CHAPTER

2 Multiscale 2D medial axes and 3D surface skeletons by the image foresting transform

Alexandre Falcão[*], Cong Feng[†], Jacek Kustra[‡], Alexandru Telea[†]

Department of Information Systems, Institute of Computing, University of Campinas (UNICAMP), Campinas, SP, Brazil[*] Institute Johann Bernoulli, University of Groningen, Groningen, The Netherlands[†] Philips Research, Eindhoven, The Netherlands[‡]

Contents

2.1 INTRODUCTION

Medial descriptors, or skeletons, are used in many applications such as path planning, shape retrieval and matching, computer animation, medical visualization, and shape processing [45,50]. In two dimensions, such descriptors are typically called medial axes. Three-dimensional shapes admit two types of skeletons, *surface* skeletons, which are sets of manifolds with boundaries that meet along a set of so-called

Y-intersection curves [12,30,6], and *curve* skeletons, which are 1D structures locally centered in the shape [8].

A fundamental and well-known problem of skeleton computation is that skeletons are inherently unstable to small perturbations of the input shape [50]. This leads to the appearance of so-called spurious branches, which have little or no application value, and considerably complicate the analysis and usage of skeletons. To alleviate this, various simplification methods have been proposed to eliminate such branches. In this context, *multiscale* methods are particularly interesting. They compute a so-called importance metric for skeletal points, which encodes the scale of the shape details, and next offer a continuous way for simplifying skeletons by simply thresholding that metric.

Several robust, simple to implement, and efficient methods exist for computing 2D multiscale skeletons; see, e.g., [35,17,53]. For surface skeletons, the situation is very different: Only a few such methods exist, and these are either computationally expensive [14,38,39], complex [26], or sensitive to numerical discretization [27].

In this chapter, we address the joint problem of computing multiscale 2D medial axes and 3D surface skeletons by a new method. For this, we cast the problem of computing the importance metrics proposed in [35,17,53] (for 2D skeletons) and in [39,26] (for 3D skeletons) as the search for an optimal path forest using the Image Foresting Transform framework [19]. In 2D, the skeletons are one-pixel wide and connected in all scales for genus-0 shapes. In 3D, the surface skeletons are one-voxel wide and can be connected in all scales for genus-0 shapes if the curve skeleton points are detected [39,26], which we do not address here. Next, we provide simple and efficient algorithms to compute these metrics for both 2D and 3D binary images. Compared to 3D techniques that use the same multiscale importance metric, our method is faster [14,38,39] or alternatively considerably simpler to implement [26]. Compared to other 3D multiscale techniques, our method is far less sensitive to numerical noise [27]. Compared to all above techniques, our method yields the same quality level in terms of centeredness, smoothness, thinness, and ease to simplify the skeleton.

This chapter is structured as follows. In Section 2.2, we overview multiscale skeletonization solutions and challenges. Section 2.3 introduces the Image Foresting Transform and its adaptations required for multiscale skeletonization. Section 2.3.3 details our multiscale skeletonization algorithms for 2D and 3D shapes. Section 2.4 compares our method with its multiscale competitors on a wide set of 2D and 3D real-world shapes. Section 2.5 concludes the chapter, summarizing our main contributions and outlining directions for future work.

2.2 RELATED WORK

In this section, we provide the basic definitions related to skeletonization and discuss skeleton regularization based on local and global measures.

2.2.1 DEFINITIONS

Given a shape $\Omega \subset \mathbb{R}^d$, $d \in \{2, 3\}$ with boundary $\partial\Omega$, we first define its Euclidean distance transform $\mathcal{D} : \mathbb{R}^d \to \mathbb{R}^+$ as

$$\mathcal{D}(\mathbf{x} \in \Omega) = \min_{\mathbf{y} \in \partial\Omega} \|\mathbf{x} - \mathbf{y}\|. \tag{2.1}$$

The Euclidean skeleton of Ω is next defined as

$$\mathcal{S} = \{\mathbf{x} \in \Omega | \exists \mathbf{f}_1, \mathbf{f}_2 \in \partial\Omega, \mathbf{f}_1 \neq \mathbf{f}_2, \|\mathbf{x} - \mathbf{f}_1\| = \|\mathbf{x} - \mathbf{f}_2\| = \mathcal{D}(\mathbf{x})\}, \tag{2.2}$$

where $\mathbf{f}_1$ and $\mathbf{f}_2$ are the contact points with $\partial\Omega$ of the maximally inscribed ball in Ω centered at $\mathbf{x}$ [23,39], also called *feature points* of $\mathbf{x}$ [25], where the feature transform $\mathcal{F} : \mathbb{R}^d \to \mathcal{P}(\partial\Omega)$ is defined as

$$\mathcal{F}(\mathbf{x} \in \Omega) = \arg\min_{\mathbf{y} \in \partial\Omega} \|\mathbf{x} - \mathbf{y}\|. \tag{2.3}$$

The vectors $\mathbf{f} - \mathbf{x}$ are called *spoke vectors* [47]. By definition (Eq. (2.2)), for any $\mathbf{x} \notin \mathcal{S}$, $\mathcal{F}(\mathbf{x})$ yields a single point, i.e., $|\mathcal{F}(\mathbf{x})| = 1$, whereas for any $\mathbf{x} \in \mathcal{S}$, $|\mathcal{F}(\mathbf{x})| \geq 2$. In practice, computing $\mathcal{F}$ can be quite expensive and/or complicated due to its multivalued nature. As such, many applications (see, e.g., [25,21,39]) use the so-called single-value feature transform $F : \mathbb{R}^d \to \partial\Omega$, defined as

$$F(\mathbf{x} \in \Omega) = \mathbf{y} \in \partial\Omega \quad \text{so that} \quad \|\mathbf{x} - \mathbf{y}\| = \mathcal{D}(\mathbf{x}). \tag{2.4}$$

For $d = 2$, $\mathcal{S}$ is a set of curves that meet at the so-called skeleton junction points [17]. For $d = 3$, $\mathcal{S}$ is a set of manifolds with boundaries that meet along a set of so-called Y-intersection curves [12,30,6]. The pair $MAT = (\mathcal{S}, \mathcal{D})$ defines the medial axis transform (MAT) of Ω, which is a dual representation of Ω, i.e., allows reconstructing the shape $\Omega_{\mathcal{S},\mathcal{D}} = \bigcup_{\mathbf{x} \in \mathcal{S}} B_{\mathcal{D}(\mathbf{x})}(\mathbf{x})$ as the union of balls $B_{\mathcal{D}(\mathbf{x})}(\mathbf{x})$ centered at $\mathbf{x} \in \mathcal{S}$ and with radii $\mathcal{D}(\mathbf{x})$.

Except when it is explicitly mentioned, we consider only the case where $\Omega \setminus \partial\Omega$ and $\partial\Omega$ have the same number of components.

2.2.2 SKELETON REGULARIZATION

In practice, skeletons are extracted from discretized (sampled) versions of Ω, using either an implicit (boundary mesh) representation [35,51,32,26] or an explicit (volumetric) representation [17,53,14,39]. In this chapter, we focus on the latter case. Here, the d-dimensional space is discretized in a uniform grid of so-called *spels* (space elements) having integer coordinates, i.e., pixels for $d = 2$ and voxels for $d = 3$. Due to this discretization, solving Eq. (2.2) on $\mathbb{Z}^d$ rather than on $\mathbb{R}^d$ yields skeletons that are not perfectly centered, not necessarily one-spel thin, and not necessarily homotopic to the input shape. Discretization also causes skeletons to have a large amount of spurious manifolds (branches). Formally, this means that skeletonization is not a Cauchy

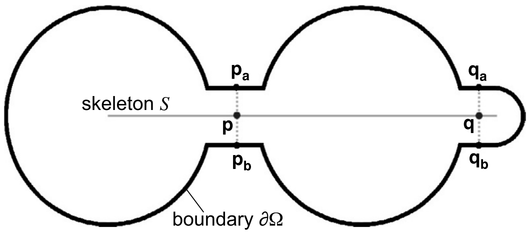

FIGURE 2.1

Problems of local regularization. For the shown shape, all local regularization metrics yield the same values for points $\mathbf{p}$ and $\mathbf{q}$. However, $\mathbf{p}$ is globally more important for the shape description than $\mathbf{q}$.

or Lipschitz continuous operation with respect to the Hausdorff distance between two shapes, but a semicontinuous operation [50]. Informally put, small variations of a shape can cause arbitrarily large variations of its skeleton.

To achieve Cauchy or Lipschitz continuity, desirable for most practical applications (which should not be sensitive to discretization issues), a *regularization* process is typically used. For this, we define the so-called *importance metric* $\rho : \Omega \to \mathbb{R}^+$, whose upper thresholding by some desired value $\tau > 0$ removes, or prunes, branches caused by small-scale details or noise on $\partial\Omega$ [43,12]. The regularized skeleton is defined as $\mathcal{S}_\tau = \{\mathbf{x} \in \mathcal{S} | \rho(\mathbf{x}) \geq \tau\}$. We distinguish between local and global importance measures, in line with [39,32,26,27], as follows.

Local measures essentially consider, for a skeletal point $\mathbf{x} \in \mathcal{S}$, only a small neighborhood of $\mathbf{x}$ to compute $\rho(\mathbf{x})$. The main advantage of these measures is that they are simple to implement and fast to compute. Local measures include the angle between the feature points and distance-to-boundary [1,22,49,25], divergence metrics [44,5], first-order moments [42], and detecting the multivalued points of ∇D [47,48]. Local measures are also, historically speaking, the first proposed skeleton regularization techniques. Good surveys of local methods are given in [45,50]. However, local measures have a fundamental issue: They cannot discriminate between locally identical, yet globally different, shape contexts. Fig. 2.1 illustrates this for a synthetic case: For the 2D shape with boundary $\partial\Omega$, the central skeletal point $\mathbf{p}$ is clearly more important to the shape description than the peripheral point $\mathbf{q}$ that corresponds to the right local protrusion. However, any local importance metric will rank $\mathbf{p}$ as important as $\mathbf{q}$ since their surroundings, including the placement of their feature points (shown in the figure), are identical. Similar situations can be easily found for 3D shapes.

Given the above, thresholding local measures can (and typically will) disconnect skeletons. Reconnection needs extra work [44,37,31,49] and makes skeleton pruning less intuitive and harder to implement [43]. Without this kind of work, no local measure can yield connected skeletons for all shapes. Note that this is a fundamental issue related to the local nature of these metrics; see also the discussion in [27]. Secondly, local metrics do not support the notion of a *multiscale* skeleton: Such skeletons $\mathcal{S}_\tau$ should ensure a continuous simplification (in the Cauchy or Lipschitz sense mentioned before) of the input shape Ω in terms of its reconstruction $\Omega_{\mathcal{S}_\tau,\mathcal{D}}$ as a function of the simplification parameter τ [35,50]. As such, although local measures can be "fixed" by reconnection work to yield connected skeletons, they still cannot provide an intuitive and easy-to-use way to simplify skeletons according to a user-prescribed threshold τ.

Global measures monotonically increase as one walks along $\mathcal{S}$ from the skeleton boundary $\partial\mathcal{S}$ inwards, toward points increasingly further away from $\partial\mathcal{S}$. For genus-0 shapes, such measures can be informally thought of as giving the removal order of skeletal points in a homotopy-preserving erosion process that starts at $\partial\mathcal{S}$ and ends when the entire skeleton has been eroded away. Given this property, thresholding them always yields connected skeletons, which also capture shape details at a user-given scale. For shapes with genus greater than 0, like having holes (in 2D) or cavities (in 3D), simple suitable postprocessing of the importance metric guarantees the joint connectivity and multiscale properties. For 2D shapes, a well-known global measure is the so-called boundary-collapse metric used to extract multiscale 2D skeletons, and proposed by various authors in different contexts [35,17,53,52]. For 3D shapes, the union-of-balls (UoB) approximation uses morphological dilation and erosion to define the scale of shape details captured by the regularized skeleton [24,32]. Dey and Sun propose as a regularization metric the medial geodesic function (MGF), equal to the length of the shortest-path between feature points [14,38] and use this metric to compute regularized 3D curve skeletons. Reniers et al. [39] extend the MGF for both surface and curve skeletons using geodesic lengths and surface areas between geodesics, respectively. A fast GPU implementation of this extended MGF for meshed shapes is given in [26].

