kappa \tag{6}$$

$$\frac{\partial \phi}{\partial t} + \|\nabla \phi\| = 0, \text{with } \Gamma_t \cong \Gamma_0, \tag{7}$$

where κ is the (mean) curvature of Γ_t.

Equations (6)-(7) are supplemented by the initial conditions

$$\phi(\mathbf{x}, t = 0) = \begin{cases} 1, & \text{if } \mathbf{x} \in \Omega \\ -1, & \text{if } \mathbf{x} \notin \Omega \end{cases} \tag{8}$$

$$\rho(\mathbf{x}, t = 0) = \begin{cases} 1, & \text{if } \mathbf{x} \in \partial\Omega \\ 0, & \text{if } \mathbf{x} \notin \partial\Omega. \end{cases} \tag{9}$$

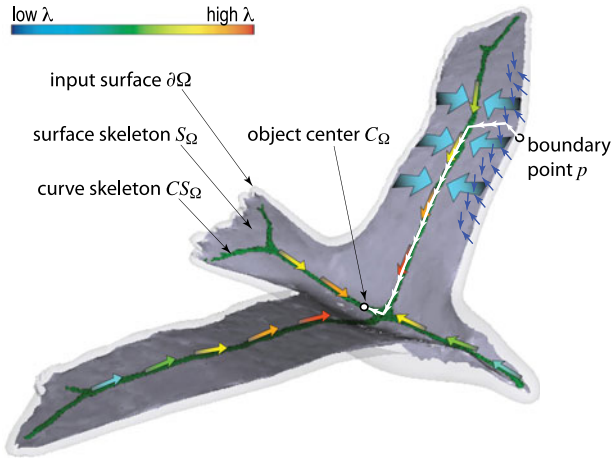

Fig. 1. Density advection model from the surface $\partial\Omega$ of a genus-0 input shape to its surface skeleton S_Ω, curve skeleton CS_Ω, and object center C_Ω. Colors depict the importance λ of different spatial regions.

Let us define the time-of-arrival function $T : \Omega \to \mathbb{R}^+$ so that

$$\phi(\mathbf{x}, t) = T(\mathbf{x}) - t. \tag{10}$$

Hence, $\Gamma_t = \{\mathbf{x} \in \Omega \,|\, T(x) = t\}$, *i.e.*, $T(\mathbf{x})$ is the time after which Γ_t passes through $\mathbf{x}$. Using Eqs. (7) and (10), we obtain $\|\nabla T\| = -\phi(\cdot, t) = 1$, the well-known Eikonal equation for arrival time T. The euclidean distance transform $DT_{\partial\Omega}$ is the weak solution of this equation under euclidean norm [57]. Hence, Eq. (7) without the constraint $\Gamma_t \cong \Gamma_0$ is the PDE generating continuous multi-scale (flat) morphological erosions. Other norms are also possible [43], leading to various distance transforms.

We finally define the skeleton importance λ as the maximum density that has reached a certain location $\mathbf{x} \in \Omega$, i.e.,

$$\lambda(\mathbf{x}) = \max_{t>0} \rho(\mathbf{x}, t). \tag{11}$$

Intuitively, our model describes a conservative advection process where mass, uniformly spread on $\partial\Omega$, flows on shortest paths from $\partial\Omega$ to its surface skeleton S_Ω; then, along S_Ω on shortest paths to the curve skeleton CS_Ω; and finally along CS_Ω on shortest paths to the shape center C_Ω (Fig. 1). Once all mass has reached C_Ω, we compute the (simplified) surface and curve skeletons by thresholding λ at increasing values.

4 Solving the System

To compute the importance λ, we solve the contraction model in Section 3.2 by discretizing Ω on a uniform cubic-cell (pixel or voxel) grid embedded in $\mathbb{R}^2$ and $\mathbb{R}^3$ respectively, as follows.

4.1 Topologically-Constrained Boundary Evolution by Density-Ordered Thinning

As stated in Section 3.2, Eq. (7) must be solved with the constraint that Γ_t and Γ_0 are homeomorphic, for all t. Even without this constraint, it is well-known that the evolution of ϕ from Eq. (7) develops discontinuities of the ϕ derivatives (shocks) within finite time [43], [57]. The skeleton S_Ω precisely coincides with the locations of these shocks [59].

Since Eq. (7) can be written as an Eikonal evolution, or boundary-value problem (Section 3.2), one way to interpret the contraction is as thresholding $DT_{\partial\Omega}$ inside Ω at increasingly higher values, producing multi-scale morphological erosions of Ω. Additionally, the topological constraint $\Gamma_t \cong \Gamma_0$ should also be satisfied by each level set of $DT_{\partial\Omega}$ corresponding to Γ_t. For achieving this, we could consider using topologically-constrained level sets [25]. The problem with this approach is that it first performs an *un-constrained* step to update level-set values, following the motion equation (Eq. (7)). Then, at points $\mathbf{x}$ where the topological constrained is violated, the so-called level-set function ψ is next 'fixed' so that the points $\mathbf{x}$ lie on the corresponding side of the boundary dictated by the constraint. This fix *artificially* alters the ψ values, which creates spurious and unwanted discontinuities in ψ, ultimately leading to a not sharply-defined (in terms of our desired sharp transition of the level-set function in $[-1, 1]$) and/or non-smooth evolution of Γ_t. In turn, this will drastically affect the quality of the extracted skeletons, as we verified in practice. As such, we chose not to use topologically-constrained level-sets for our context.

To handle all above issues, we use topology-preserving morphological thinning to define and steer the evolution (contraction) of Γ_t. Our thinning process is ordered both by $DT_{\partial\Omega}$ and by the density field ρ: As long as Γ_t is far from the skeleton S_Ω, ordering by $DT_{\partial\Omega}$ ensures a smooth Γ_t while solving Eq. (7). Additionally, since thinning relies on a binary field, Γ_t is maintained sharp during its evolution.

We next explain why the thinning order is also given by the density field ρ, which is crucial when the evolving Γ_t reaches the (yet unknown) skeleton locations. Recall that such locations correspond to shocks of Eq. (7). Hence, ordering by $DT_{\partial\Omega}$ (which is just a viscosity solution of Eq. (7)) becomes meaningless. As sketched in Fig. 1, we want the importance λ, and thus also the density ρ which determines this importance, to monotonically increase from $\partial\Omega$ to S_Ω, next from ∂S_Ω to CS_Ω, and finally from ∂CS_Ω (curve-skeleton endpoints) to C_Ω. Since Γ_t shrinks in normal direction (Eq. (7)), this is equivalent to transporting density on shortest paths from $\partial\Omega$ to S_Ω to CS_Ω and next to C_Ω. Fig. 1 shows such a path (in white) on which the mass of a point $\mathbf{p} \in \partial\Omega$ should flow during its advection to C_Ω. Consider now the set of all such paths from all points on $\partial\Omega$ to C_Ω. For a shape Ω of genus 0, following a reasoning similar to [53], these paths will form a tree having as leaves all (discrete) points of $\partial\Omega$ and C_Ω as root. The computation of ρ by means of our contracting Γ_t is analogous to traversing this tree from its leaves to the root. To ensure a correct density update, we thus need that, at any junction-point where several subtrees meet, all these trees to have been fully traversed and their roots' densities to be thus correctly updated. This is why our thinning visits points in Γ_t in increasing ρ order.

Fig. 2 illustrates our thinning for a 3D shape. When using density-ordered thinning, Γ_t (drawn red) is kept smooth during collapsing. In contrast, if not using density ordering, the collapsing Γ_t will quickly develop irregularities (Figs. 2e, 2f, 2g, and 2h, insets). In turn, these will create irregularities in the signal ρ which will ultimately lead to jagged skeletons λ_τ after simplification.

4.2 Algorithm

Summarizing the observations from Section 4.1, our contraction algorithm should:

R_1: provide a sharp definition of the evolving boundary Γ_t;
R_2: allow interleaved iterative solves of Eqs. (6) and (7);
R_3: ensure a smooth evolution of Γ_t, steered by $DT_{\partial\Omega}$ and ρ;
R_4: allow efficient computation.

Most existing thinning algorithms do not provide a representation of Γ_t which satisfies all requirements $R_1..R_3$ above. For example, the divergence-driven thinning algorithm in [59] uses a sorted heap to ensure the correct processing order, thus fails to provide an explicit Γ_t representation. In contrast, we use an explicit representation of Γ_t, modeled as a narrow-band of points (that is, pixels in 2D and voxels in 3D, respectively). Density is transported, according to Eq. (6), only within this narrow-band, which is computationally efficient (R_4).

Let us note that some thinning algorithms combine the detection and removal of a so-called *topologically-simple* point in a single pass. The thinning result may then depend on the point processing order, as discussed in [28]. In contrast, we use an approach similar to [28], where we first find all simple points (detection phase), and eliminate these next (removal phase).

Algorithm 1. Skeletonization algorithm.

Skeletonize(Shape Ω, Field λ)
Data: Ω: discretized input shape
Result: λ: importance field
$DT_{\partial\Omega}(\cdot) \leftarrow$ 3D euclidean distance transform of $\partial\Omega$;
$\rho(\cdot) \leftarrow 0$; $\lambda(\cdot) \leftarrow 0$; $M(\cdot) \leftarrow 0$; $Q_1 \leftarrow \varnothing$;
foreach $\mathbf{x} \in \Omega$ **do**
  if $DT_{\partial\Omega}(\mathbf{x}) > 0$ **then** $M(\mathbf{x}) \leftarrow 2$; // *interior points*
foreach $\mathbf{x} \in \Omega$ **do**
  if $DT_{\partial\Omega}(\mathbf{x}) > 0 \wedge DT_{\partial\Omega}(\mathbf{x}) < 2 \wedge simple(\mathbf{x}, M)$ **then**
    $\rho(\mathbf{x}) \leftarrow 1$; $M(\mathbf{x}) \leftarrow 1$; $Q_1 \leftarrow Q_1 \cup \{\mathbf{x}\}$; // *boundary points*
$d \leftarrow 0$;
repeat
  $d \leftarrow d + \Delta d$; $Q_2 \leftarrow \varnothing$;
  foreach $\mathbf{x} \in Q_1$ **do**
    foreach $\mathbf{y} \in \mathcal{N}(\mathbf{x}) \wedge M(\mathbf{y}) = 2$ **do**
      $Q_2 \leftarrow Q_2 \cup \{\mathbf{y}\}$; $M(\mathbf{y}) \leftarrow 1$; // *new boundary point*
  Sort Q_1 in increasing ρ order;
  $C \leftarrow \varnothing$;
  foreach $\mathbf{x} \in Q_1$ **do** // *process Q_1 in increasing ρ order*
    if $DT_{\partial\Omega}(\mathbf{x}) < d \wedge simple(\mathbf{x}, M)$ **then** $C \leftarrow C \cup \{\mathbf{x}\}$;
    else $Q_2 \leftarrow Q_2 \cup \{\mathbf{x}\}$; $M(\mathbf{x}) \leftarrow 1$;
  $B \leftarrow \varnothing$;
  foreach $\mathbf{x} \in C$ **do**
    if $simple(\mathbf{x}, M)$ **then**
      $M(\mathbf{x}) \leftarrow 0$; $B \leftarrow B \cup \{\mathbf{x}\}$;
      $\lambda(\mathbf{x}) \leftarrow max(\lambda(\mathbf{x}), \rho(\mathbf{x}))$;
    else $Q_2 \leftarrow Q_2 \cup \{\mathbf{x}\}$; $M(\mathbf{x}) \leftarrow 1$;
  Transport ρ from $\mathbf{x} \in B$ to interior points, using Eq. 6;
  foreach $\mathbf{x} \in B$ **do** $\rho(\mathbf{x}) \leftarrow 0$;
  $swap(Q_1, Q_2)$;
until $B = \varnothing$;

