

Curve-Skeleton Properties, Applications, and Algorithms

Nicu D. Cornea, Deborah Silver, *Member, IEEE*, and Patrick Min

Abstract—Curve-skeletons are thinned 1D representations of 3D objects useful for many visualization tasks including virtual navigation, reduced-model formulation, visualization improvement, animation, etc. There are many algorithms in the literature describing extraction methodologies for different applications; however, it is unclear how general and robust they are. In this paper, we provide an overview of many curve-skeleton applications and compile a set of desired properties of such representations. We also give a taxonomy of methods and analyze the advantages and drawbacks of each class of algorithms.

Index Terms—Computer graphics, computational geometry, object modeling, curve, surface, solid, and object representations, curve-skeletons.

1 INTRODUCTION

THREE-DIMENSIONAL models are common in many disciplines, including computer-aided design, medical imaging, computer graphics, scientific visualization, computational fluid dynamics, and remote sensing. While the 3D representation is invaluable, many applications require alternate “compact” representations of these models. One such representation is a line-like or stick-like 1D representation, which is sometimes referred to as a “skeletal representation” or “curve-skeleton” [120]. This is different from the skeletal-surface representation (medial surface), which is a higher dimensional structure. The curve-skeleton captures the essential topology of the underlying object in an easy to understand and very compact form. Examples of applications that use a curve-skeleton include: virtual navigation, registration, animation, morphing, scientific analysis, shape recognition, and shape retrieval.

One of the difficulties is that a “curve-skeleton” is an ill-defined object. This has led to a large number of algorithms and heuristics in the literature and many more constantly being proposed. Many of the algorithms in the literature use different definitions, parameters, and thresholds, and demonstrate their performance on a limited number of diverse 3D objects. Additionally, some are fine-tuned for a specific application.

As a consequence, many of these algorithms cannot be replicated and most major visualization and medical image processing packages do not use them. It is hard to decide

which algorithm to choose since there are no criteria for evaluation, thereby causing a further proliferation of new algorithms. What is needed is an analysis of the desired properties of the curve-skeleton, as required by the various applications, and how the various existing curve-skeletonization methods satisfy these properties.

In this paper, we present a list of properties for curve-skeletons based upon numerous applications. We also categorize many of the existing algorithms into classes based upon implementation, and we discuss how these classes achieve the various properties. In addition, one algorithm from each class has been implemented and tested on the same set of 3D shapes. The main goal of this paper is to provide an overview of curve-skeletonization applications and implementations to help guide visualization users and developers.

This work is an extension of our previous conference paper [34] with the following additions:

1. several new applications are added: unorganized point cloud processing, implicit modeling, and protein backbone modeling,
2. a new class of algorithms is included under the geometric methods based on Reeb graphs,
3. extended discussion of the various curve-skeleton properties, and
4. the discussion and experimental results sections are completely rewritten.

2 DEFINITIONS—THE MEDIAL AXIS, MEDIAL SURFACE, SKELETON, AND THE CURVE-SKELETON

In 2D, the medial axis [21] of a shape is a set of curves defined as the locus of points that have at least two closest points on the boundary of the shape [72]. In the 3D case, the corresponding object is called the *medial surface* [48] because in addition to curves, it can also contain surface patches. A more illustrative definition of the medial axis/surface is given by the grass-fire analogy, where the boundary of a

• N.D. Cornea is with the Department of Electrical and Computer Engineering, Rutgers, The State University of New Jersey, 94 Brett Road, Piscataway, NJ 08854-8058. E-mail: cornea@caip.rutgers.edu.

• D. Silver is with the Department of Electrical and Computer Engineering, Rutgers, The State University of New Jersey, PO Box 909, Piscataway, NJ 08855-0909. E-mail: silver@caip.rutgers.edu.

• P. Min is with the American University of Rome, via Pietro Roselli 4, 00153 Rome, Italy. E-mail: patrick.n.min@gmail.com.

Manuscript received 4 Apr. 2006; revised 29 June 2006; accepted 12 Sept. 2006; published online 2 Jan. 2007.

For information on obtaining reprints of this article, please send e-mail to: tcvg@computer.org, and reference IEEECS Log Number TVCG-0039-0406. Digital Object Identifier no. 10.1109/TVCG.2007.1002.