The 3D MGF and its 2D boundary-collapse metric counterpart have an intuitive geometric meaning: They assign to a skeleton point $\mathbf{x} \in \mathcal{S}$ the amount of shape boundary that corresponds, or "collapses" to, $\mathbf{x}$ by some kind of advective boundary-to-skeleton transport. As such, skeleton points $\mathbf{x}$ with low importance values correspond to small-scale shape details or noise; points $\mathbf{x}$ with large importance values correspond to large-scale shape parts. Fig. 2.2 illustrates the boundary-collapse principle for both 2D medial axes (A) and 3D medial surfaces (B). In both cases, for a skeletal point $\mathbf{x}$, the importance is equal to the length of the shortest path $\gamma_\mathbf{x}$ that goes on $\partial\Omega$ between the feature points $\mathbf{f}_1^\mathbf{x}$ and $\mathbf{f}_2^\mathbf{x}$. This allows an intuitive and controllable skeleton simplification: Thresholding the MGF by a value τ eliminates all skeleton points that encode less than τ boundary length or area units. Since all the above-mentioned collapse metrics monotonically increase from the skeleton boundary $\partial\mathcal{S}$ to its center, thresholding them delivers a set of nested skeleton approximations, also called a multiscale skeleton. Importantly, these progressively simplified skeletons correspond, via the MAT (Section 2.2.1), to progressively simplified versions of Ω. More precisely, for all above collapse metrics, the reconstruction of a shape Ω from its simplified skeleton $\mathcal{S}_\tau$ yields a shape where all details of Ω of size smaller than τ have been replaced by circle arcs (in 2D) or spherical caps (in 3D) [53,39,26].

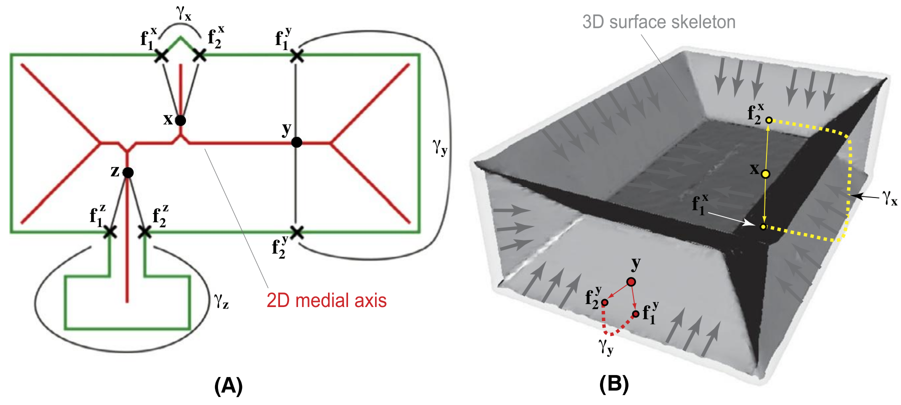

FIGURE 2.2

Multiscale collapse metric for 2D shapes (A) and 3D shapes (B).

The idea of a mass collapse process from $\partial\Omega$ to $\mathcal{S}$ was also used by other works to define multiscale skeletons. Couprie [9] proposed a discrete framework for computing 2D skeletons and 3D curve skeletons by a guided thinning process, which, for the 2D case, yields very similar results to [53,17]. However, 3D surface skeletons are not covered by this approach. Torsello et al. propose a conservative mass advection $\partial\Omega$ onto $\mathcal{S}$ to define ρ in 2D [4], which was next extended to 3D [41]. However, the numerical computation of this process is affected by serious stability issues. Recently, Jalba et al. extended this advection model to compute multiscale 2D skeletons and 3D surface and curve skeletons in a unified formulation [27]. Although this method delivers convincing results, it still suffers from numerical stability problems and is also relatively complex to implement.

Summarizing the above, we argue that multiscale regularization metrics are net superior, both in theory and practice, to local regularization metrics. However, as outlined and discussed next in more detail in Section 2.3.1, multiscale regularization is far from being simple and cheap. Our proposal, presented next, aims at solving these problems.

2.3 PROPOSED METHOD

To compute multiscale 2D and 3D skeletons of binary shapes efficiently and robustly, we propose to use the *Image Foresting Transform* (IFT) methodology [19]. We start, in Section 2.3.1, by introducing our general idea, which details the strengths and weaknesses of the MGF and advection-based regularization techniques introduced in Section 2.2.2. Next, we detail the use of the IFT to compute skeletons that combine the identified strengths of these two approaches (Section 2.3.2). Our final proposed skeletonization algorithms are detailed in Section 2.3.3.

2.3.1 MULTISCALE REGULARIZATION—STRENGTHS AND WEAKNESSES

Analyzing all multiscale skeletonization methods surveyed in Section 2.2 [35,17,53,52,39,4,41,26,27], we notice that their various importance-metric definitions can be all explained, at a high level, by introducing a vector field $\mathbf{v} : \Omega \to \mathbb{R}^d$ as follows: If we imagine that the input shape surface $\partial\Omega$ is covered by uniformly distributed mass with density $\rho(\mathbf{x} \in \partial\Omega) = 1$, then all above methods explain the importance $\rho(\mathbf{x} \in \Omega)$ of a spel $\mathbf{x}$ as the amount of mass transported, or advected, by $\mathbf{v}$ from $\partial\Omega$ to $\mathbf{x}$. Studying the properties of $\mathbf{v}$ brings several insights into multiscale regularization as follows.

First, we note that the importance values of nonskeletal spels $\mathbf{x} \notin \mathcal{S}$ should be low and nearly locally constant, so that upper-thresholding ρ by this value allows us to reliably separate $\mathcal{S}$. All above-mentioned methods define $\rho(\mathbf{x} \notin \mathcal{S})$ to be equal to the importance of the single feature point of $\mathbf{x}$, i.e., $\rho(\mathbf{x} \notin \mathcal{S}) = \rho(\mathcal{F}(\mathbf{x}))$. This property is realized if we define $\mathbf{v} = \nabla\mathcal{D}$ for all $\mathbf{x} \notin \mathcal{S}$, as it is well known that gradient lines of the distance transform only intersect at skeletal points [44,5].

To fully define the multiscale importance ρ over Ω in terms of an advection process, it thus remains to define $\mathbf{v}$ on $\mathcal{S}$. Studying the above-mentioned multiscale methods and considering for now the case of connected genus-0 shapes, we see here that all such methods aim to define a field ρ that monotonically increases from $\partial\mathcal{S}$ to its center, or root $\mathbf{r} \in \mathcal{S}$. As explained earlier, this allows easy computing of multiscale skeletons $\mathcal{S}_\tau$ by simply upper-thresholding ρ with desired values τ. In advection terms, this is equivalent to defining a field $\mathbf{v}$ that transports mass along $\mathcal{S}$ from its boundary $\partial\mathcal{S}$ to $\mathbf{r}$, along paths that finally meet at $\mathbf{r}$. For $d = 2$, mass flows from $\partial\Omega$ to the one-dimensional medial axis $\mathcal{S}$ and then along the branches of $\mathcal{S}$ to its center $\mathbf{r}$. For $d = 3$, mass flows from $\partial\Omega$ to the two-dimensional surface-skeleton $\mathcal{S}$, then along $\mathcal{S}$ toward its local center (which is the curve skeleton of Ω), and then along the curve-skeleton branches toward the center $\mathbf{r}$ thereof.

The different multiscale importance listed above can be explained in terms of different definitions of $\mathbf{v}$ over $\mathcal{S}$ as follows. In 2D, for genus-0 shapes, all surveyed methods essentially reduce to defining $\mathbf{v}$ as being locally tangent to $\mathcal{S}$ and pointing toward the root of the skeleton (which in this case is a tree) [35,17,53,52]. In 3D, explaining multiscale importance in terms of a field $\mathbf{v}$ defined on $\mathcal{S}$ is more complicated. We distinguish here two classes of methods as follows.

MGF methods: For genus-0 shapes, MGF-based methods do not actually give a formal definition of $\mathbf{v}$, but compute $\rho(\mathbf{x} \in \mathcal{S})$ as the length of the longest shortest-path on $\partial\Omega$ between any two feature points $\mathbf{f}$ and $\mathbf{g}$ of $\mathbf{x}$, i.e.,

$$\rho(\mathbf{x} \in \mathcal{S}) = \max_{\mathbf{f}\in\mathcal{F}(\mathbf{x}),\mathbf{g}\in\mathcal{F}(\mathbf{x})} GL(\mathbf{f}, \mathbf{g}), \tag{2.5}$$

where $GL(\mathbf{x}, \mathbf{y})$ is the length of the shortest path on $\partial\Omega$ between two points $\mathbf{x} \in \partial\Omega$ and $\mathbf{y} \in \partial\Omega$. This is based on the empirical observation that Eq. (2.5) defines a field ρ that smoothly and monotonically increases from $\partial\mathcal{S}$ to its center [14,39]. This allows us to compute regularized surface skeletons even for noisy and complex 3D shapes. Another advantage of the MGF is that it makes simplification intuitive to understand: thresholding ρ at a value τ eliminates all spels from $\mathcal{S}$ where the local thickness of the shape is larger than τ. However, a formal justification of the MGF in terms of an advection process on $\mathcal{S}$ has not yet been given. A second limitation of the MGF model is that is becomes very expensive to compute for large 3D shapes, where we need to trace (at least one) shortest path on $\partial\Omega$ for each point $\mathbf{x} \in \mathcal{S}$, and so its cost is $O(|\mathcal{S}| \cdot |\partial\Omega| \log |\partial\Omega|)$. Using GPU techniques can accelerate this process [26], but also massively complicates the implementation of the method.

Advection methods: Aiming to solve the above issues with the MGF, Jalba et al. define $\mathbf{v}$ on $\mathcal{S}$ as the result of a mass-conserving advection model, with topological constraints used to ensure skeleton homotopy with the input shape [27]. In contrast to the MGF, computation of all types of skeletons (2D medial axes, 3D surface skeletons, and 3D curve skeletons) is now fully captured by a single unified advection model. Also, the method in [27] is considerably faster than MGF techniques, its cost being $O(|\Omega|)$, and is also simpler to implement. However, setting the simplification threshold τ is now less intuitive than for MGF techniques since this value does not have an immediate geometric interpretation. Also, all 3D advection methods we are aware of [4,41,27] suffer various amounts of from numerical diffusion, which means that the computed importance ρ will deliver regularized skeletons $\mathcal{S}_\tau$ having jagged boundaries.

The method that we propose in the next section aims to combine the strengths and limit the disadvantages of the MGF and advection methods outlined above. In detail, we use an incremental advection-like computation of the intuitive MGF metric, which makes our method considerably faster than existing MGF methods. We provide an implementation that is simple and does not suffer from numerical diffusion issues. Our proposal delivers smooth-boundary regularized, centered, one-spel-thin, multiscale skeletons of the same overall quality as skeletons delivered by existing state-of-the-art methods.

2.3.2 IMAGE FORESTING TRANSFORM

The Image Foresting Transform (IFT) interprets d-dimensional images as graphs and reduces image operators to the computation of an *optimum-path forest* followed by a local processing of its attributes. In essence, the IFT is Dijkstra's shortest-path algorithm modified to use multiple sources and more general connectivity (path-value) functions. In our context, the IFT propagates, from $\partial\Omega$ to $\mathbf{x} \in \Omega \setminus \partial\Omega$, both the feature points $\mathcal{F}(\mathbf{x})$ and the advection field $\mathbf{v}(\mathbf{x})$. Fig. 2.3 shows in yellow the spels s with undefined $\nabla\mathcal{D}(\mathbf{x})$. These are the leaves of the optimum-path forest whose paths follow the direction of $\mathbf{v}$.