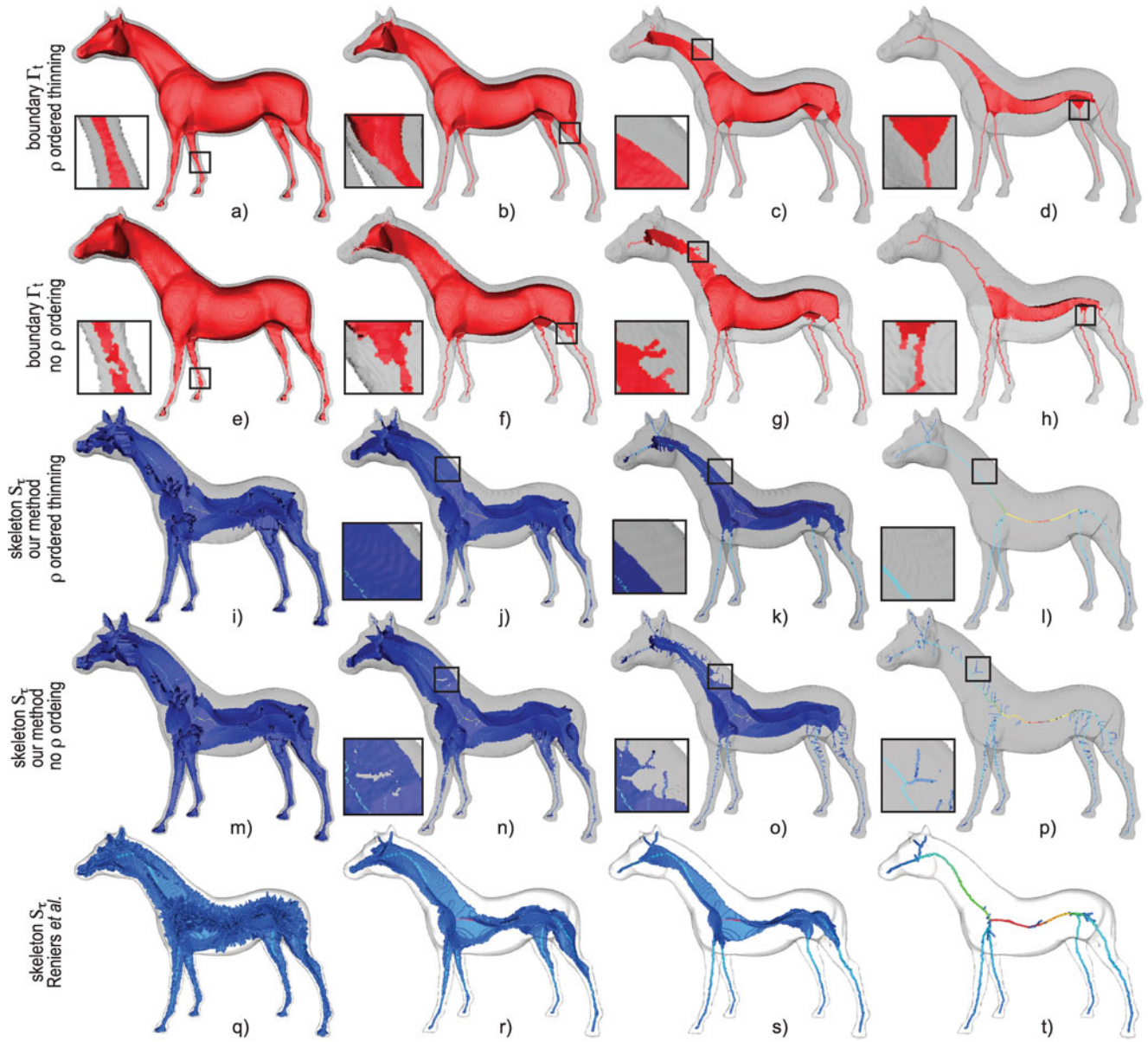

Fig. 2. Boundary Γ_t at four moments, with (a-d) and without (e-h) density-ordered thinning. Surface/curve skeletons, four simplification levels—our method with (i-l) and without (m-p) density-ordered thinning; Reniers et al. [53] (q-t). All skeletons are color-coded by importance λ using a rainbow colormap.

Our full skeletonization algorithm is now as follows (Algorithm. 1). During initialization, we compute the euclidean distance transform $DT_{\partial\Omega}$ on Ω (line 1). Next, we initialize the full-grid fields ρ (density), λ (importance) and M (binary description of the contracting shape) to their default values (line 2). We use $DT_{\partial\Omega}$ to label interior points $\mathbf{x} \in \Omega$ with $M(\mathbf{x}) = 2$ and initial boundary points $\mathbf{x} \in \partial\Omega$ with $M(\mathbf{x}) = 1$ respectively (lines 4-8). Finally, we set the density of boundary points to one and gather them in the set Q_1. This set will keep, during the algorithm execution, all points processed by the current algorithm iteration.

The main loop (lines 10-29) iteratively solves the system of Eqs. (6) and (7). Here, the field M has two roles. First, M labels points outside ($M = 0$), on the boundary ($M = 1$), and respectively inside ($M = 2$) the shrinking shape, thus efficiently keeps track of this shape. Second, we use M to check if a shape point is topologically simple or not: the function $simple(\mathbf{x}, M)$ returns true if removing $\mathbf{x}$ from the shape given by $M(\cdot) > 0$ does not change the shape's topology and false otherwise.

The first inner loop (lines 12-14) fills a set Q_2 with unprocessed, 26-connected (eight-connected in 2D) neighbors $\mathbf{y} \in \mathcal{N}(\mathbf{x})$ of the point $\mathbf{x}$ being processed. The set Q_2 captures points going to be processed in the next algorithm iteration (detailed further below). Next, we sort the current set Q_1 on increasing ρ (line 15), allowing the second inner loop (lines 17-19) to process these in increasing order of their density values. This second loop performs the detection phase of the thinning algorithm. Additionally, only points which are topologically-simple and within close distance (Δd) to the current Γ_t are added to set C for further processing. The other non-simple points are added to Q_2 for processing in the next iterations. The third inner loop (lines 21 to 25) performs the removal phase of the thinning algorithm.

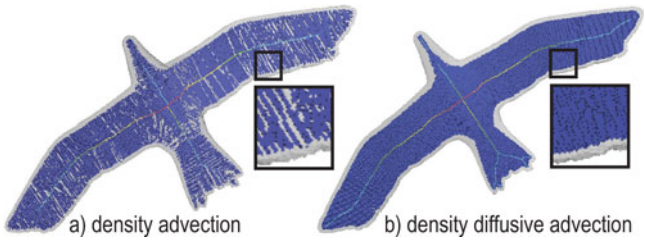

Fig. 3. Density transport via advection *vs* diffusive advection (see Section 4.3).

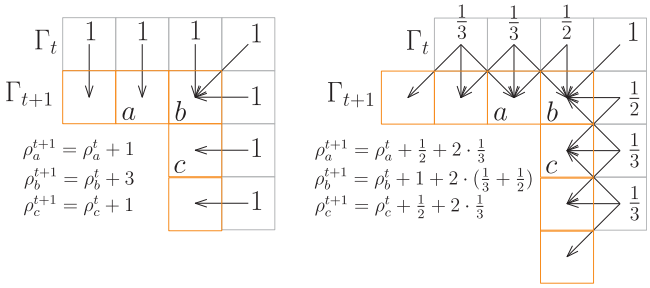

Fig. 4. Conservative advection *vs.* diffusive advection. Density is transported on the surface skeleton from Γ_t to Γ_{t+1} by *left*: conservative advection and *right*: diffusive advection. Arrows show the directions in which density is transported. The new density values at grid cells *a*, *b* and *c* are also shown.

Topologically-simple points $\mathbf{x} \in C$ are removed (by labeling them with $M(\mathbf{x}) = 0$) and collected in the narrow-band set B. At this stage, their importance λ is also computed (line 24). Non-simple points are added to Q_2.

Set B models the current boundary Γ_t, thus meeting R_1. As shown in Algorithm 1, B is built from current topologically-simple points (from Q_1) in increasing ρ order and by filtering them via the distance-threshold criterion (line 18). Hence, the boundary Γ_t is kept relatively smooth (discussed further in Section 4.3), thus R_3 is met. Once B is available, we can transport density (line 26) from points in B to interior points, thus meeting R_2. After density has been conservatively transported away from the current B, we set the density to zero at points $\mathbf{x} \in B$ (line 27).

At the end of the algorithm's main loop, the sets Q_1 and Q_2 are swapped (line 28). This is an essential aspect of our algorithm, as it facilitates an explicit and computationally-efficient representation of the boundary Γ_t (set B above). Having these sets, we can limit our computations only to a surface-like band of points around the current Γ_t, thus meeting R_4.

The algorithm stops when B becomes empty. For objects of genus 0, this happens when the shrunk shape and Q_1 contain only a single point, which is precisely the shape center C_Ω. This point clearly cannot be added to B since the topological constraint would be violated. For objects of higher genus, termination happens when the shrunk shape contains only curve-skeleton loops connected by non-terminal branches, a structure which cannot be shrunk any longer without disconnecting the skeleton (see example further in Section 4.4).

Let us now discuss the smoothness of the boundary Γ_t captured by the set B. Away from singular points of Eq. (7), Γ_t is captured (by the set B) as a level-set of $DT_{\partial\Omega}$, which is a Lipschitz-continuous function under the L_2 metric. At singular points of Eq. (7), our thinning still endorses Lipschitz continuity, but under the L_∞ metric. Intuitively, at such points, the order in which new inner points are processed in lines 12-14 reinforces the L_∞ metric when solving Eq. (6). Note that, although at non-singular points, topologically-constrained level sets [25] provide better smoothness properties of the evolving boundary (due to sub-pixel or sub-voxel precision), such methods have problems regarding the evolution of Γ_t along shocks of Eq. (7), as discussed in Section 4.1. In contrast, our approach produces a smooth shrinking Γ_t, as shown by Figs. 2a, 2b, 2c, and 2d.

4.3 Density Transport

We now focus on solving the mass conservation equation (Eq. (6); Algorithm 1, line 26). For 2D and 3D curve skeletons, discretizing Eq. (6) with the unconditionally-stable, semi-Lagrangian scheme in [35] suffices. However, generating progressively-simpler *surface* skeletons by simply thresholding the importance field λ requires additional work.

Since our algorithm solves Eq. (7) under the L_∞ norm along its singular points (see Section 4.2), and since noise, small errors and inacuracies due to the thinning process propagate into the density field evolution (Eq. (6)), simply thresholding λ would yield jaggies (indentations) in surface skeletons, for all but trivial shapes (see example in Fig. 3a). To tackle this, we propose a smoothing of the density field ρ, which leads to the desired importance field λ, as follows.