Authorized licensed use limited to: Hochschule Mannheim. Downloaded on May 13, 2025 at 08:10:22 UTC from IEEE Xplore. Restrictions apply.

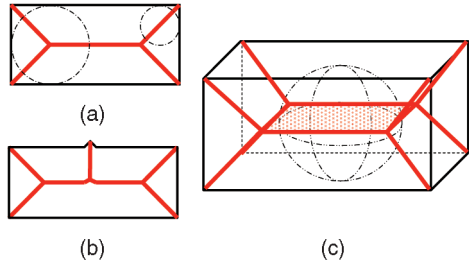


Fig. 1. A medial axis in 2D ((a) and (b)) and a medial surface in 3D (c) and a few examples of inscribed discs (2D) and ball (3D).

shape made entirely of dry grass is set on fire and the medial axis/surface consists of the loci where the fire fronts meet and quench each other. Fig. 1 shows the medial axis for a 2D shape and the medial surface of a 3D shape. Note that, in Fig. 1c, only one patch of the medial surface is shaded, but in fact the medial surface consists of many different patches bounded by the lines drawn in red and the edges of the box in black. The term *medial* is sometimes used to refer to the medial axis (in 2D) or the medial surface (in 3D).

The *skeleton* is defined as the locus of centers of maximal inscribed (open) balls (or disks in 2D) [72]. More formally, let $X \subset R^3$ be a 3D shape. An (open) ball of radius r centered at $x \in X$ is defined as $Sr(x) = \{y \in R^3, d(x, y) < r\}$, where $d(x, y)$ is the distance between two points x and y in R^3 . A ball $Sr(x) \subset X$ is maximal if it is not completely included in any other ball included in X [62]. The skeleton is then the set of centers of all maximal balls included in X . The process of obtaining a skeleton is called *skeletonization*.

Although the medial axis/surface and the skeleton are closely related, they are not exactly the same. Using the example given in [72], both the medial axis and the skeleton of a 2D ellipse are represented by a line segment, but the segment's end-points belong only to the skeleton and they do not belong to the medial axis. Given the small differences between the medial axis/surface and the skeleton, which only arise in the limit case, many authors use the terms medial axis/surface and skeleton interchangeably.

A major disadvantage of the medial surface (axis)/skeleton is its intrinsic sensitivity to small changes in the object's boundary due to the way it is defined [6], [31]. An illustrative example in 2D is shown in Fig. 1b where it can be observed how a small change in the object's boundary causes a large change in the skeleton.

In many applications, however, a concise representation of 3D objects with curve arcs or straight lines is desirable because of its simplicity. For example, animation traditionally uses an IK (inverse-kinematics) skeleton consisting of a small number of connected line segments representing, for instance, the torso, arms, and legs. Other applications, such as virtual navigation, also require a set of curve paths. This line-like representation of a 3D object is often called the *centerline* or the *curve-skeleton* [120] and is a simplified 1D representation of the original 3D object, consisting only of curves [38]. Fig. 2 shows curve-skeletons of several 3D objects.

In spite of its simplicity, until recently, there was no rigorous definition of a curve-skeleton. In a very recent work [38], Dey and Sun propose a possible definition of the

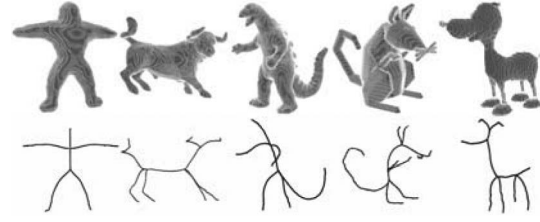


Fig. 2. Examples of curve-skeletons of different 3D objects.

curve-skeleton as a subset of the medial surface with the help of a medial geodesic function. However, as discussed in Section 3 under "Centered," defining the curve-skeleton as a subset of the medial axis may be too restrictive to be useful in some applications.

In this paper, we do not attempt to give a precise definition of the curve-skeleton. Our goal is to provide an analysis of the different aspects of this structure and lay the foundation for further efforts to formulate such a definition. For the purpose of this work, the curve-skeleton is defined as a simplified 1D representation of a 3D object.