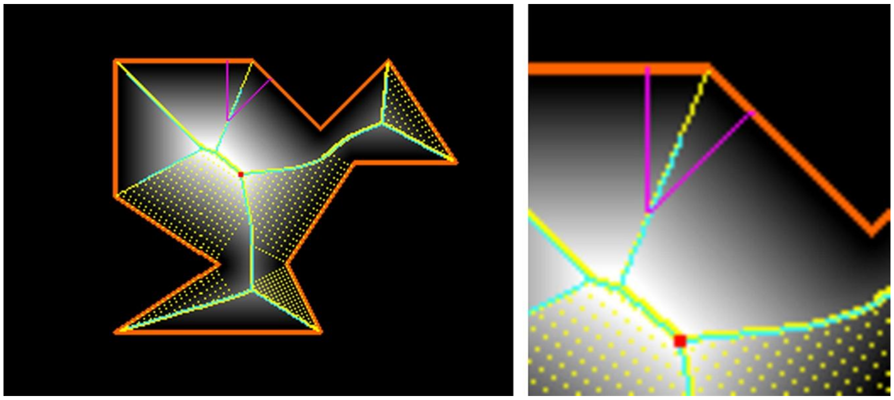

FIGURE 2.3

A polygon Ω with orange boundary $\partial\Omega$, its distance transform D (gray values), a plausible skeleton $\mathcal{S}$ (cyan), the root point r (red), the points with undefined $\nabla D(s)$ (yellow), and magenta lines connecting a given spel s with its feature points in $\partial\Omega$.

To design an image operator based on the IFT, we need to specify which image elements (points, edges, regions) are the nodes of the graph, an *adjacency relation* between them, and a *connectivity function* that assigns a value (e.g., strength, cost, distance) to any path in the graph. This methodology has been successfully used for boundary-based [20,18,33], region-based [16,13], and hybrid image segmentation [46,7]; connected filtering [15]; shape representation and description [17,11,10, 3]; and unsupervised [40], semisupervised [2], and supervised data classification [36]. In this section, we show how the IFT can be adapted to propagate the single-point feature transform $F(\mathbf{x}) \in \partial\Omega, \forall \mathbf{x} \in \Omega \setminus \partial\Omega$ (Eq. (2.4)) and also to compute the length $GL(\mathbf{x}, \mathbf{y})$ of the shortest path on $\partial\Omega$ between two feature points $\mathbf{x}, \mathbf{y} \in \partial\Omega$.

Graph definition: In the discrete space, the shape Ω is provided as a binary image $\mathbf{I} = (\mathcal{I}, I)$, where $\mathcal{I} \subset \mathbb{Z}^d$ is the image domain, and each spel $\mathbf{x} \in \mathcal{I}$ has a value $I(\mathbf{x}) \in \{0, 1\}$. For instance, $\Omega \subset \mathcal{I}$ may be the set of spels with value 1 and its complement $\overline{\Omega} = \mathcal{I} \setminus \Omega$ be defined by the spels with value 0. The image $\mathbf{I}$ can be interpreted as a graph whose nodes are the spels in $\mathcal{I}$ and arcs are defined by the adjacency relation

$$\mathcal{A}_{\mathcal{I},\delta} \quad = \quad \{(\mathbf{x}, \mathbf{y}) \in \mathcal{I} \times \mathcal{I} \ \mid \ \|\mathbf{x} - \mathbf{y}\| \leq \delta\} \tag{2.6}$$

for a given value $\delta \in \mathbb{R}^+$. Let also $\mathcal{A}_{\mathcal{I},\delta}(\mathbf{x})$ be the set of spels adjacent to a spel $\mathbf{x}$. The shape boundary $\partial\Omega$ is then defined by

$$\partial\Omega \quad = \quad \{\mathbf{x} \in \Omega | \overline{\Omega} \cap \mathcal{A}_{\mathcal{I},1}(\mathbf{x}) \neq \emptyset\}. \tag{2.7}$$

To compute the single-point feature transform F and the length $GL(\mathbf{f}, \mathbf{g})$ of shortest paths between point-pairs $(\mathbf{f}, \mathbf{g}) \in \partial\Omega \times \partial\Omega$, we constrain the adjacency relation $\mathcal{A}_{\mathcal{I},\delta}$ to it subsets $\mathcal{A}_{\Omega,\delta}$ and $\mathcal{A}_{\partial\Omega,\delta}$ defined as

$$\begin{aligned}\mathcal{A}_{\Omega,\delta} &= \{(\mathbf{x}, \mathbf{y}) \in \Omega \times \Omega \mid \|\mathbf{x} - \mathbf{y}\| \leq \delta\}, & (2.8)\\ \mathcal{A}_{\partial\Omega,\delta} &= \{(\mathbf{x}, \mathbf{y}) \in \partial\Omega \times \partial\Omega \mid \|\mathbf{x} - \mathbf{y}\| \leq \delta\}, & (2.9)\end{aligned}$$

respectively. Given the above, we are next interested in the graphs $(\Omega, \mathcal{A}_{\Omega,\delta})$ and $(\partial\Omega, \mathcal{A}_{\partial\Omega,\delta})$ that describe the shape's interior and boundary, respectively.

For the graph $(\Omega, \mathcal{A}_{\Omega,\delta})$, let a *path* $\pi_{\mathbf{t}}$ be a spel-sequence $\langle \mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n = \mathbf{t}\rangle$ with terminal spel $\mathbf{t}$ such that $(\mathbf{x}_i, \mathbf{x}_{i+1}) \in \mathcal{A}_{\Omega,\delta}$, $1 \leq i < n$. Let $\Pi\left(\Omega, \mathcal{A}_{\Omega,\delta}\right)$ be the set of all paths in $(\Omega, \mathcal{A}_{\Omega,\delta})$. A pair of spels is called *connected* in $(\Omega, \mathcal{A}_{\Omega,\delta})$ if there exists a path in $\Pi\left(\Omega, \mathcal{A}_{\Omega,\delta}\right)$ between them. A *connected component* in $(\Omega, \mathcal{A}_{\Omega,\delta})$ is a subgraph, maximal for the inclusion, therein all pairs of spels are connected. The same definitions apply to the graph $(\partial\Omega, \mathcal{A}_{\partial\Omega,\delta})$.

We next assume, with no generality loss, that the interior $\Omega \setminus \partial\Omega$ of Ω defines a single connected component in $(\Omega, \mathcal{A}_{\Omega,1})$. When this is not the case, we simply treat each such connected component separately to yield a separate skeleton. To eliminate cases where Ω and $\Omega \setminus \partial\Omega$ have different numbers of components, i.e., the removal of $\partial\Omega$ would disconnect the shape, we can preprocess Ω by, e.g., morphological dilation with the distance of the size of one spel.

Objects with holes (in 2D) can be easily treated by merging the internal skeletons derived from each component in $(\partial\Omega, \mathcal{A}_{\partial\Omega,\sqrt{d}})$ into a single one through the *skeleton by influence zones* (SKIZ) [17]. We will illustrate that for the 2D case, but since our examples of 3D objects do not present cavities, we assume for simplicity (in 3D) that $\partial\Omega$ has a single connected component in $(\partial\Omega, \mathcal{A}_{\partial\Omega,\sqrt{d}})$. Note also that the algorithm we next propose to compute the so-called interior skeleton of Ω can be also used to compute the so-called external skeleton of the complement $\overline{\Omega}$. For presentation simplicity, we focus here on the interior skeleton.

Distance and feature transforms: Using the graph $(\Omega, \mathcal{A}_{\Omega,\sqrt{d}})$, we can propagate from $\partial\Omega$ the distance value $\mathcal{D}(\mathbf{x})$ to every interior spel $\mathbf{x} \in \Omega \setminus \partial\Omega$ by using the connectivity function

$$\begin{aligned}\psi_{edt}(\langle \mathbf{t}\rangle) &= \begin{cases} 0 & \text{if } \mathbf{t} \in \partial\Omega\text{, and}\\ +\infty & \text{otherwise,}\end{cases}\\ \psi_{edt}(\pi_{\mathbf{s}} \cdot \langle \mathbf{s}, \mathbf{t}\rangle) &= \|\mathbf{s} - F(\mathbf{s})\|, & (2.10)\end{aligned}$$

where $F(\mathbf{s}) \in \partial\Omega$ is the single-point feature transform of $\mathbf{s}$ (Eq. (2.4)), and $\pi_{\mathbf{s}} \cdot \langle \mathbf{s}, \mathbf{t}\rangle$ is the extension of the path $\pi_{\mathbf{s}}$ by the arc $(\mathbf{s}, \mathbf{t}) \in \mathcal{A}_{\Omega,\sqrt{d}}$. The Euclidean distance transform $\mathcal{D}$ thus becomes

$$\mathcal{D}(\mathbf{t} \in \Omega) = \min_{\forall \pi_{\mathbf{t}} \in \Pi(\Omega, \mathcal{A}_{\Omega,\sqrt{d}})} \{\psi_{edt}(\pi_{\mathbf{t}})\}. \quad (2.11)$$

Besides propagating the distance transform $\mathcal{D}$, the IFT method also propagates the closest point $F(\mathbf{x}) \in \partial\Omega$ to any $\mathbf{x} \in \Omega$. This yields the single-point feature transform F for all spels in Ω, which we will heavily use, as outlined next.

MGF importance: To compute ρ (Eq. (2.5)), we need the complete feature transform $\mathcal{F}$. As explained in Section 2.2.1, computing $\mathcal{F}$ is difficult, especially for discrete-grid representations. In practice, this is often replaced by computing the so-called extended single-point feature transform $\mathcal{F}_{ext}$ [39,21] defined as

$$\mathcal{F}_{ext}(\mathbf{s}) \quad = \quad \{F(\mathbf{t}) \in \partial\Omega \;\mid\; \mathbf{t} \in \mathcal{A}_{\Omega,1}(\mathbf{s})\}, \tag{2.12}$$

which gathers the single-point feature transforms $F(\mathbf{t})$ of all adjacent spels $\mathbf{t} \in \mathcal{A}_{\Omega,1}(\mathbf{s})$. Having $\mathcal{F}_{ext}$, we can now immediately write a simpler version of Eq. (2.5) as

$$\rho(\mathbf{s} \in \Omega \setminus \partial\Omega) \quad = \quad \max_{\mathbf{f}=F(\mathbf{s}),\mathbf{g}\in\mathcal{F}_{ext}(\mathbf{s})} \{GL(\mathbf{f},\mathbf{g})\}. \tag{2.13}$$

This simplification applies to multiscale planar and surface skeletons, leading to less shortest-path length computations, but Eq. (2.5) is still important if one desires to merge 3D multiscale curve and surface skeletons because the union of geodesic paths between all pairs of feature points of a spel $\mathbf{s}$ on the curve skeleton draws in $\partial\Omega$ a closed contour, splitting $\partial\Omega$ into two parts such that the geodesic surface area between them can be used as importance $\rho(\mathbf{s})$ [39].

To evaluate Eq. (2.13), we compute the shortest-path length GL from $\mathbf{f} = F(\mathbf{s})$ to $\mathbf{g} \in \mathcal{F}_{ext}(\mathbf{s})$ on the boundary-graph $(\partial\Omega, \mathcal{A}_{\partial\Omega,\sqrt{d}})$ by using the connectivity function ψ_{geo} defined as

$$\begin{aligned}
\psi_{geo}(\langle\mathbf{w}\rangle) \quad &= \quad \begin{cases} 0 & \text{if } \mathbf{w} = \mathbf{f}, \text{ and} \\ +\infty & \text{otherwise,} \end{cases} \\
\psi_{geo}(\pi_{\mathbf{w}} \cdot \langle\mathbf{w},\mathbf{h}\rangle) \quad &= \quad \psi_{geo}(\pi_{\mathbf{w}}) + \|\mathbf{h}-\mathbf{w}\| + \|\mathbf{g}-\mathbf{h}\|,
\end{aligned} \tag{2.14}$$

where the term $\|\mathbf{g}-\mathbf{h}\|$ is the A^* heuristic optimization [34] used to reach $\mathbf{g}$ faster. Summarizing, we compute the shortest-path length $GL(\mathbf{f},\mathbf{g})$ as

$$GL(\mathbf{f},\mathbf{g}) \quad = \quad \min_{\pi_{\mathbf{g}}\in\Pi(\Omega,\mathcal{A}_{\Omega,\sqrt{d}})} \{\psi_{geo}(\pi_{\mathbf{g}})\}. \tag{2.15}$$

It is important to note that the IFT propagation for ψ_{edt} can also output an *optimum-path forest* P, i.e., a map that assigns a so-called predecessor $\mathbf{s} = P(\mathbf{t}) \in \Omega$ to every spel $\mathbf{t} \in \Omega \setminus \partial\Omega$, and a marker $P(\mathbf{t}) = nil \notin \Omega$ to spels $\mathbf{t} \in \partial\Omega$, respectively [19]. This defines a vector field $\mathbf{v}(\mathbf{t}) = \mathbf{t} - P(\mathbf{t})$ for every interior spel $\mathbf{t} \in \Omega \setminus \partial\Omega$. The forest P provides the direction of $\mathbf{v}$ for nonskeletal spels. The skeleton $\mathcal{S}$ is contained in the set of P's leaves (yellow lines in Fig. 2.3). The vector field $\mathbf{v}$ describes how all information computed by the IFT—that is, $\mathcal{D}$, F, and ρ—is iteratively propagated, or *advected*, from $\partial\Omega$ to all spels in the shape's interior $\Omega \setminus \partial\Omega$. This fundamentally links the MGF importance model and the advection importance model. As explained in Sections 2.2 and 2.3.1, these two importance models are typically used independently in the literature. The only work that we are aware of where an MGF model is linked with an advection model is [9]. However, our work here stands apart from [9] in terms of algorithmic model, and also by the fact that, for the 3D case, we compute multiscale *surface* skeletons (and not curve skeletons, whereas [9] approaches precisely the opposite).