The key idea of [35] for solving conservative-advection PDEs similar to Eq. (3) is to follow so-called *characteristic curves* (along which the PDE becomes an ODE) both forwards and backwards in time, while ensuring that interpolation weights are equal to one for all grid cells, i.e., the advected density is conserved. We constrain the density ρ to be zero outside the shrinking shape, so we only need the forward step. Fig. 4 left shows a schematic example, assuming that density is transported along surface-skeleton points. For illustration simplicity, and without loss of generality, we next assume that ρ is one at all points in Γ_t. The density propagation directions, given by $\nabla\phi/\|\nabla\phi\|$ (Eq. (5)), are shown by arrows. Hence, as shown in Fig. 4, the (linear) interpolation weights equal one, and the new density values at grid cells *a*, *b* and *c* have the indicated values. Since all weights equal one, mass is conserved, as desired.

One way to tackle the above inaccuracies is by endorsing density advection with a (conservative) diffusive component, yielding a smoother evolution of ρ. For this, we propose an anisotropic diffusion process, which we dub *diffusive advection* (as opposed to the well-known diffusion-advection PDE). That is, instead of transporting density solely in the (potentially-noisy) gradient directions, we also allow density to diffuse to other surrounding nearby cells (Fig. 4 right). As can be seen by following the arrows, each 'donor cell' now contributes to multiple nearby cells. The weights along each arrow per donor cell, as well as the new density values of cells *a*, *b* and *c*, are also shown. More formally, let χ^t be the *characteristic function* of the shrinking shape, obtained, e.g., by upper-thresholding the field $M(\cdot)$ of Algorithm 1

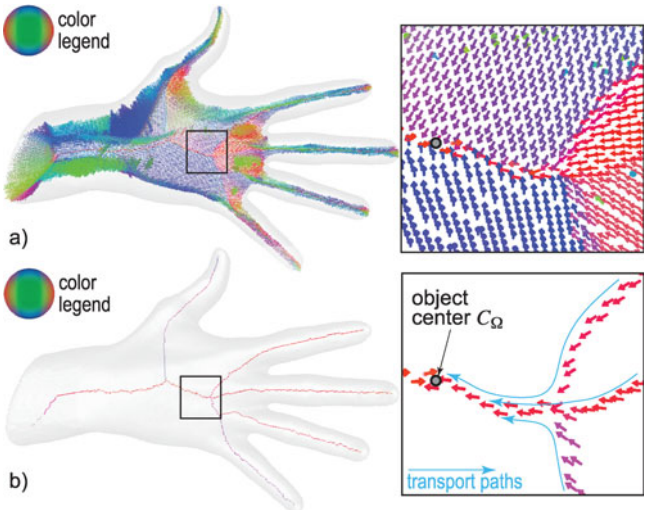

Fig. 5. Mass transport directions for the *hand* model over the surface skeleton (a) and curve skeleton (b), as computed by our model. See Section 4.3.

with value 1. Then, the new density value at a grid cell i of Γ_{t+1} for time-step $t+1$ can be expressed as

$$\rho_i^{t+1} = \rho_i^t + \sum_{j \in \mathcal{N}_i} \rho_j^t \frac{1-\chi_j^t}{\sum_{k \in \mathcal{N}_j} \chi_k^t}, \tag{12}$$

with being $\mathcal{N}_i$ the 26-connected neighborhood centered at i for the 3D case. Let $w_j^t = \frac{1-\chi_j^t}{\sum_{k \in \mathcal{N}_j} \chi_k^t}$ and assume that cell i receives density contributions from three surrounding cells $j \in \{1,2,3\}$ in Γ_t (see Fig. 4), i.e.,

$$\rho_i^{t+1} = \rho_i^t + \rho_1^t w_1^t + \rho_2^t w_2^t + \rho_3^t w_3^t. \tag{13}$$

This can be rewritten as

$$\rho_i^{t+1} - \rho_i^t = \sum_{k=1}^{3} w_k^t \left(\rho_k^t - \rho_i^t\right) + \rho_i^t \sum_{k=1}^{3} w_k^t$$

which is a discretization of anisotropic diffusion [49] with an additional reaction term

$$\frac{\partial \rho}{\partial t} = \nabla \cdot (w \nabla \rho) + \rho \mathscr{C}. \tag{14}$$

Note that $\mathscr{C}$ above could be seen as a simple curvature estimate. Comparing Eq. (6) with Eq. (14), we see that our simple discrete rule (Eq. (12)) replaces conservative advective transport by conservative diffusive transport, while taking into account the geometry of the evolving Γ_t. Finally, to force the density to flow more along gradient directions ($\mathbf{u}$, see Eq. (5)), we replace the weights w_j^t in Eq. (13) by

$$W_j^t = w_j^t / \left(1 + l_{ij}^2 / \sigma_a^2\right), \tag{15}$$

with $l_{ij} = \| \frac{\mathbf{x}_i - \mathbf{x}_j}{\|\mathbf{x}_i - \mathbf{x}_j\|} - \mathbf{u} \|$ and σ_a a sensitivity parameter. Thus, for small σ_a values, density transport happens mostly along gradient directions, as required by Eq. (6), whereas larger σ_a values result in density diffusion into cells nearby Γ_{t+1}.

Fig. 5 shows the density transport directions, generated by diffusive advection, for a simple test shape. Directions are shown only for the surface skeleton (a) and the curve skeleton (b), to reduce occlusion and increase visual readability, and are directionally color-coded (red = vectors aligned with the x shape axis, green = vectors aligned with y, blue = vectors aligned with z). As visible from both overview and detail images, the transport directions move density from the input surface along shortest paths to the surface skeleton, next to the curve skeleton, and next to the object center C_Ω, where all mass from $\partial\Omega$ ultimately collapses. The practical result thus matches well the model proposed in Section 3.2 (see also Fig. 1).

Fig. 3 compares the importance fields λ generated by the two density-transport methods above. Compared to conservative advection (Fig. 3a), diffusive advection creates a smooth density flow along the surface skeleton, leading to the desired smooth and monotonically-increasing importance field (Fig. 3b).

4.4 Detecting Different Skeleton Types

The importance field λ allows us to easily detect both skeleton types and the shape's center. Specifically, we have that $S_\Omega = \{\mathbf{x} \in \Omega \,|\, \lambda(\mathbf{x}) \geq 2.0\}$, since surface-skeleton points are, by definition, situated at equal distance from at least two different points on the boundary $\partial\Omega$ (Eq. (2)), and thus have an importance equal to at least that of two (collapsed) points of $\partial\Omega$, i.e. at least two. For genus 0 objects which admit a center in the sense denoted in Section 3.2, we have $C_\Omega = arg\,max_{\mathbf{x} \in \Omega} \lambda(\mathbf{x})$. Curve-skeleton points could be readily detected by upper thresholding the importance field λ with a large threshold τ. However, there are two shortcomings with this approach: First, the resulting curve skeleton may not be always one-voxel thick. Second, its extremities may be removed due to the large threshold value used. In other words, for high τ values, we would obtain a simplified, rather than a full, curve skeleton. To alleviate these issues we detect *salient* curve-skeleton points, during the shrinking process, using

$$CS_\Omega = \{x \in B \,|\, \rho(\mathbf{x}) > c\hat{T}(\mathbf{x}) \wedge end\ Point(\mathbf{x}, M)\}, \tag{16}$$

where $c > 0$ is a constant (explained next); $\hat{T}$ is a simple estimate for the time-of-arrival (approximating T from Eq. (10)), given by d in Algorithm 1; and $endPoint(\mathbf{x}, M)$ returns true if $\mathbf{x}$ is a curve-skeleton end point. We justify Eq. (16) as follows.

First, Eq. (16) only selects points from the surface skeleton, since only these have a density larger than two. Consider now a point $\mathbf{b}$ on the skeleton boundary ∂S_Ω, such that $\mathbf{b} \in S_\Omega \setminus CS_\Omega$ (Fig. 6). The density $\rho(\mathbf{b})$ equals the length of the circular segment $C(\mathbf{b}) \subset \partial\Omega$ (drawn green in Fig. 6), which is $\alpha DT_{\partial\Omega}(\mathbf{b}) = \alpha T(\mathbf{b})$ with α its subtended angle. Let $\mathbf{x}$ be a neighbour of $\mathbf{b}$ such that $T(\mathbf{x}) = T(\mathbf{b}) + 1$. Using the boundary evolution equation (Eq. (7)) and the arrival-time definition (Eq. (10)), it can be easily shown that $\mathbf{x}$ must be in the 'upstream' direction from $\mathbf{b}$, since $\mathbf{x} - \mathbf{b}$ and $\nabla\phi$ are parallel vectors. When the evolving boundary passes through $\mathbf{b}$ (i.e., $\mathbf{b}$ is removed by density-ordered thinning, see Algorithm 1), $\rho(\mathbf{b})$ is 'pushed' in the upstream direction through (diffusive) advection transport. Thus, $\mathbf{x}$ will directly receive most density of $\mathbf{b}$, under advective density transport. Moreover, due to the boundary-propagation order, when the

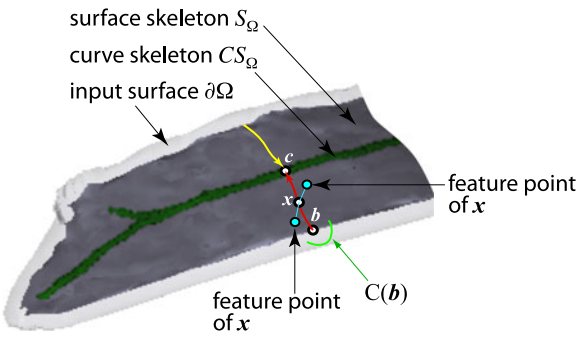

Fig. 6. Selecting curve-skeleton points for importance boosting (Section 4.4). The red arrow shows the upstream density-advection from **b** through **x** up to the curve-skeleton point **c**. The yellow arrow shows the second direction from which **c** receives density from the surface-skeleton.

interface is about to pass through **x**, point **x** must also be found to be part of the surface skeleton, and furthermore, $\rho(\mathbf{x}) > \rho(\mathbf{b})$. Indeed, since regular points $\mathbf{x} \in S_\Omega \setminus CS_\Omega$ also receive density contributions from its two (or more) feature-points on $\partial\Omega$ (boundary normals are preserved by the Eikonal equation), a rough estimate for the *minimum* density at **x** is $\rho(\mathbf{x}) > 2 + \alpha T(\mathbf{b})$. By a similar reasoning, for points $\mathbf{y} \in S_\Omega \setminus CS_\Omega$ situated in upstream direction from **b**, with $T(\mathbf{y}) > T(\mathbf{b}) + 1$, we find $\rho(\mathbf{y}) > 2T(\mathbf{y}) + \alpha T(\mathbf{b})$ as a lower bound on their density. Finally, points $\mathbf{c} \in CS_\Omega$ collect density from at least two neighbor-points $\mathbf{y} \in S_\Omega \setminus CS_\Omega$; this is so since the curve CS_Ω locally divides the 2D-manifold surface S_Ω in two parts. For instance, in Fig. 6, **c** will receive density from at least two surface-skeleton neighbor points situated at the tips of the red, respective yellow, arrows. Additionally, **c** receives a (larger) density contribution from (at least) another downstream curve-skeleton neighbor, so $\rho(\mathbf{c}) > 4\sum_i T(\mathbf{y}_i) \geq 4T(\mathbf{c})$. Hence, setting $c = 4$ in Eq. (16) performs conservative curve-skeleton detection. We verified empirically that setting $c = 4$ cleanly and robustly separates important curve-skeleton points from surface-skeleton points. Additionally, the end-point test in Eq. (16) ensures that only points which are at the sharp 'tips' of the shrinking surface Γ_t are considered as salient curve-skeleton candidates.