2.1 The Discrete Case

The above definitions were formulated in continuous space. However, many of the applications that need skeletonization have discrete 3D data sets, such as those acquired using medical scanners. In discrete space, the definitions are analogous to the continuous case, but problems may occur because of discretization. For example, a maximal ball may touch the discrete boundary of an object in a single point in cases other than the limit case, such as the case of an object whose width is an even number of voxels. Since the diameter of the ball is always an odd number of voxels (assuming the center of the ball must be one of the voxels), the ball will be maximal when it touches the boundary on only one side. As a result, in order to include all centers of maximal balls, the discrete skeleton may be more than one image element (pixel or voxel) thick. Furthermore, resolution can cause a loss of detail for certain objects, such as merging, or even disappearance, of small features.

Some skeletonization algorithms work on continuous geometric data, others deal with discrete objects only. Conversion between these two representations can be performed using well-known algorithms: continuous geometric data can be transformed into a discrete representation by voxelization [116], while voxelized data can be converted into a geometric representation using a surface extraction algorithm [75]. In this paper, we will consider mainly the discrete case, but, for the sake of completeness, we also include references to methods operating on continuous geometric data.

In order to facilitate understanding of some of the definitions below, we need to introduce a few concepts from digital topology. For a complete review of digital topology, please see [62].

2.2 Short Review of Digital Topology

Let us consider the discrete space Z^3 . Each point p in this space is called a voxel (from volume element) defined by its three integer coordinates (p_x, p_y, p_z) . A voxel can be viewed

as a cube, having 6 faces, 12 edges, and 8 corners. Two voxels p and $q \in Z^3$ are 6-adjacent if they have a common face; they are 18-adjacent if they have a common face or edge and 26-adjacent if they have a common face, edge or corner. The set of 6-adjacent voxels to a voxel p is also known as the 6-neighborhood of p denoted by $N_6(p)$. Similarly, $N_{18}(p)$ and $N_{26}(p)$ are the 18 and 26-neighborhoods of p . By $N_6^*(p)$ we denote $N_6(p) \setminus \{p\}$; $N_{18}^*(p)$ and $N_{26}^*(p)$ are similarly defined.

An n -path is a sequence of voxels p_1, \dots, p_k with p_i n -adjacent to p_{i+1} , where n could be 6, 18, or 26. An n -connected component is then a set of voxels such that any two such voxels are connected by an n -path included in that component.

A 3D binary digital picture is described by the quadruple $P = (Z^3, m, n, B)$, where $B \subseteq Z^3$ is the set of black voxels representing the object in the picture (also known as object voxels), while $Z^3 \setminus B$ represents the background (white) voxels. The pair (m, n) specifies the object and background connectivity, respectively. In order to avoid topological paradoxes such as objects being both connected and disconnected [62], different values must be chosen for m and n ; common choices are (26, 6) and (18, 6).

A cavity is a background connected component surrounded by an object component (an empty space inside the object). While a mathematical definition of a tunnel is difficult [62], [105], [121], we can turn to intuitive examples of tunnels in well-known objects. The center hole of a donut or the empty space between the handle of a teapot and its main body are examples of tunnels. Algorithms to detect and characterize cavities and tunnels in a 3D object are described in [121].

3 CURVE-SKELETON PROPERTIES

In this section, we describe a set of desirable curve-skeleton properties, which we compiled from analyzing the literature on the subject and a number of different applications of curve-skeletons in computer graphics and visualization. These applications will be presented in Section 4.

For the following discussion, we will consider the discrete 3D case unless otherwise specified. We will use $Sk(O)$ to denote the curve-skeleton of a 3D object O . Our discussion includes the following curve-skeleton properties: homotopic, invariant under isometric transformations, reconstruction, thinness, centeredness, reliability, smoothness, component-wise differentiation, robustness, efficient to compute, and hierarchic.

Homotopic (topology preserving): The curve-skeleton should be topologically equivalent to the original object [62], [72], [105]. Preservation of topology can be stated simply as follows: *Two objects have the same topology if they have the same number of connected components, tunnels, and cavities.*

As pointed out in [62], the above formulation applied to an object O_2 derived from an object O_1 : “object O_2 preserves the topology of object O_1 ” is meaningful only if an additional constraint is added: Object O_2 is obtained from O_1 by only removing object voxels (no adding). Otherwise, object O_2 could end up having a completely new configuration, but still have the same topology. For example, by

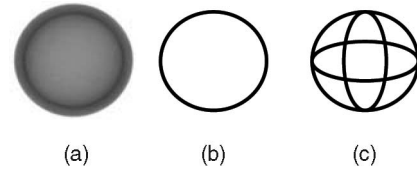


Fig. 3. Possible curve-skeletons of a hollow sphere.

adding object voxels to O_2 , O_2 may grow limbs where O_1 did not have them, but still have the same number of connected components, tunnels, and cavities. With this observation, the above definition of topology preserving is meaningful in the context of skeletonization, where the skeleton S is a subset of the original object O .