Summarizing the above: To compute ρ (Eq. (2.13)), we need to compute the single-point distance transform F and the shortest-path length between feature points in the extended feature transform $\mathcal{F}_{ext}$. The algorithms for both these operations are described in Sections 2.3.2.1 and 2.3.3, respectively.

2.3.2.1 Single-Point Feature Transform

The single-point feature transform F is computed by the same algorithm used for computing the Euclidean distance transform $\mathcal{D}$, but returns F rather than $\mathcal{D}$. The full algorithm we use for computing F is listed below. It also returns $\partial\Omega$, which we next need to compute shortest-paths between feature points, and a component label map $L_c\colon \mathbf{s} \in \Omega \to \lambda(\mathbf{s}) \in \{1, 2, \ldots, c\}$ that assigns a subsequent integer number to each component of $\partial\Omega$ in $(\partial\Omega, \mathcal{A}_{\partial\Omega,\sqrt{d}})$ and its closest spels in Ω. The map L_c is used for SKIZ computation in 2D. Indeed, the component label propagation to every spel $\mathbf{s} \in \Omega \setminus \partial\Omega$ is not needed, but it helps to illustrate the location of the SKIZ (Section 2.3.3).

Algorithm 1 (SINGLE-POINT FEATURE TRANSFORM)**.**

INPUT:	An object Ω in dimension d represented on a uniform $\mathbb{Z}^d$ grid.
OUTPUT:	The single-point feature transform F, object boundary $\partial\Omega$, and component label map L_c.
AUXILIARY:	Priority queue Q, distance transform $\mathcal{D}$, and variable $tmp \in \mathbb{R}$.

```
1.  Compute $\partial\Omega$ of $\Omega$ by Eq. (2.7).
2.  For each $\mathbf{s} \in \Omega \setminus \partial\Omega$, $\mathcal{D}(\mathbf{s}) \leftarrow +\infty$.
3.  For each $\mathbf{s} \in \partial\Omega$, do
4.      $\mathcal{D}(\mathbf{s}) \leftarrow 0$; $F(\mathbf{s}) \leftarrow \mathbf{s}$; $L_c(\mathbf{s}) \leftarrow \lambda(\mathbf{s}) \in \{1, 2, \ldots, c\}$, according to
5.      its component in $(\partial\Omega, \mathcal{A}_{\partial\Omega,\sqrt{d}})$; and insert $\mathbf{s}$ in $Q$.
6.  While $Q \neq \emptyset$, do
7.      Remove $\mathbf{s}$ from $Q$, where $\mathcal{D}(\mathbf{s})$ is minimal over $Q$.
8.      For each $\mathbf{t} \in \mathcal{A}_{\Omega,\sqrt{d}}(\mathbf{s})$ such that $\mathcal{D}(\mathbf{t}) > \mathcal{D}(\mathbf{s})$, do
9.          $tmp \leftarrow \|\mathbf{t} - F(\mathbf{s})\|^2$.
10.         If $tmp < \mathcal{D}(\mathbf{t})$, then
11.             $\mathcal{D}(\mathbf{t}) \leftarrow tmp$; $F(\mathbf{t}) \leftarrow F(\mathbf{s})$; $L_c(\mathbf{t}) \leftarrow L_c(\mathbf{s})$.
12.             If $\mathcal{D}(\mathbf{t}) \neq +\infty$, then
13.                 Update position of $\mathbf{t}$ in $Q$.
14.             Else
15.                 Insert $\mathbf{t}$ in $Q$.
16. Return $(F, \partial\Omega, L_c)$
```

Lines 1–5 essentially extract the object boundary $\partial\Omega$, initialize the trivial-path values of Eq. (2.10) in $\mathcal{D}(\mathbf{s})$ for all $\mathbf{s} \in \Omega$, set $F(\mathbf{s})$ for $\mathbf{s} \in \partial\Omega$, assign a distinct integer to each component of $\partial\Omega$ in $(\partial\Omega, \mathcal{A}_{\partial\Omega,\sqrt{d}})$, and insert $\mathbf{s} \in \partial\Omega$ in the priority queue Q. The main loop in Lines 6–15 propagates to every spel $\mathbf{t} \in \Omega \setminus \partial\Omega$ its single-point feature $F(\mathbf{t}) \in \partial\Omega$ in a nondecreasing order of distance values $\mathcal{D}(\mathbf{t})$ between $\mathbf{t}$ and $\partial\Omega$. In Line 7, when a spel $\mathbf{s}$ is removed from Q, $\mathcal{D}(\mathbf{s})$ stores the closest squared distance between $\mathbf{s}$ and $\partial\Omega$, $F(\mathbf{s})$ stores its single-point feature, and $L_c(\mathbf{s})$ indicates its closest component in $(\partial\Omega, \mathcal{A}_{\partial\Omega,\sqrt{d}})$. The loop in Lines 8–15 evaluates if $\mathbf{s}$ can offer a lower squared distance value $\|\mathbf{t} - F(\mathbf{s})\|^2$ (value of an extended path $\pi_{\mathbf{s}} \cdot \langle \mathbf{s}, \mathbf{t}\rangle$ in Eq. (2.10)) to the current value assigned to an adjacent spel $\mathbf{t}$ in $\mathcal{D}(\mathbf{t})$ (Lines 9–10). If this is the case, then Line 11 updates distance, single-point feature of $\mathbf{t}$ with respect to $\partial\Omega$, its closest component in $(\partial\Omega, \mathcal{A}_{\partial\Omega,\sqrt{d}})$, and Lines 12–15 update the status of $\mathbf{t}$ in Q.

Note that the use of the squared Euclidean distance $\|\mathbf{t} - F(\mathbf{s})\|^2$ in Line 9 allows us to implement Q by bucket sorting [20] since all distances are integers on a pixel/voxel grid representation. As such, Algorithm 1 has average complexity $O(|\Omega|)$.

2.3.2.2 *Shortest-Path Length Computation*

As mentioned earlier in Section 2.3.1, computing the multiscale regularization metric ρ for $d = 3$ heavily depends, cost-wise, on the rapid computation of shortest-path lengths on $\partial\Omega$ between single-point features. Accelerating these shortest-path computations is key to accelerating multiscale 3D skeletonization. To achieve this, we maintain, for each $\mathbf{f} \in \partial\Omega$, a set $\mathcal{C}(\mathbf{f}) = \{\mathbf{s} \in \Omega \setminus \partial\Omega | F(\mathbf{s}) = \mathbf{f}\}$. We use $\mathcal{C}$ to *incrementally* compute all shortest-path lengths between $\mathbf{f}$ and other single-point features $\mathbf{g} \neq \mathbf{f}, \mathbf{g} \in \partial\Omega$ as follows: The IFT algorithm returns $GL(\mathbf{f}, \mathbf{g})$ whenever $\mathbf{g}$ is reached; for any $\mathbf{h} \in \partial\Omega$ for which $GL(\mathbf{f}, \mathbf{h}) \leq GL(\mathbf{f}, \mathbf{g})$, we return immediately the already computed path length $GL(\mathbf{f}, \mathbf{h})$ and thus only continue computation for points $\mathbf{h} \in \partial\Omega$ where $GL(\mathbf{f}, \mathbf{h}) \geq GL(\mathbf{f}, \mathbf{g})$. To do the above, we store the computed shortest-path lengths $GL(\mathbf{f}, \mathbf{g})$ (Eq. (2.15)) between a given feature point $\mathbf{f} \in \partial\Omega$ and all other spels $\mathbf{g} \in \partial\Omega$ into a map $L_{\mathbf{f}} : \partial\Omega \rightarrow \mathbb{R}^+$, $L_{\mathbf{f}}(\mathbf{g}) = GL(\mathbf{f}, \mathbf{g})$. The computation of the shortest-path length between a given spel $\mathbf{f} \in \partial\Omega$ and all other spels $\mathbf{g} \in \partial\Omega$ is presented in Algorithm 4, Section 2.3.3.

For $d = 2$, the problem is trivial since $\partial\Omega$ may consist of closed *one-dimensional* contours: For an arbitrary spel $\mathbf{f}_0$ in each contour $C \subset \partial\Omega$, we first compute in $L_{\mathbf{f}}(\mathbf{g})$ the path length from $\mathbf{f}_0$ to each spel $\mathbf{g} \in C$ while circumscribing C from $\mathbf{f}_0$ in a single orientation (clockwise or anticlockwise). Now, for any two spels $\mathbf{f}, \mathbf{g} \in C$, let $\Delta(\mathbf{f}, \mathbf{g}) = |L_{\mathbf{f}}(\mathbf{g}) - L_{\mathbf{f}}(\mathbf{f})|$. The geodesic length $GL(\mathbf{f}, \mathbf{g})$ between $\mathbf{f}$ and $\mathbf{g}$ is then given by

$$GL(\mathbf{f}, \mathbf{g}) = \min\{|C| - \Delta(\mathbf{f}, \mathbf{g}), \Delta(\mathbf{f}, \mathbf{g})\}, \tag{2.16}$$

where $|C|$ is the perimeter length of the contour C. However, for the purpose of finding one-spel-wide skeletons by Eq. (2.13), we can drop the absolute difference and redefine $\Delta(\mathbf{f}, \mathbf{g}) = L_{\mathbf{f}}(\mathbf{g}) - L_{\mathbf{f}}(\mathbf{f})$ for $\mathbf{f} = F(\mathbf{s})$ and $\mathbf{g} \in \mathcal{F}_{ext}(\mathbf{s})$ on the boundary graph $(\partial\Omega, \mathcal{A}_{\partial\Omega,\sqrt{d}})$.

The next section presents the IFT-based multiscale skeletonization algorithms for $d = 2$ and $d = 3$, respectively.

2.3.3 MULTISCALE SKELETONIZATION—PUTTING IT ALL TOGETHER

The complete 2D multiscale skeletonization algorithm is presented below.