Once the points $\mathbf{x} \in CS_\Omega$ are found using Eq. (16), we 'boost' their importance during the iterative process by adding a constant fraction δ of the input shape's mass $|\partial\Omega|$. As a result, curve-skeleton points will have a significantly higher importance than surface-skeleton points. Also, as δ is constant for all $\mathbf{x} \in CS_\Omega$, the monotonic increase of importance along the curve skeleton branches is preserved. All in all, this allows us to threshold the final importance λ at precisely δ to cleanly and easily detect a full (unsimplified) CS_Ω, and to threshold λ at higher values to obtain progressively simplified curve skeletons.

Fig. 7 shows the effect of increasing the importance threshold τ for a shape of genus 3. Skeleton voxels are colored by importance λ via a rainbow colormap. The first three τ values (Figs. 7a, 7b, 7c) yield three increasingly simple surface skeletons. The last of these (Fig. 7c) is a mix of surface and curve skeleton parts, i.e., is an example of the meso skeletons described in [67]. As τ increases to δ, we find the full curve skeleton (Fig. 7d). Increasing τ further removes the end branches of the curve skeleton, without disconnecting its loops (Fig. 7e). For visual clarity, we used in Fig. 7e a local normalization of the colormap, based on the range of voxels in the respective simplified skeleton, rather than on the full importance range $[0, \lambda_{max}]$ used for all other figures. This shows better how the importance increases from the curve skeleton endpoints to its center, and is high over its loops. Finally, for the highest considered τ value, we get the fully simplified curve-skeleton without disconnections containing only loops connected by internal branches (Fig. 7f).

4.5 Implementation Details and Parameter Settings

Implementation. Our algorithm is implemented in C++ using OpenGL for skeleton rendering. We compute $DT_{\partial\Omega}$ on the CPU by the method of Meijster et al. [27], or on the GPU (if available) by the method of Cao et al. [13]. Both methods compute the exact euclidean distance transform and are linear in $|\Omega|$, the number of foreground pixels or voxels in the input shape. We implement $simple()$ using the Euler number for $\Omega \subset \mathbb{R}^2$ and Malandain's criterion [9] for $\Omega \subset \mathbb{R}^3$ respectively. We detect curve-skeleton endpoints (function $endPoint()$, Eq. (16)) as those voxels $\mathbf{x} \in B$ with one (26-connected) neighbor **y** for which $M(\mathbf{y}) > 0$. We use the integral method of Neumann et al. [46] to estimate the gradient directions along which density is transported (Eq. (5)). Sorting Q_1 is implemented by the standard template library (STL) *sort* function. Algorithm 1 requires K iterations of the loop in line 1, where K is roughly equal to $\frac{1}{2}\max_{\mathbf{x}\in\Omega} DT_{\partial\Omega}(\mathbf{x})$, the maximal thickness of Ω. Since at each iteration we sort the set Q_1, which is worst-case equal to the input boundary $\partial\Omega$, the total complexity of our method is $O(K|\partial\Omega|\log(|\partial\Omega|))$.

Parameters. Throughout the paper we use the following parameter settings. To obtain a good approximation of the motion equation (Eq. (7)), we set the distance step Δd in

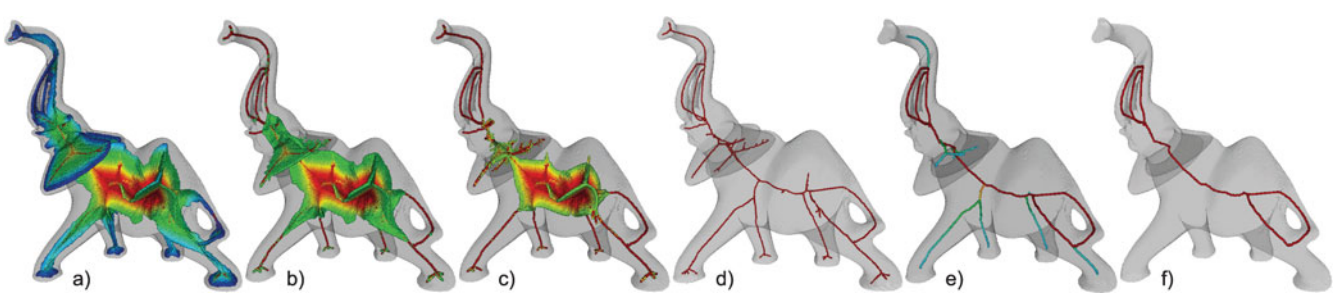

Fig. 7. Progressively simplified skeletons: surface skeletons (a-c) and curve skeletons (d-f) for six increasing τ values. See Section 4.4.

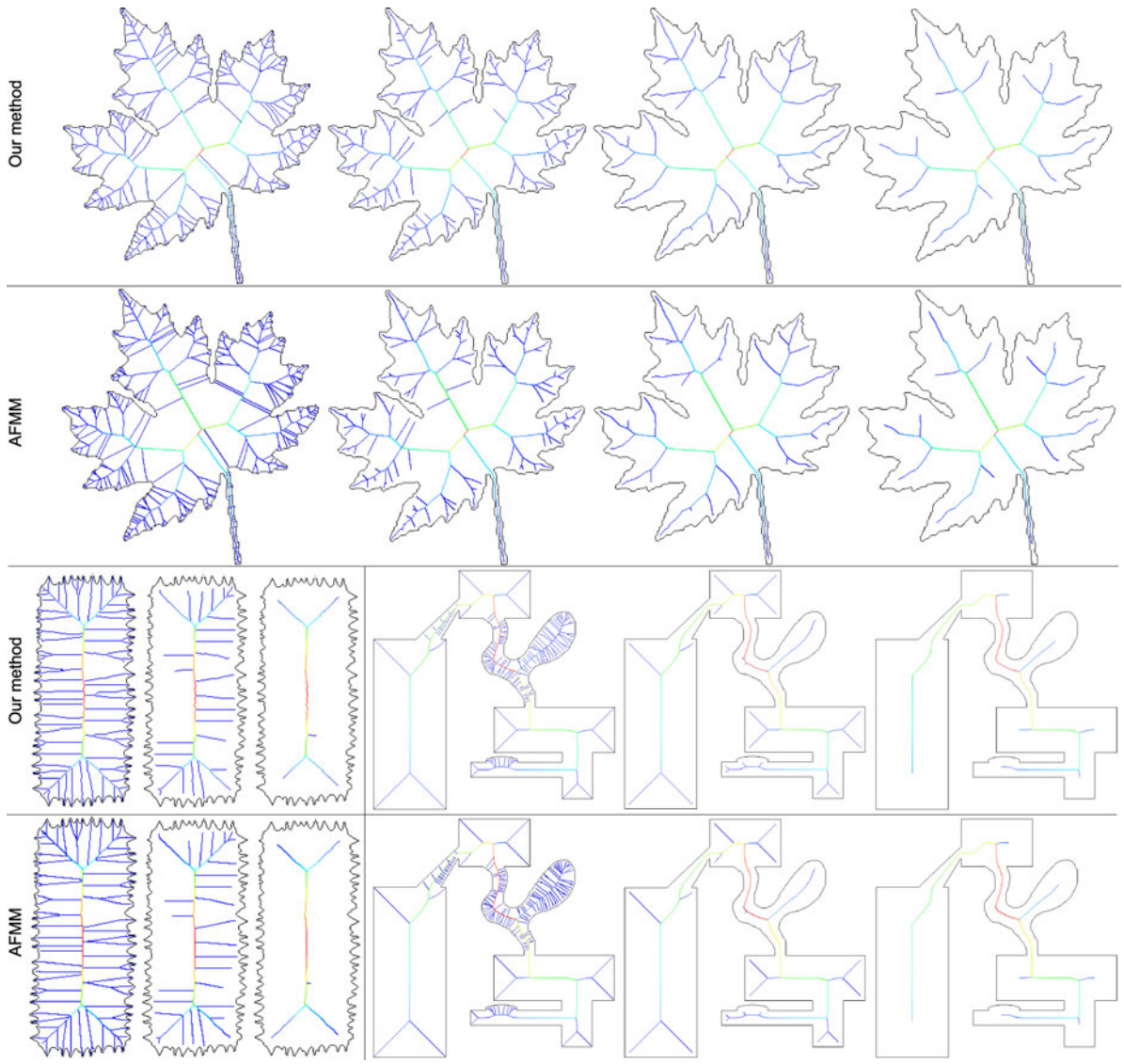

Fig. 8. Comparison of importance-colored 2D skeletons computed with our method and the AFMM [69], for increasing skeleton-simplification levels.

Algorithm 1 to $\Delta d = 1$. The parameter σ_a, controlling the density spread along gradient directions (Eq. (15)), is set to $\sigma_a = 0.2$. The parameter c in Eq. (16), i.e., the saliency of the detected curve-skeleton points, is set to $c = 4.0$. The parameter δ, representing the importance difference between curve and surface skeleton points (Section 4.4), is set to $\delta = 0.1$ of the total input surface mass $|\partial\Omega|$. The above values have been tested on a set of over 60 shapes, voxelized at various resolutions, and have consistently delivered good results like the ones shown in our figures here. As such, the only free parameter is the skeleton simplification threshold τ, whose use is explained in Section 4.4.

5 Comparative Results

5.1 Two-Dimensional Skeletons

For 2D shapes, we compared our method with the augmented fast marching method (AFMM) [69]. AFMM is a good example of 2D skeletonization methods that computes centered, accurate, connected, and pixel-thin skeletons regularized by the collapsed boundary-length importance metric (like [20], [47]).

Fig. 8 shows skeletons of several shapes from the database in [56] extracted with AFMM and with our method, for several simplification thresholds τ. Our method and the AFMM produce visually identical skeletons, both in terms of position, but also branches kept at a given τ. This is a non-trivial result, given that our method and the AFMM have completely different models behind. Moreover, since λ_{AFMM} at a skeleton point $\mathbf{x}$ equals the length of boundary that collapses to $\mathbf{x}$ when advected in $\nabla DT_{\partial\Omega}$, and since $\lambda \approx \lambda_{AFMM}$ (Fig. 8), this supports the claim that our λ is indeed equal to the collapsed boundary length.