Of course, we cannot have cavities in a 1D curve, so, in a strict sense, a curve-skeleton cannot preserve the topology of an object with cavities. To accommodate objects with cavities, a relaxed definition of topology preserving can be formulated using the loops of a 1D curve [107]: The curve-skeleton should have at least one loop around each cavity of the original object. Think of a hollow sphere: The curve-skeleton can be just a circle—a single loop—or many circles in different orientations but all surrounding the same cavity. The latter version may better convey the true shape of the object as shown in Fig. 3, but, clearly, this is application dependent.

However, tunnels in the original object also create loops in the curve-skeleton. Thus, we will reformulate the *relaxed definition* as follows: *The curve-skeleton S preserves the topology of the original object O in a relaxed sense if it has the same number of connected components and at least one loop for each tunnel and cavity in the original object.*

This formulation has the same constraint as the previous one with respect to configuration. Of course, if the object does not have any tunnels or cavities, the curve-skeleton should have no loops at all.

Such a definition could be useful for iconic/abstract representation of objects, where all topological features must be represented by the curve-skeleton. In addition, it could be used to develop an algorithm that checks the homotopy property of a curve-skeleton. The loops in a curve-skeleton can easily be determined by performing a depth-first search on the curve-skeleton, while tunnels and cavities of a 3D object can be determined using the method described in [121].

Invariant under isometric transformations: Given an isometric transformation T (a transformation in which the distances between points are preserved), the curve-skeleton of the transformed object $T(O)$, denoted by $Sk(T(O))$, should be the same as the transformed curve-skeleton of the original object. Formally, the invariance criterion is given by: $T(Sk(O)) = Sk(T(O))$. This property is important for matching applications where the curve-skeleton is used as a shape descriptor. In such applications, it is common to have similar objects in different orientations that nevertheless still need to be matched and, for this reason, the shape descriptor must be insensitive to object orientation.

Reconstruction [43], [88] refers to the ability to recover the original object from the curve-skeleton. Given the relation of curve-skeleton to the medial surface/skeleton

and the definition of the skeleton as the set of centers of maximal inscribed balls, an obvious choice of reconstruction method is to compute the union of maximal inscribed balls centered at each curve-skeleton point (or discrete medial surface point) [21], [44]. The radius of each ball is given by the “distance transform value,” which specifies the distance to the closest point on the boundary of the object. If we denote the reconstruction operation by $Rec(skeleton)$, then accurate reconstruction means that $Rec(Sk(O)) = O$.

A 3D object can be completely reconstructed from its medial surface/skeleton representation by computing the union of maximal inscribed balls. This property has an immediate application in shape compression and volume animation [44]. However, in general, when using the ball-growing approach, accurate reconstruction is not possible from the curve-skeleton alone since it is only a subset of the medial surface. That is, in general, $Rec(Sk(O)) \neq O$. To test the degree of reconstruction (accuracy) possible from a given curve-skeleton, every point must be equipped with the distance transform value determined in the original object. Then, the difference volume $O - Rec(Sk(O))$ will provide a quantifiable measure of the ability to reconstruct the object.

Reconstruction can be improved by storing more information in each curve-skeleton point and/or by increasing the number of branches in the curve-skeleton. For instance, one could store the three radii of a maximal inscribed ellipsoid and replace ball-growing with an ellipsoid-growing algorithm. Alternatively, one could simply extract a curve-skeleton with many more branches that should reconstruct more of the original object using the classic ball-growing approach.

Intuitively, the ability to reconstruct an object from an abstraction, such as the curve-skeleton, might seem to be an indication of the quality of that shape abstraction for shape analysis tasks. After all, if the degree of reconstruction is very low, it means the curve-skeleton does not capture much of the original object. However, recent work has shown this is not the case. In [113], some of the best performing shape descriptors for shape matching cannot reconstruct the object at all. Additionally, in [33], the curve-skeleton showed good results for retrieving similar shapes from a large database of general 3D objects.