Algorithm 2 (MULTISCALE SKELETON COMPUTATION IN 2D)**.**

INPUT:	An object Ω in dimension $d = 2$.
OUTPUT:	Multiscale skeleton importance ρ.
AUXILIARY:	Boundary $\partial\Omega$ with perimeter-length $\lvert\partial\Omega\rvert$; path length map $L_{\mathbf{f}}$; single-point feature transform F; component label map L_c; variable $tmp \in \mathbb{R}$.

```
1.  $(F, \partial\Omega, L_c) \leftarrow$ Algorithm 1$(\Omega)$.
2.  For each component $C \in (\partial\Omega, \mathcal{A}_{\partial\Omega,\sqrt{d}})$, do
3.      Select an arbitrary point $\mathbf{f}_0 \in C$.
4.      For each $\mathbf{g} \in C$ found by circumscribing $C$ from $\mathbf{f}_0$, do
5.          $L_{\mathbf{f}}(\mathbf{g}) \leftarrow$ path length from $\mathbf{f}_0$ to $\mathbf{g}$ on $C$.
6.  For each $\mathbf{s} \in \Omega \setminus \partial\Omega$, do
7.      $\rho(\mathbf{s}) \leftarrow 0$.
8.      Compute $\mathcal{F}_{ext}(\mathbf{s})$ by Eq. (2.12).
9.      For each $\mathbf{g} \in \mathcal{F}_{ext}(\mathbf{s})$, do
10.         If $L_c(\mathbf{g}) > L_c(F(\mathbf{s}))$, then set $\rho(\mathbf{s}) \leftarrow +\infty$ and return to 6.
11.         $tmp \leftarrow L_{\mathbf{f}}(\mathbf{g}) - L_{\mathbf{f}}(F(\mathbf{s}))$.
12.         If $tmp > |\partial\Omega| - tmp$, then $tmp \leftarrow |\partial\Omega| - tmp$.
13.         If $tmp > \rho(\mathbf{s})$, then $\rho(\mathbf{s}) \leftarrow tmp$.
14. Return $\rho$.
```

Line 1 finds the object boundary $\partial\Omega$, the single-point feature transform F, and the component label map L_c by Algorithm 1. The remaining lines follow the procedure described in Section 2.3.2.2 for $d = 2$. Lines 2–5 compute in $L_{\mathbf{f}}(\mathbf{g})$ the path length from an arbitrary spel $\mathbf{f}_0 \in C$, selected for each component $C \in (\partial\Omega, \mathcal{A}_{\partial\Omega,\sqrt{d}})$, to every spel $\mathbf{g} \in C$ while circumscribing the contour C. The main loop in Lines 6–13 computes for each spel $\mathbf{s} \in \Omega \setminus \partial\Omega$ the shortest-path length by Eq. (2.16) between point feature $F(\mathbf{s})$ and each $\mathbf{g} \in \mathcal{F}_{ext}(\mathbf{s})$ (Lines 11–12), and use them to update the MGF $\rho(\mathbf{s})$ in Line 13, as proposed in Eq. (2.13). The SKIZ is detected whenever a point feature $\mathbf{g} \in \mathcal{F}_{ext}(\mathbf{s})$ comes from a distinct component than $F(\mathbf{s})$. For one-pixel-wide connected SKIZ, $\mathbf{s}$ is selected as belonging to the SKIZ whenever $L_c(\mathbf{g}) > L_c(F(\mathbf{s}))$. In this case, $\rho(\mathbf{s})$ is set to the maximum possible value, and the algorithm returns to Line 6 (see example in Fig. 2.4). It should be clear that Algorithm 2 has complexity $O(|\Omega|)$.

A comment regarding the multiscale skeleton homotopy with the input shape is needed here. As visible from Fig. 2.4, the importance ρ has now a different varia-

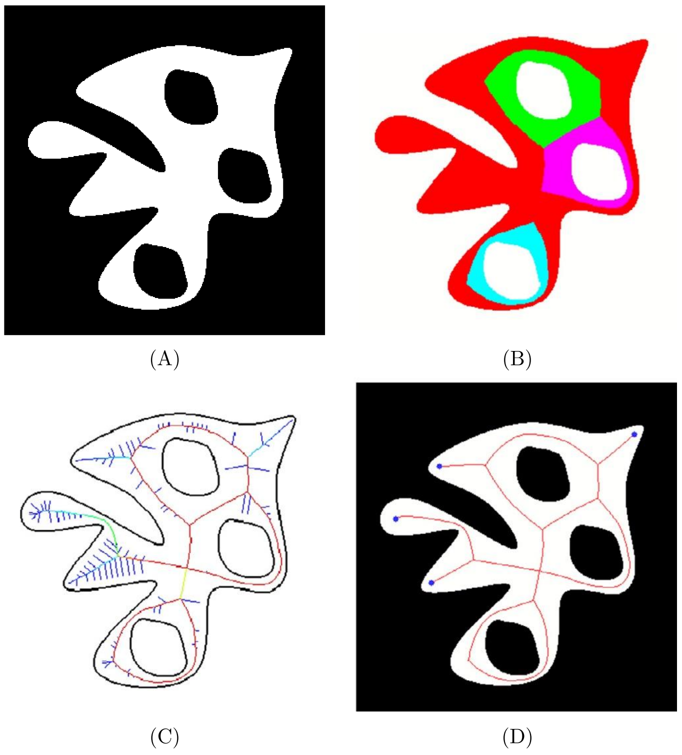

FIGURE 2.4

(A) A 2D object Ω with three holes. (B) The component label map L_c as computed by Algorithm 1. (C) The color-coded multiscale skeleton ρ of Ω, using the rainbow color map. The SKIZ is shown in red since its spels are assigned to the maximum importance in ρ. (D) A connected one-pixel-wide skeleton for a given scale of ρ, with its terminal points shown in blue.

tion across $\mathcal{S}$ than for genus-0 shapes (see, e.g., Fig. 2.5). Clearly, for sufficiently high thresholds, the skeleton in Fig. 2.4 will get disconnected, i.e., the three loops surrounding the holes in Ω will get separated from the central skeletal branch. Note that this *also* happens when using all other definitions of the same importance metric proposed by [35,17,53]. The root of the problem is that the collapsed-boundary importance metric used in all above works (and ours too) makes sense, in a multiscale way, only for genus-0 shapes whose skeleton is a tree. In other words, we know how to gradually simplify a tree (by removing its leafs), but we do not know how to do the same for a graph having loops. Issues here are how to assign an importance value to a loop (based on which geometric and/or topological criterion); and should the simplification of a loop remove it all at once, or should it allow its gradual disconnection. All these are (valid) questions that, however, go beyond our scope here.

For completeness, we note that disconnection of 2D nongenus-0 figures during simplification can be easily achieved, if this is a key issue. To do this, we can simply postprocess the computed skeleton $\mathcal{S}$: Trace all shortest paths in $\mathcal{S}$ linking each pair of loop components in $\mathcal{S}$ and assign spels along a value $+\infty$. This will only allow next the multiscale simplification of the tree parts of $\mathcal{S}$.

Essentially, Algorithm 2 is identical to the methods presented in [35,17,53]. As such, its main added-value is of theoretical nature—showing that 2D multiscale skeletonization can be easily cast in the IFT framework.

The situation in 3D ($d = 3$) is however very different: Here, our proposed multiscale skeletonization is *both* conceptually similar to the 2D case *and* very computationally efficient. This is in stark contrast with existing methods that are either similar in 2D and 3D but quite complex and do not provide an *explicit* definition of the regularization metric [27], or with existing methods that provide strongly related metrics in 2D [53,17,35] and 3D [14,39] but show a massive performance drop in the 3D case. The algorithm listed next shows our 3D multiscale skeletonization method. In contrast to the 2D proposal (Algorithm 2), we now use the efficient incremental shortest-path computation proposed in Algorithm 4.

Algorithm 3 (MULTISCALE SKELETON COMPUTATION IN 3D).

INPUT:	An object Ω in dimension $d = 3$.
OUTPUT:	Multiscale skeleton importance ρ.
AUXILIARY:	Priority queue Q; list $\mathcal{V}$ of boundary points that have been inserted in Q; A^* path-cost map G; boundary $\partial\Omega$; sets $\mathcal{C}(\mathbf{s})$, $\forall \mathbf{s} \in \partial\Omega$; shortest-path length map $L_{\mathbf{f}}$; single-point feature transform F.

```
1.  $(F, \partial\Omega) \leftarrow$ Algorithm 1$(\Omega)$.
2.  For each $\mathbf{s} \in \Omega \setminus \partial\Omega$, do
3.      Insert $\mathbf{s}$ in $\mathcal{C}(F(\mathbf{s}))$.
4.  For each $\mathbf{f} \in \partial\Omega$, do
5.      $L_{\mathbf{f}}(\mathbf{f}) \leftarrow +\infty$; $G(\mathbf{f}) \leftarrow +\infty$.
6.  $Q \leftarrow \emptyset$; $\mathcal{V} \leftarrow \emptyset$.
7.  For each $\mathbf{f} \in \partial\Omega$, do
8.      $L_{\mathbf{f}}(\mathbf{f}) \leftarrow 0$; $G(\mathbf{f}) \leftarrow 0$; insert $\mathbf{f}$ in $Q$; insert $\mathbf{f}$ in $\mathcal{V}$.
9.      While there exists $\mathbf{s} \in \mathcal{C}(\mathbf{f})$, do
10.         Remove $\mathbf{s}$ from $\mathcal{C}(\mathbf{f})$.
11.         $\rho(\mathbf{s}) \leftarrow 0$; compute $\mathcal{F}_{ext}(\mathbf{s})$ by Eq. (2.12).
12.         For each $\mathbf{g} \in \mathcal{F}_{ext}(\mathbf{s})$, do
13.             $L_{\mathbf{f}}(\mathbf{g}) \leftarrow$ Algorithm 4$(\partial\Omega, \mathbf{g}, Q, \mathcal{V}, G, L_{\mathbf{f}})$.
14.             If $L_{\mathbf{f}}(\mathbf{g}) > \rho(\mathbf{s})$, then $\rho(\mathbf{s}) \leftarrow L_{\mathbf{f}}(\mathbf{g})$.
15.     For each $\mathbf{g} \in \mathcal{V}$
16.         $L_{\mathbf{f}}(\mathbf{g}) \leftarrow +\infty$; $G(\mathbf{g}) \leftarrow +\infty$.
17.     $Q \leftarrow \emptyset$; $\mathcal{V} \leftarrow \emptyset$.
18. Return $\rho$.
```

Line 1 finds the object boundary $\partial\Omega$ and the single-point feature transform F by Algorithm 1. We are not interested in L_c since Algorithm 3 assumes that $\partial\Omega$ is a single surface. Lines 2–3 compute the sets $\mathcal{C}(\mathbf{f}) = \{\mathbf{s} \in \Omega \setminus \partial\Omega | F(\mathbf{s}) = \mathbf{f}\}$ that speed up shortest-path length computations, as described in Section 2.3.2.2 for $d = 3$. Note that G stores the A^* path costs, whereas $L_\mathbf{f}$ stores the desired path lengths, in Algorithm 4. The shortest-path lengths from each boundary point $f \in \partial\Omega$ to other boundary points $g \in \partial\Omega$ are *incrementally* computed in Algorithm 4 (Eq. (2.15)). Therefore, the trivial-path value initialization of ψ_{geo} (Eq. (2.14)) must be performed outside Algorithm 4 (Lines 4–5 before the main loop of Lines 7–17, and Lines 15–16 and 8 to restart computation for every initial boundary point $f \in \partial\Omega$). Lines 4–5 execute for the entire boundary $\partial\Omega$, so the purpose of set $\mathcal{V}$ is to revisit only the boundary points used in Algorithm 4, when reinitializing $L_\mathbf{f}$ and G. Line 8 initializes the priority queue Q and sets $\mathcal{V}$ with one initial boundary point f for Algorithm 4. The loop of Lines 9–14 computes the 3D MGF $\rho(\mathbf{s})$ by Eq. (2.13) for each spel $\mathbf{s}$ whose the single-point feature is the current point $\mathbf{f} \in \partial\Omega$. Line 10 removes a spel $\mathbf{s}$ from $\mathcal{C}(\mathbf{f})$, Line 11 initializes $\rho(\mathbf{s})$ and finds $\mathcal{F}_{ext}(\mathbf{s})$ by Eq. (2.12). For each point feature $\mathbf{g} \in \mathcal{F}_{ext}(\mathbf{s})$, Line 13 finds $GL(\mathbf{f}, \mathbf{g})$ (Eq. (2.15)) and stores it in $L_\mathbf{f}(\mathbf{g})$, and Line 14 updates $\rho(\mathbf{s})$ according to Eq. (2.13). Algorithm 4 is presented next.