5.2 Surface Skeletons

Fig. 9 (bottom six rows) compares our method with four voxel-based methods: multiscale skeletons [53], Hamilton-Jacobi (HJ) [59], integer medial axis (IMA) [27], and iterative thinning process (ITP) [31]; and with the multiscale mesh-based skeletonization (MBS) in [29]. Test shapes cover a wide range, including natural and synthetic, smooth and detailed, and objects of various genii (all voxelized at 512^3 resolution by *binvox* [48]). Our surface skeletons look very similar to

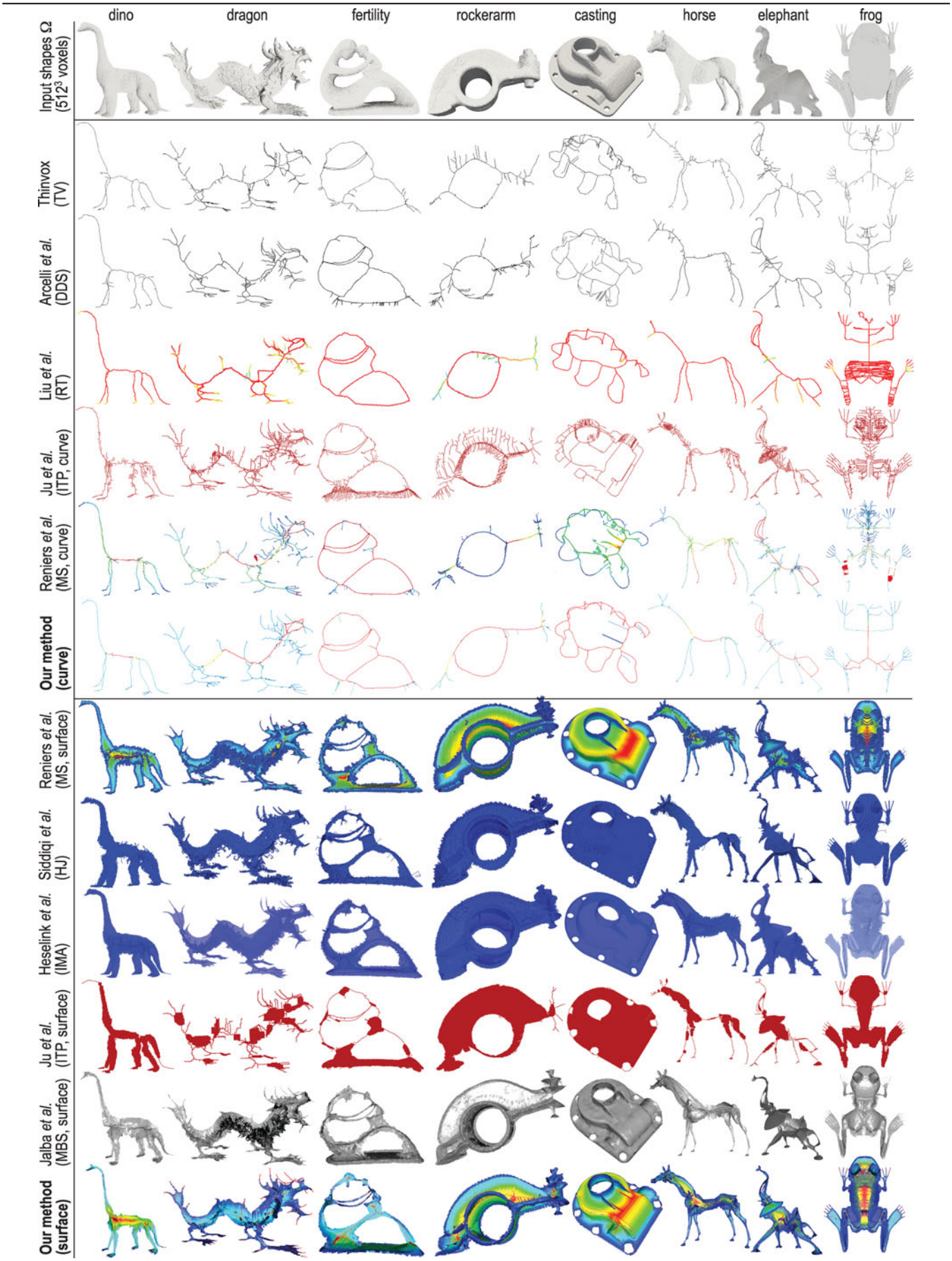

Fig. 9. Comparison of our 3D surface and curve skeletons with 10 related methods. Top 6 rows: curve skeletons. Bottom 6 rows: surface skeletons.

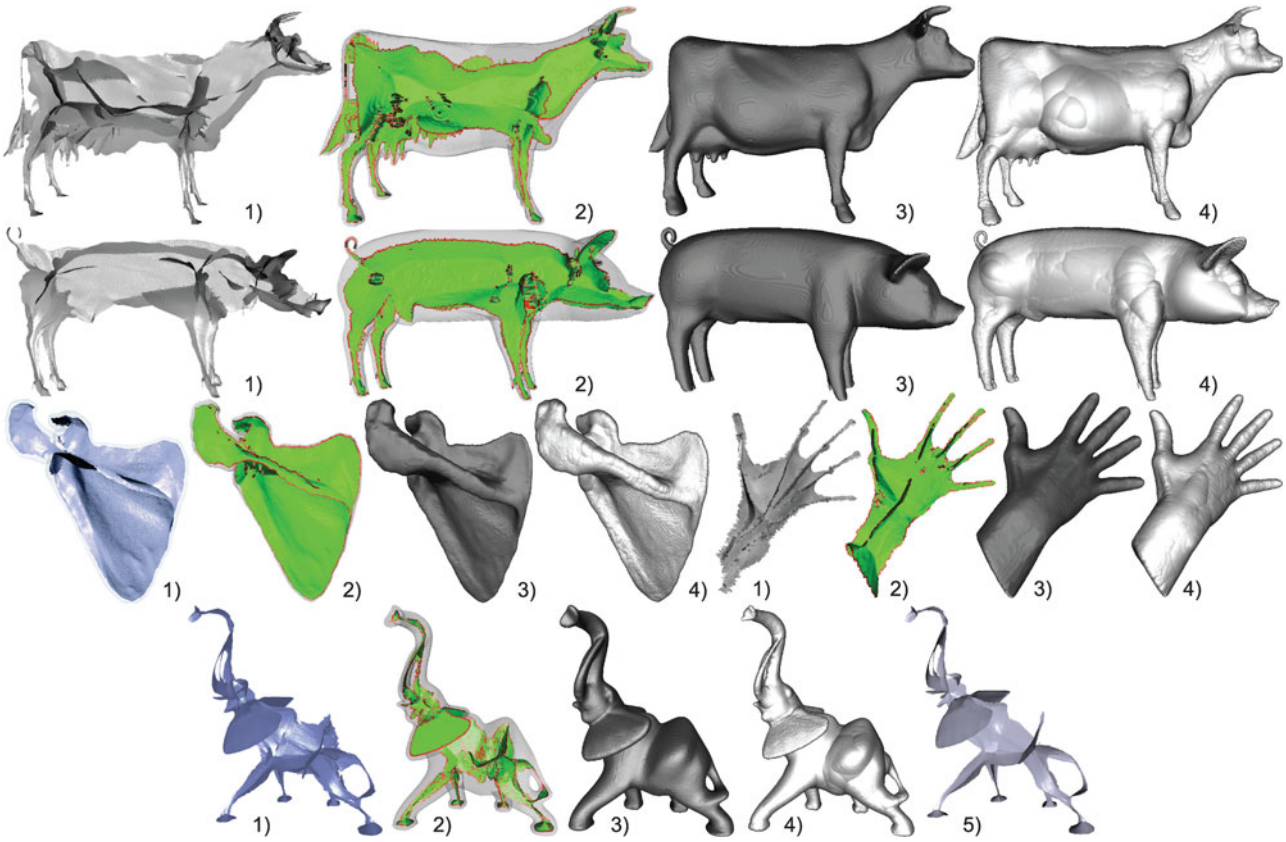

Fig. 10. Comparison of surface skeletons: (1) Jalba et al. (MBS) [29]; (2) our method; Original shapes (3) *vs* our skeleton-based reconstruction (4). All voxel models have a 512^3 resolution. Last row: detailed comparison, including also (5) the surface-skeleton of Miklos et al. [44].

those created by other methods, and show similar power in capturing the input shape genus and boundary details.

The comparison with MS is particularly interesting. To our knowledge, MS is the only existing voxel-based technique that computes multiscale surface skeletons whose importance uses a boundary-collapse metric. For MS, this metric is

$$\lambda_{SS}(\mathbf{x} \in S) = \min_{\gamma=(\mathbf{f}_1^{\mathbf{x}} \rightsquigarrow \mathbf{f}_2^{\mathbf{x}}) \subset \partial\Omega} \|\gamma\| \tag{17}$$

i.e., the length $\|\gamma\|$ of the shortest geodesic path γ on $\partial\Omega$ between the two feature points $\mathbf{f}_1^{\mathbf{x}}$ and $\mathbf{f}_2^{\mathbf{x}}$ of a skeleton point $\mathbf{x}$.

As for the 2D case (Section 5.1), we see that our surface skeletons *and* importance values (color-coded in Fig. 9, row 12, by a rainbow colormap) are very similar to the MS ones (Fig. 9, row 7). Fig. 2, two bottom rows, details this insight by showing four surface skeletons obtained by thresholding our importance λ, and λ_{SS}, at four increasing values. The monotonic increase of both λ and λ_{SS}, from low values on the surface-skeleton boundary to high values on the curve skeleton, and the resulting skeletons, are similar. This is an even more interesting result than the similarity of our results with the AFMM. Our importance $\lambda(\mathbf{x})$ equals the amount of boundary mass which reaches $\mathbf{x}$ subject to Eqs. (6), (7). The fact that $\lambda \approx \lambda_{SS}$ supports the conjectures in [18], [53] that all boundary points on such a geodesic γ, if advected in $\nabla DT_{\partial\Omega}$, would reach the skeleton point $\mathbf{x}$. However, a formal proof of these conjectures still lacks. Separately, Fig. 2 (bottom row) shows that our λ monotonically increases as we advance inwards on the skeletal structures (Section 4), hence that thresholding λ yields connected skeletons.