Thin: Curve-skeletons should be one-dimensional, that is, at most one voxel thick in all directions, except at joints where the curve-skeleton might become thicker to ensure connectivity between the different branches.

We can distinguish three types of curve-skeleton points [22]: regular points on a 1D curve that have exactly two neighbors, end-points of a curve that have exactly one neighbor, and junction points (where curves meet), which can have three or more neighbors. The thinness property can easily be checked if the junction points are known in advance. Some curve-skeletonization methods directly identify junction points [32], [68]. If junction points are not known in advance, they have to be identified with another method.

Thinness and reconstruction are two conflicting properties. Even for objects whose medial surface actually contains only curves (like tubular objects), a one-voxel thick curve-skeleton may not contain all the necessary maximal balls to

accurately reconstruct the object (remember that a discrete medial surface/skeleton is usually more than one voxel thick owing to the discrete nature of the object).

Centered: An important characteristic of a curve-skeleton is its centeredness within the object. To achieve perfect centeredness, it is required for the curve-skeleton to lie on the medial surface since the medial surface is centered within the object. This criterion alone is not enough and, in addition, we require the curves to be centered within the medial surface patches they belong to [38], [107]. In shape compression and some scientific applications such as vortex core extraction [11], exact centeredness of the curve-skeleton may be essential. However, in most cases, exact centeredness of the extracted curve-skeleton is not required or desired. Given the well-known sensitivity of the medial surface to small perturbation on the boundary of the object [6], [31], constraining the curve-skeleton to lie on the medial surface may make it sensitive to such changes as well.

Instead of exact centeredness, an approximate centeredness (what we call relaxed centeredness) is probably enough for many applications such as virtual navigation or animation. For example, in a virtual colonoscopy application, reliability (see below) and smoothness of the navigation path are more important than exact centeredness [61]. We still want the navigation path to be close to the center of the object, but being one or two voxels away from the exact center is not a big concern.

One possible way to quantify the centeredness of a curve-skeleton is to seed a number of uniformly distributed radial rays in a plane normal to the direction of the curve-skeleton at each one of its points and measure the distance to the boundary along each of these rays. Centered points should have the same distance to the boundary along each pair of opposite rays.

Reliable: *Reliability* [52], [61] refers to the property of the curve-skeleton that every boundary point (point on objects surface) is visible from at least one curve-skeleton location. In other words, for any boundary point, there exists a straight line connecting it to a curve-skeleton point that does not cross any boundary. The term reliable is used in relation to virtual endoscopy where it ensures that the interior organ surface is fully (reliably) examined by the physician performing the virtual procedure.

A brute-force algorithm to test the reliability of the curve-skeleton checks the visibility of each boundary point with a straight line to every curve-skeleton point. Boundary points that cannot be connected without intersecting the surface are not visible. Efficient visibility computation can be done following the solutions from [52].

Junction Detection and Component-Wise Differentiation: The curve-skeleton should be able to distinguish the different components of the original object, reflecting its part/component structure. This says that the *logical components* of the object should have a one-to-one correspondence with the logical components of the curve-skeleton (which are curve arcs).

There is no rigorous definition of logical components of a 3D shape, although several attempts have been made. For example in [122], meaningful components are defined as

components that can be perceptually distinguished from the remaining object. In [60], the component structure of a 2D shape is defined using a combination of substance and connection measures computed around junction points of the medial axis using “visual conductance.” The Reeb graph can also be used to identify object components (see Section 5.3), but its definition is dependent on the choice of the generating function.

As long as the curve-skeleton has identifiable joints or junction points, a partitioning of the original object can be performed to produce a one-to-one correspondence between the different components in the object and the curve-skeleton (for use in animation or mesh decomposition, for example).

We would like to make a clear distinction between curve-skeletonization methods that can identify the joints or junction points *before* or *during* the extraction of the curve-skeleton and the methods that extract these joints *after* the curve-skeleton is produced. If the resulting curve-skeleton is only one voxel thick in all directions, detection of joints as a postprocessing step is trivial: They are the points having more than two neighbors. It is much harder to identify these junction points before extracting the full curve-skeleton. When extracting joints as a postprocessing step, the identified joints are as good as the underlying curve-skeleton and no claims can be made about their significance or stability with respect to the original shape. Joints identified as a first step of the curve-skeletonization process carry more significance simply because they must be related to some intrinsic property of the original object since the curve-skeleton is constructed afterward. It is the difference between the joints being a by-product of the curve-skeleton or being its source.