Algorithm 4 (INCREMENTAL SHORTEST-PATH LENGTH COMPUTATION)**.**

INPUT:	Boundary $\partial\Omega$; terminal node $\mathbf{g} \in \partial\Omega$; priority queue Q; boundary points $\mathcal{V}$ that have been inserted in Q; A^* cost map G; shortest-path-length map $L_\mathbf{f}$.
OUTPUT:	Shortest-path length $L_\mathbf{f}(\mathbf{g})$ at the terminal node with respect to the current starting node $\mathbf{f}$ chosen in Algorithm 3.
AUXILIARY:	Variable $tmp \in \mathbb{R}$.

```
1.  If $L_\mathbf{f}(\mathbf{g}) \neq +\infty$, then return $L_\mathbf{f}(\mathbf{g})$.
2.  While $Q \neq \emptyset$ do
3.      Remove $\mathbf{w}$ from $Q$, where $G(\mathbf{w})$ is minimal over $Q$.
4.      If $\mathbf{w} = \mathbf{g}$, then return $L_\mathbf{f}(\mathbf{g})$.
5.      For each $\mathbf{h} \in \mathcal{A}_{\partial\Omega,\sqrt{d}}(\mathbf{w})$ such that $G(\mathbf{h}) > G(\mathbf{w})$, do
6.          $tmp \leftarrow L_\mathbf{f}(\mathbf{w}) + \|\mathbf{h} - \mathbf{w}\| + \|\mathbf{g} - \mathbf{h}\|$.
7.          If $tmp < G(\mathbf{h})$, then
8.              $G(\mathbf{h}) \leftarrow tmp$; $L_\mathbf{f}(\mathbf{h}) \leftarrow L_\mathbf{f}(\mathbf{w}) + \|\mathbf{h} - \mathbf{w}\|$.
9.              If $G(\mathbf{h}) \neq +\infty$, then
10.                 Update position of $\mathbf{h}$ in $Q$.
11.             Else
12.                 Insert $\mathbf{h}$ in $Q$ and in $\mathcal{V}$.
```

Line 1 halts computation whenever the shortest-path length from $\mathbf{f}$ to $\mathbf{g}$ on $\partial\Omega$ has already been computed in a previous execution of Algorithm 4. The main loop of Lines 2–12 computes the shortest-path length to every boundary point $\mathbf{w} \in \partial\Omega$ in a nondecreasing order of the cost values in G until it finds the terminal point $\mathbf{g}$ in Line 4. In Line 3, when a point $\mathbf{w} \in \partial\Omega$ is removed from Q, $G(\mathbf{w})$ stores the minimum A^* path cost, and $L_\mathbf{f}(\mathbf{w})$ stores the shortest-path length from $\mathbf{f}$ to $\mathbf{w}$ on $\partial\Omega$, which may be used for early termination in Line 1 in a next execution of Algorithm 4. The loop in Lines 5–12 evaluates if $\mathbf{w}$ can offer a lower path cost $L_{\mathbf{f}}(\mathbf{w}) + \|\mathbf{h} - \mathbf{w}\| + \|\mathbf{g} - \mathbf{h}\|$ (value of an extended path $\pi_{\mathbf{w}} \cdot \langle \mathbf{w}, \mathbf{h} \rangle$ in Eq. (2.14)) to the current value assigned to an adjacent point $\mathbf{h} \in \partial\Omega$ (Lines 6–7). If this is the case, then Lines 8–12 update $G(\mathbf{h})$, $L_f(\mathbf{h})$, and the status of $\mathbf{h}$ in Q, accordingly.

The complexity of Algorithm 3 would be $O(|\Omega \setminus \partial\Omega||\partial\Omega| \log |\partial\Omega|)$ with a naive implementation of shortest-path length computation. In practice, however, Algorithm 4 finishes in Lines 1 or 4 much earlier than visiting all boundary points. This makes a considerable reduction in the processing time of Algorithm 3, as we will see next.

2.4 COMPARATIVE ANALYSIS

We next present and discuss our results as compared to other state-of-the-art multiscale skeletonization methods.

2.4.1 2D MEDIAL AXES

We first consider medial axes of 2D objects. Here, we compare our IFT method with its two main competitors, the augmented fast-marching method (AFMM) [53] (basically identical to [35,17]) and the more recent advection-based method (AS) in [27]. We compared the above three methods on a set of over 30 2D shapes, taken from relevant papers in the field [44,5,35,53]. Fig. 2.5 shows three such shapes with their progressively simplified skeletons. It is clearly visible that all three methods yield nearly identical skeletons, both in terms of location *and* importance values. In other words, our IFT-based method can compute multiscale 2D skeletons, which are nearly identical to those computed by existing methods. As visible, our method handles complex, noisy, and variable-scale shapes with the same ease as the other two analyzed methods.

2.4.2 3D MEDIAL SURFACES

2.4.2.1 *Global Comparison*

For 3D shapes, we compared our IFT-based methods with two classes of competing techniques. First, and most interesting, we considered all techniques that we are aware of that produce multiscale skeletons, in the sense described in Section 2.2.2. These are the multiscale MGF-based method in [39] (MS), the advection-based method in [27] (AS), and the multiscale ball-shrinking method that implements the MGF metric in [39] for mesh models [26] (MBS). Secondly, to illustrate the advantage of multiscale regularization, we compare our method with three local regularization nonmultiscale methods: Hamilton–Jacobi skeletons (HJ [44]), the Integer Medial Axis (IMA [25]), and Iterative Thinning Process (ITP [28]). We have chosen these methods since they are well known in the 3D skeletonization arena, are relatively efficient, produce good-quality 3D surface skeletons, and have public implementations.

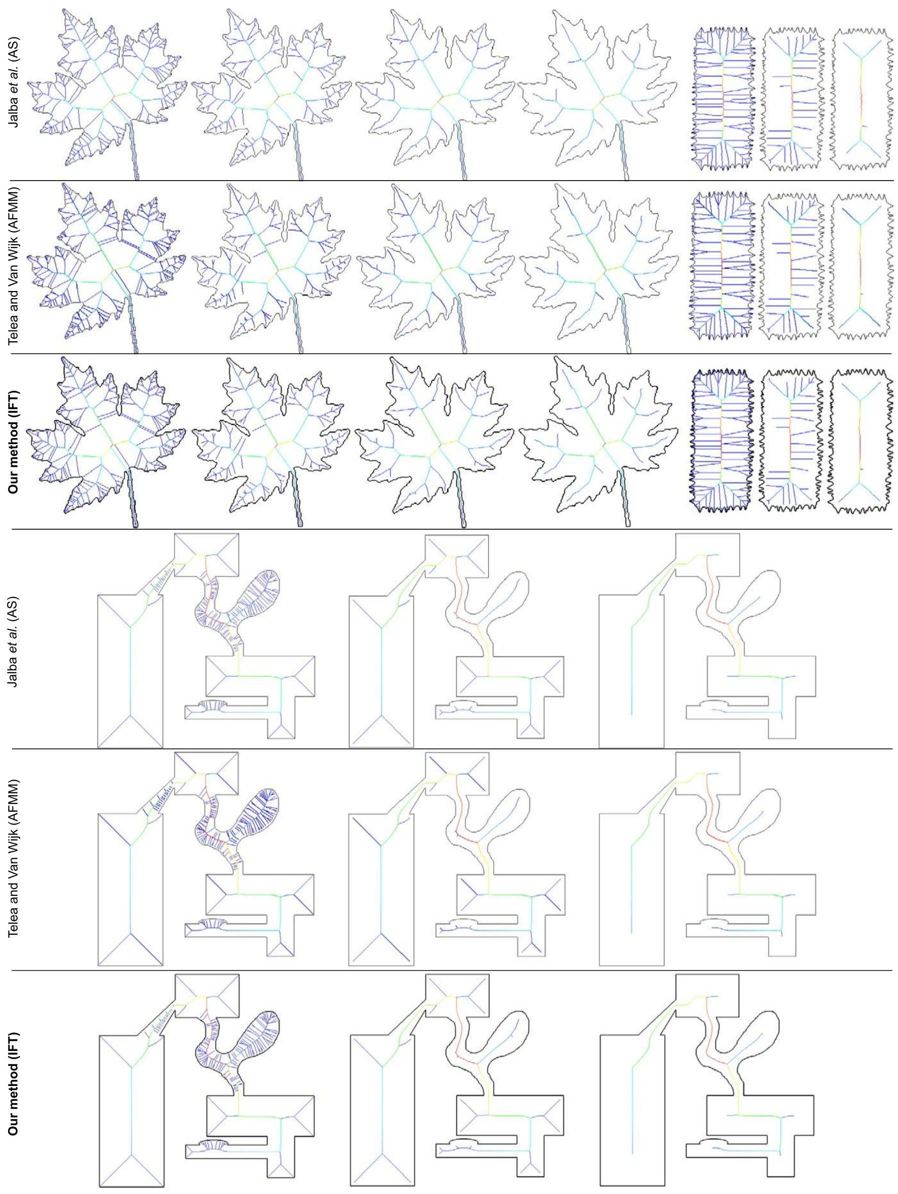

FIGURE 2.5

Our multiscale 2D skeletons compared with AFMM and AS.

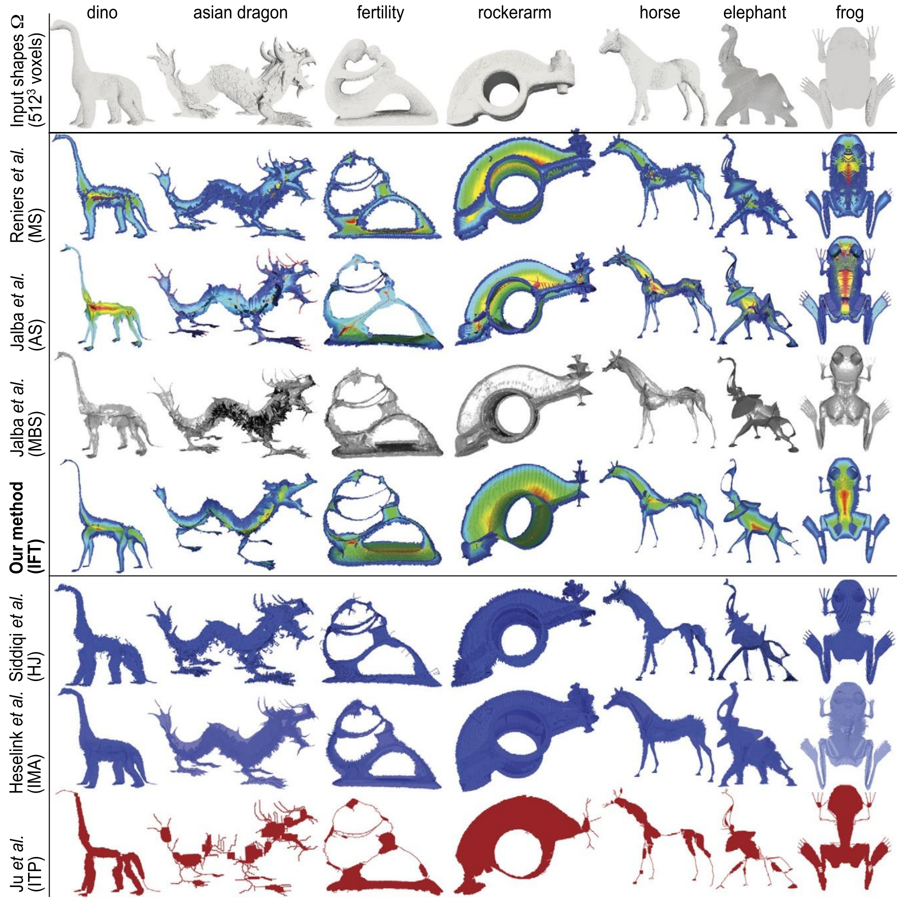

FIGURE 2.6

Global comparison of our 3D skeletonization method (IFT) with three multiscale skeletonization methods (MS, AS, MBS) and with three additional nonmultiscale methods (HJ, IMA, ITP). See Section 2.4.2.1.

Fig. 2.6 shows the results of the above-mentioned comparisons for seven shapes, processed by seven skeletonization methods. The multiscale skeletons computed by MS and AS are color-coded to reflect the importance metric, using a rainbow colormap, just as in Fig. 2.5. Multiscale skeletons computed by MBS are not importance color-coded in Fig. 2.6; the MBS importance is discussed separately in more detail in Section 2.4.2.2.