To better qualitatively assess our surface skeletons, Fig. 10 compares these with skeletons computed by the high-resolution MBS method [29], which uses the multiscale-importance in [53], but has a different skeleton detector, and uses a mesh rather than a voxel representation. For more insight, we colored our surface-skeleton border voxels red. Our skeletons are very similar to the MBS ones. Our method captures roughly the same amount of skeleton detail as MBS, even though the latter uses mesh, rather than voxel, skeleton-and-shape representations. Fig. 10 (4) shows our reconstruction of the input shape by drawing balls centered at the surface-skeleton voxels $\mathbf{x}$ and whose radii equal $DT_{\partial\Omega}(\mathbf{x})$, using the rendering technique in [29]. As visible, our reconstructions are very close to the input shapes (Fig. 10 (3)). The small bubble-like differences are explained by the fixed resolution of the voxel grid. This verifies the reconstructibility criterion for our method and, implicitly, shows that our skeletons are correctly centered. The last row in Fig. 10 compares our method for the *elephant* shape from Fig. 7, which has three tunnels, large thin-and-flat areas (ears), near-cylindrical parts (legs), high positive-curvature areas (ear borders), and high negative-curvature areas (ear-head junctions). Our surface skeleton is very close to the one produced by MBS, and also to the

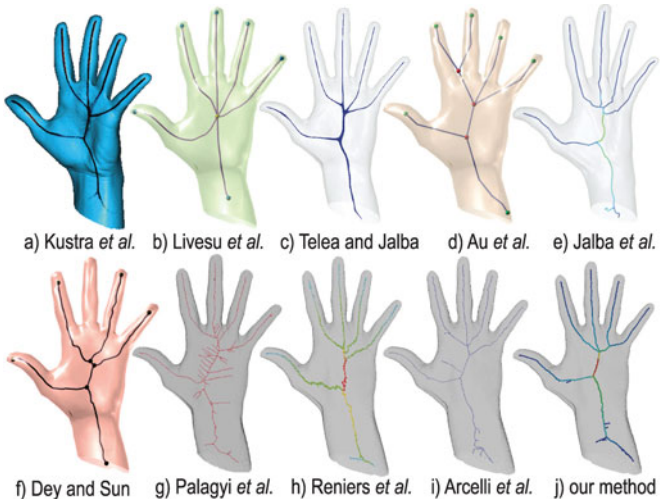

Fig. 11. Comparison of curve-skeletonization methods. Mesh-based methods (a-e). Voxel-based methods (f-i). Our method (j).

skeleton produced by the discrete-scale axis (DSA) mesh-based method of Miklos et al. [44], one of the highest-accuracy existing surface-skeletonization methods.

5.3 Curve Skeletons

Fig. 9 (top six rows) compares our method with five curve-skeleton methods: Thinvox (TV) [48], distance-driven skeletonization (DDS) [3], robust thinning (RT) [39], iterative thinning process [31], and multiscale skeletons [53]. In contrast to surface skeletons, we see now more variation between the compared methods. Our method delivers consistently thin (unlike MS), smooth (unlike ITP), noise-free (unlike ITP), and genus-preserving (unlike RT and MS) curve skeletons. Fig. 11 shows extra insight, by comparing our method with six additional mesh-based curve-skeletonization methods (Kustra et al. [34]; Livesu et al. [40]; Telea and Jalba [30]; Au et al. [6]; Dey and Sun [18]; and Jalba et al. [29]). As visible, our method yields well-centered curve skeletons which compare favorably, in terms of smoothness and lack of spurious branches, with the highest-quality mesh-based skeletons.

As for surface skeletons (Section 5.2), let us detail the parallel with MS curve-skeletons. MS detects curve skeletons as those points having at least two equal-length geodesics on $\partial\Omega$ between their feature points (MGF criterion in [18]). MS extends MGF by assigning a curve-skeleton importance λ_{CS} equal to the *area* bounded on $\partial\Omega$ by the above two geodesics. Figs. 9 and 11 show that our curve-skeletons and MS are very similar. The importances λ and λ_{CS} are also quite similar (see Fig. 2l *vs* Fig. 2t and Fig. 9, row 5 *vs* 6), except for the *rockerarm*, *casting*, and *frog* models, where λ_{CS} is smaller. Upon closer inspection, this shows a defect of MS: Low λ_{CS} points appear on curve-skeleton loops, whose geodesics do *not* cut $\partial\Omega$ in two separate parts according to the Jordan theorem [70], so following [53] these points should get a high importance, to prevent loop disconnection when simplifying skeletons. Our method correctly finds such loops and assigns them a high importance.

6 Discussion

6.1 Method Properties

Unification. Our method extracts 2D skeletons, 3D surface and 3D curve multiscale skeletons. To our knowledge, this is the first time that all these three skeleton types, including multiscale regularization, are computed by a single method which uses a single simplification metric. From both a theoretical and a practical perspective, we believe this to be an important result.

Robustness and accuracy. Our method computes thin, centered, homotopy-preserving, and connected skeletons from potentially noisy 2D and 3D shapes of arbitrary genus. Fig. 12 quantitatively compares our skeletons with four other methods, using the technique in [61]. In detail, given

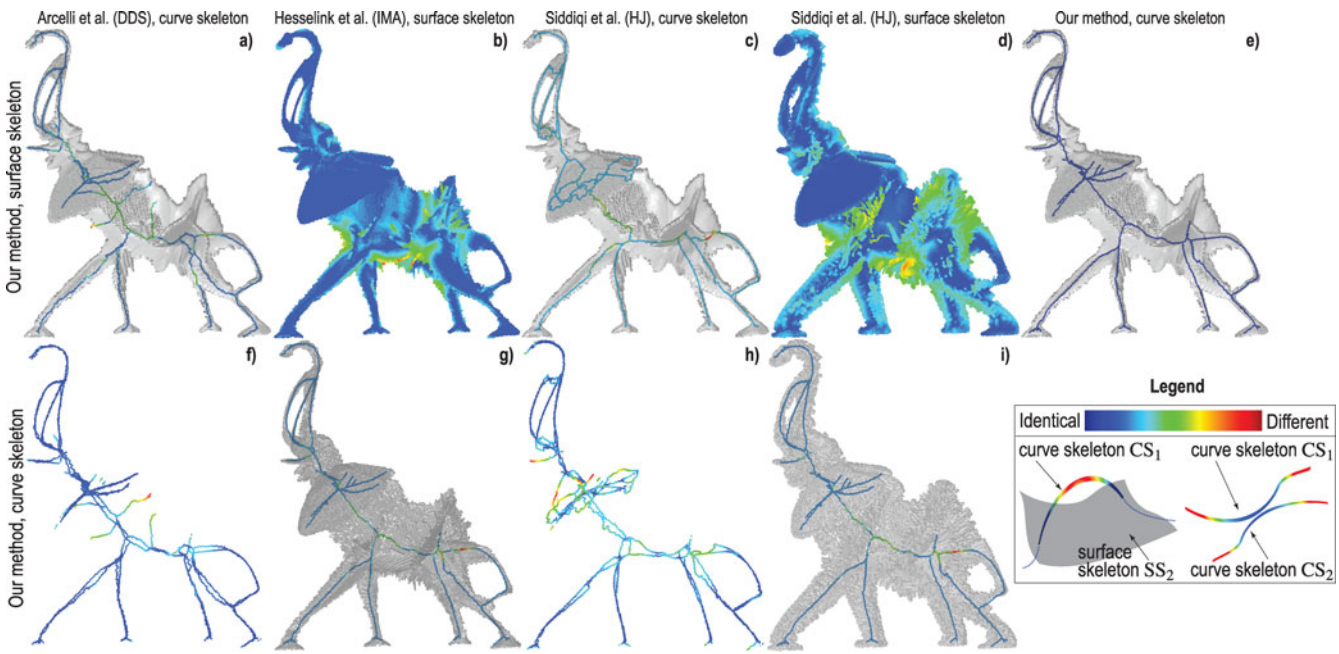

Fig. 12. Quantitative comparison of surface and curve skeletons. Color mapping indicates skeleton differences (see Section 6.1).

TABLE 1
Performance Comparison

Model (512^3 voxels)	$\|\Omega\|$	$\|\partial\Omega\|$	Our method	TV	DDS	RT	ITP (curve)	MS (curve)	MS (surface)	HJ	IMA	ITP (surface)	MBS (surface)
Dino	3,952,385	214,232	364.8 (10.83 sec)	109.5	9.9	14.7	60.9	30.6	38.32	36.9	62.5	66.5	1.62 sec
Dragon	1,208,845	119,144	421.4 (2.86 sec)	61.2	10.6	10.8	25.7	7.2	27.2	14.8	45.8	25.7	14.0 sec
Fertility	7,010,557	343,667	365.5 (19.18 sec)	133.0	13.2	16.8	51.9	15.4	25.6	30.9	158.0	51.9	4.13 sec
Rockerarm	2,726,188	186,362	500.1 (5.45 sec)	111.4	14.3	22.3	51.9	17.9	13.5	52.3	99.4	49.3	4.2 sec
Casting	4,280,458	544,345	400.6 (10.68 sec)	109.8	12.6	19.7	48.6	9.3	29.1	49.7	87.5	53.2	3.28 sec
Horse	9,235,212	329,222	361.9 (25.51 sec)	120.5	10.2	25.9	54.9	13.7	32.9	52.1	94.3	54.9	13.65 sec
Elephant	7,885,332	310,919	294.9 (26.73 sec)	112.8	10.6	9.1	43.8	20.8	30.4	42.3	109.7	42.2	3.92 sec
Frog	7,796,250	323,397	335.5 (23.23 sec)	123.0	12.9	16.0	48.1	10.5	63.3	54.1	148.5	65.6	2.74 sec

Skeletonization speeds in columns 4-13 are given in foreground voxels/millsecond.

two (curve or surface) skeletons S_1 and S_2, we first define the distance field

$$D_{12}(\mathbf{x} \in \Omega) = \begin{cases} \min_{\mathbf{y} \in S_2} \|\mathbf{x} - \mathbf{y}\| = DT_{S_2}(\mathbf{x}) & \text{if } \mathbf{x} \in S_1, \\ 0 & \text{if } \mathbf{x} \notin S_1. \end{cases}$$

To compare two same-kind (curve or surface) skeletons, we draw the field $D_{12} + D_{21}$ over the voxels $S_1 \cup S_2$, normalized by its maximum value, using a rainbow colormap. Close skeleton fragments are blue, while outlier ones are red. Comparing a curve skeleton CS_1 with a surface skeleton SS_2 shows how well is CS_1 *contained* within SS_2. For this, we color voxels in CS_1 by D_{12}, and voxels in $SS_2 \setminus CS_1$ with gray. Hence, voxels in $CS_1 \cap SS_2$ are blue, and voxels in CS_1 far from SS_2 become red (Fig. 12 inset). Looking at Fig. 12, we see that our surface skeletons are very similar to those produced by IMA and HJ (Figs. 12b, 12d). Warm colors, showing differences, occur mainly on the surface-skeleton boundary, and are due to the different simplification methods (and simplification levels) used by the compared methods. Our curve skeletons are fully contained in our surface skeletons, as expected (Fig. 12e), but also nearly fully contained in the IMA and HJ skeletons (Figs. 12g, 12i). Conversely, the DDS and HJ curve skeletons are well contained in our surface skeletons (Figs. 12a, 12c). The largest differences, found between curve skeletons themselves (Figs. 12f, 12h), are still quite small in absolute value.

Scalability. Table 1 (column 4) shows the speed of our method, implemented in C++ on a Linux 3.5 GHz PC with 32 GB RAM and an NVidia 690 GTX for the shapes in Fig. 9. Columns 2 and 3 give the shapes' surface areas $|\partial\Omega|$ and volumes $|\Omega|$, in voxels. We note a high and relatively shape-independent throughput (foreground voxels/second), in line with the complexity stated in Section 4.5. Compared to the tested voxel-based methods (columns 5-13), our method is one order of magnitude faster on average. The next-fastest method is TV [48]. However, TV does not compute surface skeletons nor an importance metric. Compared to MS, the only other voxel-based method which computes a multiscale importance metric, we are on average 10 times faster. The last column in Table 1 shows the speed of the MBS mesh-based skeletonization in [29], the second other method we are aware of (apart from MS) which computes multiscale skeletons. Compared to our absolute timings (Table 1, column 4, figures in brackets), MBS is 2.6 times faster on average. However, MBS is parallelized on the GPU, while our method is sequential and on the CPU. Our cost is essentially dominated by the number of foreground voxels (Section 4.5), while MBS is dominated by the cost of its geodesic tracing, which is $O(n^{3/2})$ for a meshed surface of n vertices. As such, our method is of comparable speed or even faster than MBS for shapes like *dragon* or *rockerarm*, which have a low volume (thus, generate few voxels for our method) but are represented by highly-refined meshes (thus, are costly for MBS). Conversely, our method is about ten times slower than MBS for shapes having a high volume, and whose mesh-representation uses few vertices, e.g., *dino* and *frog*.