Component-wise differentiation is different from homotopy in that it deals with logical perceptual components of a single connected object while the latter is concerned with geometrical connected components forming different objects.

Checking whether a curve-skeleton satisfies this property is a difficult task because the definition of object component is not precise enough, involving human perception, which is inherently subjective. However, application specific definitions could be used for such a purpose: For example, a curve-skeleton suitable for animation tasks would be one that has a separate branch for each of the limbs and/or parts of limbs of the model being animated. For simple models, the limbs can easily be defined manually or even automatically in special cases. A hierarchical skeleton (see below) could be useful here to produce levels of meaningful components.

Connected: This is a consequence of homotopy. If the curve-skeleton corresponds to a single connected object, then, by maintaining the topology of this object, the curve-skeleton would have to consist of a single connected component itself.

Robust: As shown in Fig. 1b, the medial axis is very sensitive to small changes in the boundary. A desirable property of the curve-skeleton is to exhibit weak sensitivity to noise on the boundary of the object, that is, the curve-skeletons of a noise-free object and the curve-skeleton of the same object with noise should be similar. A robust curve-skeleton cannot be perfectly centered. Exact centeredness

would constrain the curve-skeleton to the medial surface, which is extremely sensitive to boundary perturbations.

Smooth: Smoothness is not only an aesthetic property, but is actually useful in some applications. For example, in virtual navigation, which uses the curve-skeleton as a camera translation path, the path should be as smooth as possible to avoid abrupt changes in the displayed image.

We can define the smoothness of a curve segment as the variation of the curve tangent direction as we move along the curve. More precisely, we can measure the angles between tangent directions at successive locations along the curve and take the standard deviation of these values as a measure of variation. To ensure smooth navigation, the variation in tangent directions as we move from one point to the next along a segment should be as small as possible.

Hierarchy: Because the curve-skeleton is an approximation of the complex components of an object, the curve-skeletonization process and the curve-skeleton itself should reflect the natural hierarchy of these complexities [35], [59]. A hierarchical approach is useful because it can generate a set of curve-skeletons of different complexities that could be used in many different applications. In a strict hierarchy, the curve-skeleton at a certain level in the hierarchy contains all curve-skeletons from the layers below as subsets. Such a strict hierarchy is useful in applications using different resolutions during processing, such as multiresolution matching.

A different kind of hierarchy, mostly useful in animation, consists of defining the hierarchical relations between parts of the same object. For example, the torso is the root of the hierarchy with the limbs and head as its children. This kind of definition is useful for animation as it allows a whole tree of object components to be manipulated by manipulating the root of the tree. For example, the entire arm tree (arm, forearm, hand, and all fingers) can be moved by manipulating the shoulder joint.

There are two other criteria that relate to the *algorithm* used to compute the curve-skeleton. First, the algorithm should be **efficient**: many applications need real-time computations. Second, some algorithms can handle **point sets** (i.e., where the connectivity is not specified and there is no inside/outside information) or other object specifications, not just a voxelized representation.

Not all properties described above are essential to all types of applications. Furthermore, some of the properties may be conflicting, such as thinness and reconstruction or robust and centered.

As a result, various algorithms that extract curve-skeletons usually satisfy only a subset of these properties, depending on the application.

In the next section, we describe a number of applications that use curve-skeletons.

4 USES OF CURVE-SKELETONS IN VISUALIZATION

Since they were first introduced, curve-skeletons have found uses in many areas (e.g., image processing, visualization, animation, etc.). In this section, we present an overview of some of these applications. This overview is not comprehensive, many other applications exist.

One of the first uses of the curve-skeleton was in **virtual navigation** [96], [130], exploiting its centeredness property to generate collision-free paths through a scene or through an object. Given a scene composed of 3D objects, the curve-skeleton of the background gives a collision-free path through the scene. In **virtual endoscopy**, curve-skeletons are used to specify collision free paths for navigation through human organs. Traditional endoscopic methods are invasive and often uncomfortable to patients. A virtual endoscopy system can produce images similar to those obtained using the traditional technique but in a noninvasive way. After imaging, the organ is “skeletonized” and a virtual camera is translated along this curve-skeleton path allowing the inspection of the respective organ. Clinical applications include colonoscopy [54], [61], bronchoscopy [96], angioscopy [12], and others. A *reliable* navigation path ensures the interior organ surface can be fully examined by the physician performing the virtual procedure [52], [130].