Quality-wise, our 3D surface skeletons are voxel-thin, centered within the shape (within the margin allowed by the voxel resolution), and have the same number of connected components and loops as the input shape by construction. These are key properties required by any skeletonization method [8]. For example, IMA yields centered and voxel-thin skeletons, but these can get disconnected when simplified too much since this method essentially uses the local angle-and-distance-based simplification criterion of the θ-SMA method of Foskey et al. [22]. Note that the disconnection implied above is not due to the existence of loops in the skeleton: IMA can easily disconnect also skeletons of genus-0 shapes. This does not happen with our method. Conversely, HJ yields connected skeletons, but for this, the method uses a thinning process ordered by the divergence of the distance transform gradient, which must explicitly checked to preserve homotopy [44]. The ITP method computes skeletons that are voxel-thin and homotopic to the input shape but not well centered in the shape, as seen by the various zig-zag branches of the *dragon* model (Fig. 2.6, bottom row).

The sensitivity of the skeletons shown in Fig. 2.6 to noise or small-scale details on the input shape surface varies quite a lot. As known, local regularization methods such as HJ, IMA, and ITP are more noise-sensitive than global regularization methods such as MS, AS, and MBS [50]. Our method (IFT) falls in the latter class of global methods, so it is less sensitive to noise and produces smoother surface skeletons, as visible in Fig. 2.6, fourth row from bottom.

2.4.2.2 Detailed Comparison

To gain more insight, we next compare our IFT method with several methods we are aware of that compute *multiscale* 3D surface skeletons (AS, MS, and MBS). The first two methods (AS, MS) are voxel-based, whereas the last one (MBS) is mesh-based. Figs. 2.7 and 2.8 show results for a selected set of shapes. Since all the above-mentioned methods produce multiscale skeletons, we regularized these by removing very low importance (spurious) skeleton points to yield comparably simplified skeletons. Several observations can be made when studying the compared methods as follows.

Regularization: Figs. 2.7 and 2.8 show that IFT delivers 3D surface skeletons that are, geometrically speaking, very similar to the ones produced by AS, MS, and MBS. This, in itself, is a good indication of quality of IFT. Indeed, surface-skeletonization methods should deliver similar results, given that they all aim to approximate the *same* surface skeleton definition (Eq. (2.2)). Secondly, we see that the IFT delivers the same degree of small-scale noise removal to create smooth and clean skeletal manifolds as AS, MS, and MBS, so it can be used for robust skeleton regularization. The IFT regularization is as easy to use as the one proposed by the other methods, the setting of a single importance thresholding parameter τ. Note that this is far simpler than the regularization proposed by local methods, e.g., HJ, IMA, or ITP, which require the careful setting of one or several parameters to obtain comparable results.

A more subtle insight regards the gradient of the importance metric ρ from the skeleton boundary $\partial\mathcal{S}$ to its center, visible in Figs. 2.7 and 2.8 in terms of the blue-to-red color change. All tested methods (IFT, AS, MS, MBF) yield a ρ that increases monotonically from $\partial\mathcal{S}$ to the center of $\mathcal{S}$. Separately, we see that ρ for IFT, MBS,

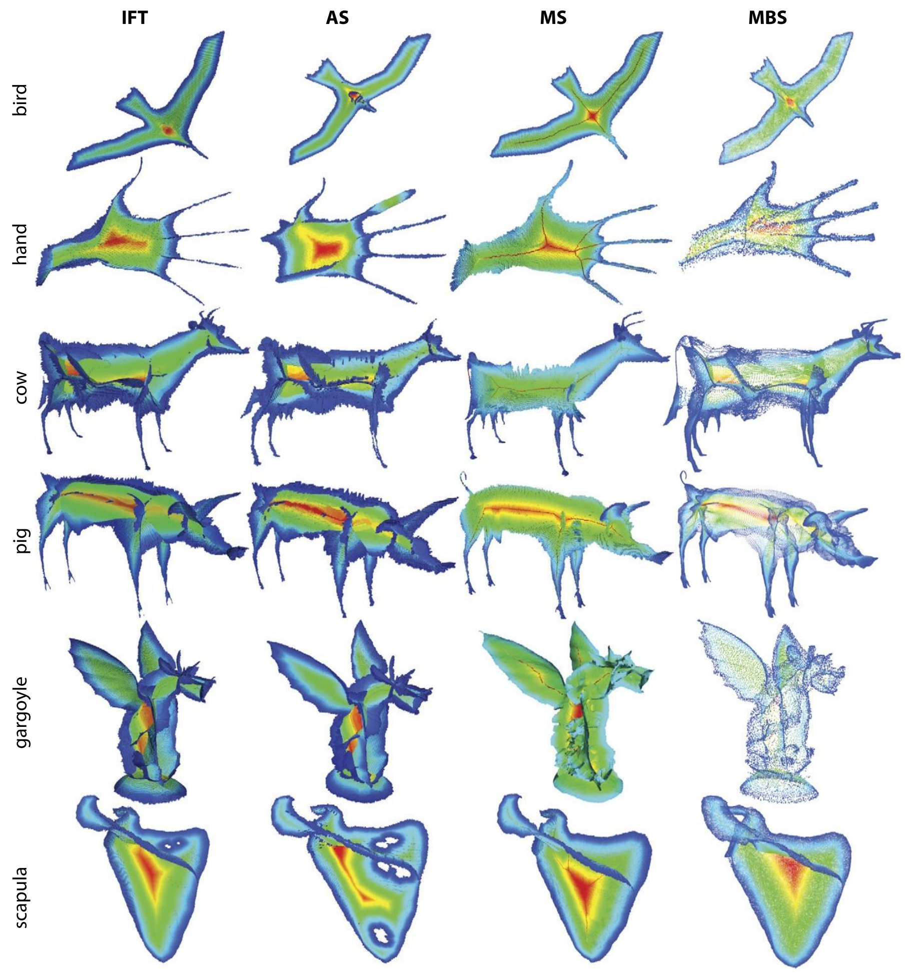

FIGURE 2.7

Detail comparison of 3D surface skeletons computed by our method (IFT) and other multiscale methods (MS, AS, and MBS). See Section 2.4.2.2.

and MS is not just increasing from $\partial\mathcal{S}$ to the center of $\mathcal{S}$, but has a very similar gradient. This implies that our method (IFT) delivers an importance metric ρ that is very similar to the ones delivered by MS and MBS. Since MS and MBS compute the medial geodesic function (MGF) metric, it follows that IFT also computes a very similar metric. This is indeed the expected outcome given the IFT algorithm (see Section 2.3.3). In contrast, the gradient of ρ delivered by AS is quite different. This is explained by the fact that AS is the only multiscale skeletonization method in the studied set that does not explicitly use the MGF metric.

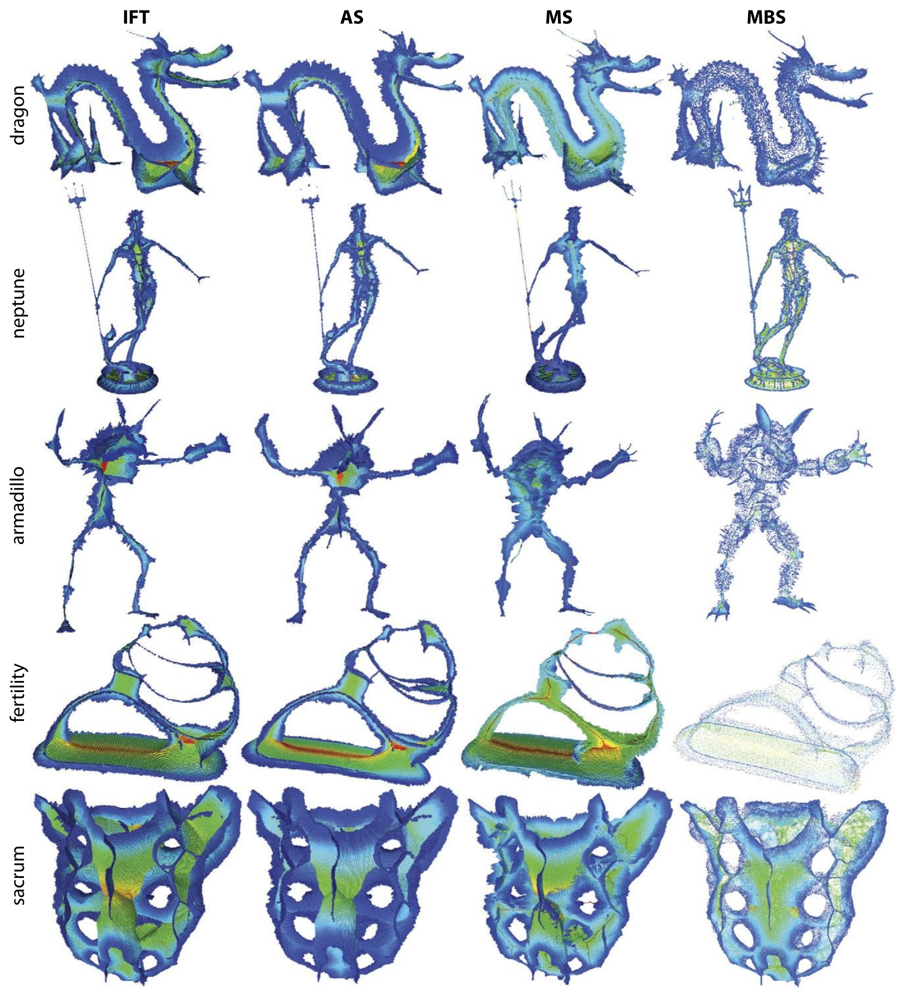

FIGURE 2.8

Additional examples of 3D multiscale skeletons computed by our method (IFT) and other multiscale methods (MS, AS, and MBS). See Section 2.4.2.2.

Connectivity: IFT, AS, and MS deliver a compact surface skeleton, whereas MBS delivers only a disconnected point cloud. This makes IFT (and AS and MS) more interesting than MBS for practical applications where one requires a compact surface skeleton. Indeed, point-cloud skeletons require complex post-processing methods for reconstructing a compact representation [26,29]. Voxel skeletons do not have this problem.

Table 2.1 Timings (seconds) for the compared surface skeletonization methods

Dataset	$\|\Omega\|$	MS [39]	AS [27]	IFT	$\|\partial\Omega\|^{mesh}$	MBS [26]
Bird	445,690	64.21	13.86	8.64	46,866	18.69
Hand	776,869	62.94	2.07	4.36	49,374	15.5
Cow	6,343,478	177.80	39.86	17.73	181,823	96.54
Pig	5,496,145	181.95	34.84	23.89	225,282	142.02
Gargoyle	6,614,154	566.52	25.66	79.43	25,002	7.54
Scapula	2,394,694	1717.37	29.99	609.33	116,930	102.57
Dragon	7,017,452	322.81	39.3	32.86	100,250	49.01
Neptune	2,870,546	322.75	47.25	68.72	28,052	5.85
Armadillo	1,854,858	45.43	7.2	4.25	172,952	104.65
Fertility	1,264,132	99.62	6.15	8.46	24,994	6.15
Sacrum	12,637,931	2015.59	39.83	417.54	204,710	213.49

Scalability: We implemented all tested methods in C++ and ran them on an Intel 3.5 GHz 8-core 32 MB RAM PC. The methods MBS, AS, and MS use CPU multithreading parallelization, as described in the respective papers. No GPU parallelization was used for MBS. Our method (IFT) is purely serial. Table 2.1 shows the timings for the compared methods for the shapes depicted in Figs. 2.7 and 2.8. Column $|\Omega|$ gives the number of foreground voxels of the tested models with MS, AS, and IFT. For MBS, the comparable metric, the number of sample points of the input mesh, is given in column $|\partial\Omega|^{mesh}$.

When testing scalability on large voxel volumes, we found that the MS implementation from [39] encountered problems: For the *scapula* shape (Fig. 2.7, bottom row), MS could not handle the 512^3-voxel resolution of our model, so we reduced the resolution to 370^3. At this resolution, the shape shows visible holes due to the very thin wall thickness (a few voxels). In contrast, IFT and AS (which are both voxel-based methods) could handle 512^3-voxel volumes without problems.