Simplicity. Our framework has a single, simple, 2D and 3D implementation (under 2,000 lines of C++), with no complex computational geometry operations or degenerate cases, unlike some mesh-based methods [44], [53], [64]. Its only user parameter, the importance threshold τ, is simple to use: given the initial uniform density on $\partial\Omega$, τ selects skeletal structures which encode input-shape details whose length (in 2D) or area (in 3D) is larger than τ, similarly to [20], [47], [53], [69].

Classical properties. We next summarize the behavior of our method *vs* several recognized desirable skeleton properties. *1. Centeredness.* Centeredness is ensured by the unit-speed evolution of Γ_t (Eq. (5)). We verify centeredness, both for curve and surface skeletons, against several methods known to formally respect this property [3], [27], [53], [69] (see Fig. 10). *2. Thinness.* Our 2D and 3D skeletons are one-cell (pixel or voxel) thin, by construction. To argue this, suppose that this would not be so. Then, a skeleton λ_τ would (a) be thicker than one cell, and (b) would have, over a cross-section, the same importance λ. If such a thick cross-section existed, our shrinking algorithm would continue, since the shape can be further shrunk without disconnecting it (see Fig. 2). *3. Homotopy.* Homotopy of the skeleton with the input shape is guaranteed by construction, by the constraint in Eq. (4) and its corresponding implementation (Algorithm 1). *4. Reconstructibility.* The ability to reconstruct (smoothed versions of) the input shape from (simplified versions of) its skeleton is shown in Fig. 10. As visible, our reconstruction is quite accurate (compare e.g., Fig. 10 with Fig. 4 in [29]), modulo the natural limitation imposed by the fixed voxel grid. *5. Rotational invariance.* The discretization of our proposed PDE system (Eqs. (3)-(5)) described in Section 4.3 is rotationally invariant by construction. We

have verified that we indeed obtain nearly voxel-identical multiscale skeletons for the same input shape rotated at random angles with respect to the cell grid (not shown here for sake of brevity). *6. Curve*vs*surface skeletons.* Our curve skeletons are by construction included in the surface skeletons of the same shape, since they are both obtained by thresholding the same single-and-global importance field λ (see Section 4.4). *7. Multiscale and noise resistance.* The field λ describes the whole space between the input surface $\partial\Omega$, surface skeleton S_Ω, curve skeleton CS_Ω, and shape center C_Ω. Thresholding λ with increasing values yields the S_Ω from Ω; simplified surface skeletons (without branches due to small-scale shape details); the (simplified) CS_Ω; and the shape center, or zero-dimensional skeleton, of Ω. Reniers et al. get similar results, but they need two *separate* surface and curve skeleton importances and corresponding algorithms [53]. We compute λ making no distinction between the two skeleton types. Telea and Jalba compute multiscale curve skeletons by collapsing surface skeletons inwards, following the idea that the former can be seen as the medial loci of the latter [30]. Yet, as in [53], their surface and curve skeleton algorithms are fundamentally different, and they also do not propose a curve-skeleton importance metric. Thresholding λ at values τ between the average importance of S_Ω and CS_Ω yields a meso-skeleton structure [67] that continuously shrinks from S_Ω towards CS_Ω as τ increases (see Section 4.4).

Limitations. Our method stores four voxel scalar volumes (ρ, λ, M, and $DT_{\partial\Omega}$ in Alg. 1), *i.e.* can handle shapes up to roughly 1000^3 voxels on a 16 GB PC. Mesh-based skeletonization methods [29], [41] need far less memory. For instance, the mesh models for all shapes in this paper, which are up to 1 M triangles, need only 24 MB with the method in [29]. Separately, we acknowledge that our comparisons highlight differences between our skeletons and those produced by other methods, but do not explicitly show which skeletons are more suitable for a specific application, e.g., shape retrieval, classification, or segmentation. A thorough qualitative comparison with this goal is an important topic for future work.

6.2 Comparison with Hamiltonian Methods

Equation (3) is similar with Torsello and Hancock's mass conservation model $\nabla \cdot (\rho \mathbf{u}) = 0$ used for 2D skeletonization [5]. Yet, several key differences exist. Numerically, TH transforms the mass conservation $\nabla \cdot (\rho \mathbf{u}) = 0$ into a system of two ODEs (Eqs. (7) in [5]) by the substitution $\sigma = \log(\rho)$. These ODEs are solved with a second-order Crank-Nicholson divergence-discretization scheme coupled with semi-Lagrangian advection. In contrast, we use the *conservative* semi-Lagrangian scheme of Fedkiw [35], which only needs linear interpolation and clamping. This has several advantages. First, our scheme is numerically very stable, and conserves density well. Second, we do not need the second-order divergence discretization of TH. Third, by computing the density logarithm σ, TH also needs exponentiation to find the final density $\rho = \exp(\sigma)$. We noticed, in practice, that this creates important numerical problems, such as an infinite value of ρ at several points in Ω.

Torsello and Rossi extend the TH density advection to extract 3D medial surfaces [54]. In their model, density advection stops when reaching the surface skeleton. In contrast, we continue advection by collapsing the surface skeleton to the curve skeleton and the latter to the shape center, yielding all desired skeletal representations within a single process.

7 Conclusions

We have presented a unified framework for computing 2D skeletons and 3D surface and curve skeletons. We detect all skeleton types by a single algorithm, and also compute a single importance metric which assigns to each skeletal point the amount of (2D or 3D) input boundary described by that point. Comparing our skeletons and their computed importance with results computed by related methods shows a very good match. We present a simple implementation of our method which achieves good performance results on a range of complex 2D and 3D shapes, i.e., over three times faster than the fastest voxel-based skeletonization method we are aware of, and over 10 times faster than comparable multiscale methods.

Future work can target several directions. Porting our method to massively-parallel platforms (*e.g.*, CUDA) using sparse voxel grids will increase scalability. Separately, modulating the input-surface density by e.g., curvature or application-specific metrics would allow different feature-sensitive skeleton simplifications.

Acknowledgments

The authors acknowledge the financial support of CNPq, Brazil, through the grant 202535/2011-8, and the help of the authors of [3] in providing us their DDS method implementation. Andrei C. Jalba is the corresponding author.

References

[1] N. Ahuja and J. Chuang, "Shape representation using a generalized potential field model," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 19, no. 2, pp. 169–176, Feb. 1997.

[2] N. Amenta, S. Choi, and R. Kolluri, "The power crust," in *Proc. 6th ACM Symp. Solid Model. Appl.*, 2001, pp. 65–73.

[3] C. Arcelli, G. Sanniti di Baja, and L. Serino, "Distance-driven skeletonization in voxel images," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 33, no. 4, pp. 709–720, Apr. 2011.

[4] C. Aslan, A. Erdem, E. Erdem, and S. Tari, "Disconnected skeleton: Shape at its absolute scale," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 30, no. 12, pp. 2188–2203, Dec. 2008.

[5] A. Torsello and E. Hancock, "Correcting curvature-density effects in the Hamilton-Jacobi skeleton," *IEEE Trans. Image Process.*, vol. 15, no. 4, pp. 877–891, Apr. 2006.

[6] O. K. C. Au, C. Tai, H. Chu, D. Cohen-Or, and T. Lee, "Skeleton extraction by mesh contraction," in *Proc. ACM Annu. Conf. Comput. Graph. Interactive Techn.*, 2008, pp. 441–449.

[7] X. Bai and L. Latecki, "Path similarity skeleton graph matching," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 30, no. 7, pp. 1282–1292, Jul. 2008.

[8] X. Bai, L. Latecki, and W.-Y. Liu, "Skeleton pruning by contour partitioning with discrete curve evolution," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 29, no. 3, pp. 449–462, Mar. 2007.

[9] G. Bertrand and G. Malandain, "A new characterization of three-dimensional simple points," *Pattern Recognit. Lett.*, vol. 2, no. 15, pp. 169–175, 1994.

[10] T. Biben, K. Kassner, and C. Misbah, "Phase-field approach to three-dimensional vesicle dynamics," *Phys. Rev. E*, vol. 72, p. 041921, 2005.

[11] S. Bouix, K. Siddiqi, and A. Tannenbaum, "Flux driven automatic centerline extraction," *Med. Image Anal.*, vol. 9, no. 3, pp. 209–221, 2005.

[12] J. Cao, A. Tagliasacchi, M. Olson, H. Zhang, and Z. Su, "Point cloud skeletons via Laplacian based contraction," in *Proc. IEEE 6th ACM Shape Model. Int. Conf.*, 2010, pp. 187–197.

[13] T. Cao, K. Tang, A. Mohamed, and T. Tan, "Parallel banding algorithm to compute exact distance transform with the GPU," in *Proc. SIGGRAPH Interactive 3D Graph. Games Symp.*, 2010, pp. 134–141.

[14] M. Chang, F. Leymarie, and B. Kimia, "Surface reconstruction from point clouds by transforming the medial scaffold," *Comput. Vis. Image Understanding*, vol. 113, no. 11, pp. 1130–1146, 2009.

[15] M. Chuang and M. Kazhdan, "Fast mean-curvature flow via finite-elements tracking," *Comput. Graph. Forum*, vol. 30, no. 6, pp. 1750–1760, 2011.

[16] N. Cornea, D. Silver, X. Yuan, and R. Balasubramanian, "Computing hierarchical curve-skeletons of 3D objects," *Vis. Comput.*, vol. 21, no. 11, pp. 945–955, 2005.

[17] J. Damon, "Global medial structure of regions in $\mathbb{R}^3$," *Geometry Topol.*, vol. 10, pp. 2385–2429, 2006.

[18] T. Dey and J. Sun, "Defining and computing curve skeletons with medial geodesic functions," in *Proc. 4th Eurograph. Symp. Geometry Process.*, 2006, pp. 143–152.

[19] T. Dey and W. Zhao, "Approximating the medial axis from the Voronoi diagram with a convergence guarantee," *Algorithmica*, vol. 38, pp. 179–200, 2003.

[20] A. Falcão, L. Costa, and B. da Cunha, "Multiscale skeletons by image foresting transform and its applications to neuromorphometry," *Pattern Recognit.*, vol. 35, no. 7, pp. 1571–1582, 2002.

[21] M. Foskey, M. Lin, and D. Manocha, "Efficient computation of a simplified medial axis," in *Proc. Shape Model.*, 2003, pp. 135–142.

[22] Y. Ge and J. Fitzpatrick, "On the generation of skeletons from discrete euclidean distance maps," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 18, no. 11, pp. 1055–1066, Nov. 1996.