In **traditional computer graphics**, skeletons are used extensively to specify animation [1], [19], [83]. These skeletons (sometimes referred to as IK-skeletons) control the polygonal representation of the character being animated. Surface polygons are attached to, and manipulated through, this simple stick-like figure. While most of the IK-skeletons are specified by an animator, recently there have been methods to compute the skeleton and the “skinning” (polygon correspondence) automatically [20], [73], [122], [129]. A simplification of the curve-skeleton can be successfully used as an IK skeleton by replacing curve arcs with straight lines. Volumetric objects can also be animated and manipulated using the same type of paradigm [44]. In [28] and [125], the IK skeleton is automatically detected from a sequence of volume data sets.

Surgical planning and radiation treatment require accurate extraction (**segmentation**) and **quantification** of specific anatomical structures from CT (computed tomography), MRI (magnetic resonance imaging), MRA (magnetic resonance angiogram), or ultrasound data. This is especially true for blood vessels and nerve structures. Since these structures have a characteristic tubular shape, methods aimed specifically at extracting the centerline of such tubular objects from medical images have been developed [9], [10], [40], [41] using field-specific knowledge (intensity variation of the blood vessels, connectivity) or simply the volumetric representation of the vessels [89]. The centerline can also be used to aid in other image processing operations such as edge detection and segmentation [98], [99]. Other uses include curved planar reformation (flattening) [55], [58], detection of stenosis [90], [115], aneurisms or vessel wall calcifications [117], deforming volumes: unwinding convoluted objects to allow a more efficient inspection of the overall structure or to remove occlusion (e.g., colon straightening [114]).

A common operation in medical imaging is the **registration** of two images from the same patient taken with different modalities (MRI, CT, MRA). Registration is performed by aligning some structures that are visible in both images. One approach is to reduce the dimensionality of the problem by extracting the skeleton of the structure from both images and then aligning the skeletons [10], [42], [98].

Another application is **matching** of 3D objects: Given a query object, the task is to find similar or identical objects in a database by using the curve-skeleton [27], [33], [53], [118]. If the curve-skeleton can differentiate the part structure of the original object, part matching is also possible, where only parts of the objects are matched against the query. In addition to matching, it directly provides registration of the part in the whole object [33], [118].

Shape metamorphosis (**morphing**) is the process of generating smooth transitions between two shapes, creating the impression that one object is being smoothly transformed into another. One of the most difficult tasks in generating a successful metamorphosis is determining the correspondences between the two shapes used to drive the interpolation process. Various trade-offs are made between allowing the user full control over the process (thus, turning it into a mostly manual process) and completely automating the correspondence finding algorithm. The curve-skeleton can be used in this context for its simplicity, allowing the user to quickly specify correspondences on the skeletons or enabling matching algorithms to find correspondences more efficiently. Additionally, the interpolation process can be performed directly on the skeleton [18], [64], [136].

Decomposing a polygonal mesh into components is desirable for applications that treat objects as a sum of components. Such a decomposition can be assisted by using the curve-skeleton if it has the ability to distinguish the components of the original object [28], [70]. In [59], an inverse approach is taken, where a 1D skeleton is extracted using the mesh decomposition results. Related geometric uses of skeletons include **surface reconstruction** [4], [127] and **mesh repair** [68]. In [49] and [134], the more challenging problem of **segmenting and quantifying an unorganized point cloud** is approached using a curve-skeleton.

In [119], the curve-skeleton is used to define a “skeletal dimensional reduction” for the **CAD** field. It is also shown how such a representation can be used to reduce boundary value problems over complex solids to lower-dimensional problems over the skeleton. Skeletons have also been used to improve the efficiency of collision detection of volumetric objects [45] or in surgical simulations [131] and as a general data structure for graphical objects [100]. In [55], a curve-skeleton in combination with convolution surfaces is used for **implicit modeling** of 3D objects.