Performance-wise, Table 2.1 shows that IFT is roughly 3 to 10 times faster than MS, which is the only voxel-based method that implements the same MGF importance metric. This is an important result, as it tells us that the IFT algorithm produces significant speed-ups for the geodesic length evaluation, which was one of its main goals. Compared to AS, IFT is faster on some models but considerably slower on the *sacrum* and *scapula* models. This is explained by the fact that the complexity of AS is roughly $O(K|\partial\Omega| \log |\partial\Omega|)$, where K is $\max_{\mathbf{x}\in\Omega} \mathcal{D}(\mathbf{x})$, that is, the shape thickness. In contrast, the complexity of MS is roughly $O(L|\partial\Omega|)$, where K is the average geodesic-path length between two feature points on $\partial\Omega$. For large and locally tubular shapes, such as *cow* or *pig*, IFT is thus faster. For relatively thin and large-surface shapes, such as *scapula* and *sacrum*, the geodesic computation cost becomes very high, so IFT is slower than AS. However, as outlined earlier, this extra price of IFT delivers a higher-quality regularization in terms of smoothness of the importance metric. We note a similar effect when comparing IFT with MBS: for locally tubular shapes, IFT is faster than MBS, especially when the latter considers high-resolution mesh models. For locally thin and large-surface shapes, IFT becomes slower than MBS. Again, this extra price of IFT is counterbalanced by the higher-quality regularization metric it delivers and also by the fact that IFT delivers connected skeletons, whereas MBS delivers only a skeletal point-cloud. All in all, we argue that the performance of IFT compares favorably with methods using the same importance metric (MS) but also with other multiscale skeletonization methods (AS, MBS). This is especially salient when considering that we implemented IFT as a purely *serial* algorithm, whereas MS, AS, and MBS all use CPU-side 8-core multithreading parallelization.

2.5 CONCLUSION

In this chapter, we have presented a novel way of computing multiscale 2D medial axes and 3D surface skeletons of image, respectively voxel datasets. For this, we cast the problem of computing the medial geodesic function (MGF) regularization metric, known for its ability to deliver high-quality multiscale skeletons in the computation of optimal path forests with the Image Foresting Transform (IFT) framework. We show that the delivered 2D and 3D skeletons compare very favorably from the perspective of similarity and regularization with several other known multiscale skeletonization methods. Our IFT-based implementation is very simple and delivers good performance. To our knowledge, or method is the second one (aside [27]) that can compute both 2D and 3D multiscale medial skeletons with a unified formulation.

Several extensions of this work are possible. Performance-wise, extending IFT to use multithreaded parallelization has the potential to make this method the fastest multiscale skeletonization technique for 2D skeletons and 3D surface skeletons on the CPU in existence. Application-wise, the IFT framework allows one to easily change the cost function, thereby enabling one to design a whole family of multiscale regularization metrics beyond the MGF metric. Such metrics could, in turn, support various types of applications, such as feature-sensitive regularization. Finally, an interesting extension regards the computation of multiscale 3D curve skeletons.

REFERENCES

[1] N. Amenta, S. Choi, R. Kolluri, The power crust, in: Proc. SMA, ACM, 2001, pp. 65–73.
[2] W.P. Amorim, A.X. Falcão, M.H. de Carvalho, Semi-supervised pattern classification using optimum-path forest, in: 2014 27th SIBGRAPI Conference on Graphics, Patterns and Images, 2014, pp. 111–118.
[3] F. Andaló, P. Miranda, R. da Silva Torres, A. Falcão, Shape feature extraction and description based on tensor scale, Pattern Recognit. 43 (1) (2010) 26–36.
[4] A. Torsello, E. Hancock, Correcting curvature-density effects in the Hamilton–Jacobi skeleton, IEEE Trans. Image Process. 15 (4) (2006) 877–891.

[5] S. Bouix, K. Siddiqi, A. Tannenbaum, Flux driven automatic centerline extraction, Med. Image Anal. 9 (3) (2005) 209–221.
[6] M. Chang, F. Leymarie, B. Kimia, Surface reconstruction from point clouds by transforming the medial scaffold, Comput. Vis. Image Underst. 113 (11) (2009) 1130–1146.
[7] K. Ciesielski, P. Miranda, A. Falcão, J. Udupa, Joint graph cut and relative fuzzy connectedness image segmentation algorithm, Med. Image Anal. 17 (8) (2013) 1046–1057.
[8] N. Cornea, D. Silver, X. Yuan, R. Balasubramanian, Computing hierarchical curve-skeletons of 3D objects, Vis. Comput. 21 (11) (2005) 945–955.
[9] M. Couprie, Topological maps and robust hierarchical Euclidean skeletons in cubical complexes, Comput. Vis. Image Underst. 117 (4) (2013) 355–369.
[10] R. da, S. Torres, A. Falcão, Contour salience descriptors for effective image retrieval and analysis, Image Vis. Comput. 25 (1) (2007) 3–13.
[11] R. da, S. Torres, A. Falcão, L. da, F. Costa, A graph-based approach for multiscale shape analysis, Pattern Recognit. 37 (6) (2004) 1163–1174.
[12] J. Damon, Global medial structure of regions in $\mathbb{R}^3$, Geom. Topol. 10 (2006) 2385–2429.
[13] P. de Miranda, A. Falcão, J. Udupa, Synergistic arc-weight estimation for interactive image segmentation using graphs, Comput. Vis. Image Underst. 114 (1) (2010) 85–99.
[14] T. Dey, J. Sun, Defining and computing curve skeletons with medial geodesic functions, in: Proc. SGP, IEEE, 2006, pp. 143–152.
[15] A. Falcão, B. da Cunha, R. Lotufo, Design of connected operators using the image foresting transform, in: Proc. SPIE, vol. 4322, 2001, pp. 468–479.
[16] A. Falcão, F. Bergo, Interactive volume segmentation with differential image foresting transforms, IEEE Trans. Med. Imaging 23 (9) (2004) 1100–1108.
[17] A. Falcão, L. da, F. Costa, B. da Cunha, Multiscale skeletons by image foresting transform and its applications to neuromorphometry, Pattern Recognit. 35 (7) (2002) 1571–1582.
[18] A. Falcão, J. Udupa, A 3D generalization of user-steered live-wire segmentation, Med. Image Anal. 4 (4) (2000) 389–402.
[19] A.X. Falcão, J. Stolfi, R.A. Lotufo, The image foresting transform: theory, algorithms, and applications, IEEE Trans. Pattern Anal. Mach. Intell. 26 (1) (2004) 19–29.
[20] A.X. Falcão, J.K. Udupa, F.K. Miyazawa, An ultra-fast user-steered image segmentation paradigm: live wire on the fly, IEEE Trans. Med. Imaging 19 (1) (2000) 55–62.
[21] C. Feng, A. Jalba, A. Telea, Improved part-based segmentation of voxel shapes by skeleton cut spaces, Math. Morph. Theory Appl. 1 (1) (2016) 60–78.
[22] M. Foskey, M. Lin, D. Manocha, Efficient computation of a simplified medial axis, in: Proc. Shape Modeling, 2003, pp. 135–142.
[23] P. Giblin, B. Kimia, A formal classification of 3D medial axis points and their local geometry, IEEE Trans. Pattern Anal. Mach. Intell. 26 (2) (2004) 238–251.
[24] J. Giesen, B. Miklos, M. Pauly, C. Wormser, The scale axis transform, in: Proc. Annual Symp. Comp. Geom, 2009, pp. 106–115.
[25] W. Hesselink, 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. 30 (12) (2008) 2204–2217.
[26] A. Jalba, J. Kustra, A. Telea, Surface and curve skeletonization of large 3D models on the GPU, IEEE Trans. Pattern Anal. Mach. Intell. 35 (6) (2013) 1495–1508.
[27] A. Jalba, A. Sobiecki, A. Telea, An unified multiscale framework for planar, surface, and curve skeletonization, IEEE Trans. Pattern Anal. Mach. Intell. 38 (1) (2015) 30–45.
[28] T. Ju, M. Baker, W. Chiu, Computing a family of skeletons of volumetric models for shape description, Comput. Aided Des. 39 (5) (2007) 352–360.

[29] J. Kustra, A. Jalba, A. Telea, Robust segmentation of multiple manifolds from unoriented noisy point clouds, Comput. Graph. Forum 33 (1) (2014) 73–87.
[30] F. Leymarie, B. Kimia, The medial scaffold of 3D unorganized point clouds, IEEE Trans. Vis. Comput. Graph. 29 (2) (2007) 313–330.
[31] G. Malandain, S. Fernandez-Vidal, Euclidean skeletons, Image Vis. Comput. 16 (5) (1998) 317–327.
[32] B. Miklos, J. Giesen, M. Pauly, Discrete scale axis representations for 3D geometry, in: Proc. ACM SIGGRAPH, 2010, pp. 394–493.
[33] P.A.V. Miranda, A.X. Falcão, T.V. Spina, Riverbed: a novel user-steered image segmentation method based on optimum boundary tracking, IEEE Trans. Image Process. 21 (6) (2012) 3042–3052.
[34] N.J. Nilsson, Artificial Intelligence: A New Synthesis, 1st edition, Morgan Kaufmann Publishers, Inc., 1998.
[35] R.L. Ogniewicz, O. Kubler, Hierarchic Voronoi skeletons, Pattern Recognit. 28 (3) (1995) 343–359.
[36] J. Papa, A. Falcão, V. de Albuquerque, J. Tavares, Efficient supervised optimum-path forest classification for large datasets, Pattern Recognit. 45 (1) (2012) 512–520.
[37] S. Pizer, K. Siddiqi, G. Szekely, J. Damon, S. Zucker, Multiscale medial loci and their properties, Int. J. Comput. Vis. 55 (2–3) (2003) 155–179.
[38] S. Prohaska, H.C. Hege, Fast visualization of plane-like structures in voxel data, in: Proc. IEEE Visualization, 2002, pp. 29–36.
[39] D. Reniers, J.J. van Wijk, A. Telea, Computing multiscale skeletons of genus 0 objects using a global importance measure, IEEE Trans. Vis. Comput. Graph. 14 (2) (2008) 355–368.
[40] L. Rocha, F. Cappabianco, A. Falcão, Data clustering as an optimum-path forest problem with applications in image analysis, Int. J. Imaging Syst. Technol. 19 (2) (2009) 50–68.
[41] L. Rossi, A. Torsello, An adaptive hierarchical approach to the extraction of high resolution medial surfaces, in: Proc. 3DIMPVT, 2012, pp. 371–378.
[42] M. Rumpf, A. Telea, A continuous skeletonization method based on level sets, in: Proc. VisSym, 2002, pp. 151–158.
[43] D. Shaked, A. Bruckstein, Pruning medial axes, Comput. Vis. Image Underst. 69 (2) (1998) 156–169.
[44] K. Siddiqi, S. Bouix, A. Tannenbaum, S. Zucker, Hamilton–Jacobi skeletons, Int. J. Comput. Vis. 48 (3) (2002) 215–231.
[45] K. Siddiqi, S. Pizer, Medial Representations: Mathematics, Algorithms and Applications, Springer, 2009.
[46] T. Spina, P. de Miranda, A. Falcão, Hybrid approaches for interactive image segmentation using the live markers paradigm, IEEE Trans. Image Process. 23 (12) (2014) 5756–5769.
[47] S. Stolpner, S. Whitesides, K. Siddiqi, Sampled medial loci and boundary differential geometry, in: Proc. IEEE 3DIM, 2009, pp. 87–95.
[48] S. Stolpner, S. Whitesides, K. Siddiqi, Sampled medial loci for 3D shape representation, Comput. Vis. Image Underst. 115 (5) (2011) 695–706.
[49] A. Sud, M. Foskey, D. Manocha, Homotopy-preserving medial axis simplification, in: Proc. SPM, 2005, pp. 103–110.
[50] A. Tagliasacchi, T. Delame, M. Spagnuolo, N. Amenta, A. Telea, 3D skeletons: a state-of-the-art report, Comput. Graph. Forum 35 (2) (2016) 573–597.
[51] A. Tagliasacchi, H. Zhang, D. Cohen-Or, Curve skeleton extraction from incomplete point cloud, in: Proc. SIGGRAPH, 2009, pp. 541–550.

[52] A. Telea, Feature preserving smoothing of shapes using saliency skeletons, in: Visualization in Medicine and Life Sciences, Springer, 2012, pp. 153–170.
[53] A. Telea, J.J. van Wijk, An augmented fast marching method for computing skeletons and centerlines, in: Proc. VisSym, 2002, pp. 251–259.