[23] P. Giblin and B. Kimia, "A formal classification of 3D medial axis points and their local geometry," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 26, no. 2, pp. 238–251, Jan. 2004.

[24] J. Giesen, B. Miklos, M. Pauly, and C. Wormser, "The scale axis transform," in *Proc. 25th Annu. Symp. Comput. Geometry*, 2009, pp. 106–115.

[25] X. Han, C. Xu, and J. L. Prince, "A topology preserving level set method for geometric deformable models," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 25, no. 6, pp. 755–768, Jun. 2003.

[26] M. Hassouna and A. Farag, "Variational curve skeletons using gradient vector flow," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 31, no. 12, pp. 2257–2274, Dec. 2009.

[27] W. Hesselink and J. Roerdink, "Euclidean skeletons of digital image and volume data in linear time by the integer medial axis transform," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 30, no. 12, pp. 2204–2217, Dec. 2008.

[28] M. Iwanowski and P. Soille, "Fast algorithm for order independent binary homotopic thinning," in *Proc. 8th Int. Conf. Adaptive Natural Comput. Algorithms*, 2007, pp. 606–615.

[29] A. Jalba, J. Kustra, and A. Telea, "Surface and curve skeletonization of large 3D models on the GPU," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 35, no. 6, pp. 1495–1508, Jun. 2013.

[30] A. Jalba and A. Telea, "Computing curve skeletons from medial surfaces of 3D shapes," in *Proc. Theory Practice Comput. Graph*, 2012, pp. 99–106.

[31] T. Ju, M. Baker, and W. Chiu, "Computing a family of skeletons of volumetric models for shape description," *Comput. Aided Des.*, vol. 39, no. 5, pp. 352–360, 2007.

[32] S. Katz and A. Tal, "Hierarchical mesh decomposition using fuzzy clustering and cuts," *ACM Trans. Graph.*, vol. 22, no. 3, pp. 954–961, 2003.

[33] R. Kimmel, D. Shaked, N. Kiryati, and A. Bruckstein, "Skeletonization via distance maps and level sets," *Comput. Vis. Image Understanding*, vol. 62, no. 3, pp. 382–391, 1995.

[34] J. Kustra, A. Jalba, and A. Telea, "Probabilistic view-based 3D curve skeleton computation on the GPU," in *Proc. 8th Int. Joint Conf. Comput. Vis., Imaging Comput. Graph. Theory Appl.*, 2013, vol. 2, pp. 237–246.

[35] M. Lentine, J. Gretarsson, and R. Fedkiw, "An unconditionally stable fully conservative semi-Lagrangian method," *J. Comput. Phys.*, vol. 230, no. 8, pp. 2857–2879, 2011.

[36] F. Leymarie and B. Kimia, "The medial scaffold of 3D unorganized point clouds," *IEEE Trans. Vis. Comput. Graph.*, vol. 29, no. 2, pp. 313–330, Feb. 2007.

[37] F. Leymarie and M. Levine, "Simulating the grassfire transform using an active contour model," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 14, no. 1, pp. 56–75, Jan. 1992.

[38] X. Li, T. Woon, T. Tan, and Z. Huang, "Decomposing polygon meshes for interactive applications," in *Proc. Interactive 3D Symp.*, 2001, pp. 35–42.

[39] L. Liu, E. Chambers, D. Letscher, and T. Ju, "A simple and robust thinning algorithm on cell complexes," *Comput. Graph. Forum*, vol. 29, no. 7, pp. 2253–2260, 2010.

[40] M. Livesu, F. Guggeri, and R. Scateni, "Reconstructing the curve-skeletons of 3D shapes using the visual hull," *IEEE Trans. Vis. Comput. Graph.*, vol. 18, no. 11, pp. 1891–1901, Nov. 2012.

[41] J. Ma, S. W. Bae, and S. Choi, "3D medial axis point approximation using nearest neighbors and the normal field," *Vis. Comput.*, vol. 28, no. 1, pp. 7–19, 2012.

[42] G. Malandain and S. Fernandez-Vidal, "Euclidean skeletons," *Image Vis. Comput.*, vol. 16, no. 5, pp. 317–327, 1998.

[43] P. Maragos and M. A. Butt, "Curve evolution, differential morphology, and distance transforms applied to multiscale and eikonal problems," *Fundamental Inf.*, vol. 41, no. 1,2, pp. 91–129, 2000.

[44] B. Miklos, J. Giesen, and M. Pauly, "Discrete scale axis representations for 3D geometry," in *Proc. ACM SIGGRAPH Paper*, 2010, pp. 394–493.

[45] M. Mortara, G. Patanet, M. Spagnuolo, B. Falcidieno, and J. Rossignac, "Plumber: A method for multiscale decomposition of 3D shapes into tubular primitives and bodies," in *Proc. ACM Symp. Solid Model. Appl.*, 2004, pp. 339–344.

[46] L. Neumann, B. Csébfalvi, A. König, and E. Gröller, "Gradient estimation in volume data using 4D linear regression," *Comput. Graph. Forum*, vol. 19, no. 3, pp. 351–358, 2000.

[47] R. L. Ogniewicz and O. Kubler, "Hierarchic Voronoi skeletons," *Pattern Recognit.*, vol. 28, pp. 343–359, 1995.

[48] K. Pálagyi and A. Kuba, "Directional 3D thinning using 8 subiterations," in *Proc. Discrete Geometry Comput. Imagery*, 1999, pp. 325–336.

[49] P. Perona and J. Malik, "Scale-space and edge detection using anisotropic diffusion," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 12, no. 7, pp. 629–639, Jul. 1990.

[50] S. Pizer, K. Siddiqi, G. Szekely, J. Damon, and S. Zucker, "Multiscale medial loci and their properties," *Int. J. Comput. Vis.*, vol. 55, nos. 2/3, pp. 155–179, 2003.

[51] S. Prohaska and H. C. Hege, "Fast visualization of plane-like structures in voxel data," in *Proc. IEEE Vis.*, 2002, pp. 29–36.

[52] C. Pudney, "Distance-ordered homotopic thinning: A skeletonization algorithm for 3D digital images," *Comput. Vis. Image Understanding*, vol. 72, no. 3, pp. 404–413, 1998.

[53] D. Reniers, J. J. van Wijk, and A. Telea, "Computing multiscale skeletons of genus 0 objects using a global importance measure," *IEEE Trans. Vis. Comput. Graph.*, vol. 14, no. 2, pp. 355–368, Mar./Apr. 2008.

[54] L. Rossi and A. Torsello, "An adaptive hierarchical approach to the extraction of high resolution medial surfaces," in *Proc. 3D Imag., Model., Process., Vis. Transmiss.*, 2012, pp. 371–378.

[55] M. Rumpf and A. Telea, "A continuous skeletonization method based on level sets," in *Proc. Symp. Data Vis.*, 2002, pp. 151–158.

[56] T. Sebastian, P. Klein, and B. Kimia, "Recognition of shapes by editing their shock graphs," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 26, no. 5, pp. 550–571, May 2004.

[57] J. Sethian, *Level Set Methods and Fast Marching Methods*. Cambridge, U.K.: Cambridge Univ. Press, 2002.

[58] D. Shaked and A. Bruckstein, "Pruning medial axes," *Comput. Vis. Image Understanding*, vol. 69, no. 2, pp. 156–169, 1998.

[59] K. Siddiqi, S. Bouix, A. Tannenbaum, and S. Zucker, "Hamilton-Jacobi skeletons," *Int. J. Comput. Vis.*, vol. 48, no. 3, pp. 215–231, 2002.

[60] K. Siddiqi and S. Pizer, *Medial Representations: Mathematics, Algorithms and Applications*. New York, NY, USA: Springer, 2009.

[61] A. Sobiecki, A. Jalba, and A. Telea, "Comparison of curve and surface skeletonization methods for voxel shapes," *Pattern Recog. Lett.*, vol. 47, pp. 147–156, 2014.

[62] A. Sobiecki, H. Yasan, A. Jalba, and A. Telea, "Qualitative comparison of contraction-based curve skeletonization methods," in *Proc. 11th Int. Symp. Math. Morphol.*, 2013, pp. 425–439.

[63] S. Stolpner, S. Whitesides, and K. Siddiqi, "Sampled medial loci and boundary differential geometry," in *Proc. IEEE Workshop 3D Digital Imaging Model.*, 2009, pp. 87–95.
[64] S. Stolpner, S. Whitesides, and K. Siddiqi, "Sampled medial loci for 3D shape representation," *Comput. Vis. Image Understanding*, vol. 115, no. 5, pp. 695–706, 2011.
[65] R. Strzodka and A. Telea, "Generalized distance transforms and skeletons in graphics hardware," in *Proc. Symp. Data Vis.*, 2004, pp. 221–230.
[66] A. Sud, M. Foskey, and D. Manocha, "Homotopy-preserving medial axis simplification," in *Proc. ACM Symp. Solid Phys. Model.*, 2005, pp. 103–110.
[67] A. Tagliasacchi, I. Alhashim, M. Olson, and H. Zhang, "Mean curvature skeletons," *Comput. Graph. Forum*, vol. 31, no. 5, pp. 1735–1744, 2012.
[68] A. Tagliasacchi, H. Zhang, and D. Cohen-Or, "Curve skeleton extraction from incomplete point cloud," in *Proc. SIGGRAPH Papers*, 2009, pp. 541–550.
[69] A. Telea and J. J. van Wijk, "An augmented fast marching method for computing skeletons and centerlines," in *Proc. Symp. Data Vis.*, 2002, pp. 251–259.
[70] H. Thomas, "Jordan's proof of the Jordan curve theorem," *Stud. Logic, Grammar Rhetoric*, vol. 10, no. 23, pp. 567–582, 2007.
[71] M. van Dortmont, H. van de Wetering, and A. Telea, "Skeletonization and distance transforms of 3D volumes using graphics hardware," in *Proc. 13th Int. Conf. Discrete Geometry Comput. Imagery*, 2006, pp. 617–629.
[72] M. Wan, F. Dachille, and A. Kaufman, "Distance-field based skeletons for virtual navigation," in *Proc. IEEE Vis.*, 2001, pp. 239–246.

Andrei C. Jalba received the BSc degree in 1998 and the MSc degree in 1999 in applied electronics and information engineering from the "Politehnica" University of Bucharest, Romania. He received the PhD degree from the Institute for Mathematics and Computing Science, University of Groningen, The Netherlands, in 2004. Currently, he is an assistant professor at the Eindhoven University of Technology. His research interests include computer graphics and vision, shape processing, and parallel computing.

Andre Sobiecki received the BSc degree in technology information systems from Santa Catarina State University in 2009, and the MSc degree in electrical engineering from the Centro Universitario da FEI in 2012. Currently, he is working toward the PhD degree in computer science at the University of Groningen, The Netherlands. His general interests are in computer vision, segmentation, skeletonization, and digital inpainting of 2D and 3D images.

Alexandru C. Telea received the PhD degree 2000 in computer science from the Eindhoven University of Technology, The Netherlands. Until 2007, he was an assistant professor in visualization and computer graphics at the same university. Since 2007, he is a professor of computer science at the University of Groningen, The Netherlands. His interests include 3D multiscale shape processing, scientific and information visualization, and software analytics.