In **analysis of scientific data**, curve-skeletons are used to make complex topologies more easily understandable. Furthermore, skeletons can be used for reduced modeling and to explain simple physical phenomena. Examples include **plume visualization** [108], **vortex core extraction** [11], **feature tracking** [128], **protein backbone modeling** [50], [69], and many others.

The previous discussion is by no means exhaustive, but gives a sample of popular uses of skeletons in visualization. Some applications have extra data available to help in the curve-skeletonization process such as velocity fields in the case of vortex core extraction [11] or blood flow data in the case of vessel tracking (e.g., [9], [10], [40]), while others use only the 3D object. In this paper, we concentrate on the more general problem where extra information is not available.

5 ALGORITHM CLASSES

There are many different skeletonization algorithms for both 2D and 3D. Although some of the 2D algorithms reportedly extend to 3D, we restrict our discussion to algorithms explicitly designed for 3D. The discussion below reviews general 3D curve-skeletonization algorithms, i.e., the generation of a 1D curve-like representation from a 3D object. However, for completeness we do include some medial surface algorithms since these medial surfaces could be further reduced to a curve-skeleton [107], [120]. Unless otherwise stated, we consider the 3D objects to be represented by voxels on a regular grid.

A commonly used classification scheme present in the literature divides the skeletonization algorithms into the following classes [80], [123]: topological thinning (grassfire propagation), distance transform-based (ridge detection) and Voronoi diagram-based. However, many of the surveyed methods that produce curve-skeletons use pieces from several classes listed above to obtain a curve-skeleton. For example, there are thinning algorithms that use the distance field information to determine the thinning order or some distance field methods which use thinning to prune the skeleton. Instead, we initially categorize the algorithms based on the underlying implementation of the initial step into the following classes:

1. thinning and boundary propagation,
2. distance field-based,
3. geometric, and
4. general-field functions.

5.1 Thinning and Boundary Propagation

Thinning methods attempt to produce a curve-skeleton by iteratively removing voxels from the boundary of an object until the required thinness is obtained. All thinning algorithms operate in the discrete (voxel) space and rely on the concept of **simple point**, introduced by Morgenthaler in 1981 [84]. A simple point [13], [62], [63] is an object point (voxel) that can be removed without changing the topology of the object (see [62] for a complete review of digital topology). An important property of simple points is that they can be locally characterized, that is, one can determine if a voxel is simple or not by only inspecting its local neighborhood, making thinning algorithms much more efficient.

The thinning process starts from the object's boundary and continues inward until no more simple points can be removed. At every iteration, each boundary voxel is tested against a set of topology preserving conditions and possibly removed. The conditions are usually implemented as *templates* (or *masks*) of size $3 \times 3 \times 3$ or larger. The center of a mask is placed on the voxel being tested and covers its entire local neighborhood. Each of the voxels in the mask has a value of "0," "1," or "don't care." A value of "0" must match a background voxel, a value of "1" must match an object voxel, while a "don't care" can match either a background or an object voxel.

However, removing all simple points from the object produces excessive shortening of the curve-skeleton branches. This is because all the end-points of curve-skeleton curves are simple points themselves (i.e., removing

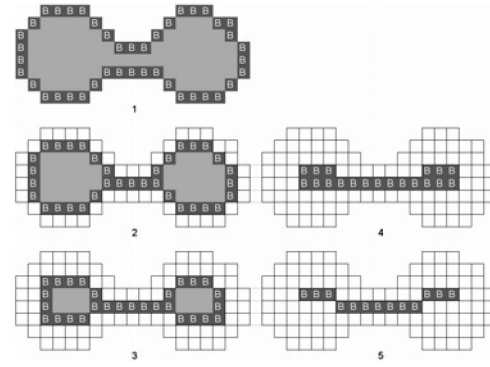


Fig. 4. The thinning process on an example 2D shape. Boundary points are marked "B" at the beginning of each iteration and then removed if they are simple.

them will not change the topology of the object). Additional conditions are used to prevent removal of *surface* or *curve endpoints* in order to maintain the geometrical properties of the object.

There are several subclasses of thinning methods based on how simple points are detected and considered for removal.

Directional thinning methods remove voxels only from one particular direction in each pass (for example, North, South, Up, Down, East, and West) using different numbers of directions and conditions to identify endpoints [28], [48], [66], [74], [94], [95], [126]. These methods are sensitive to the order in which the different directions are processed and the resulting skeletons may not be centered within the object.

Subfield sequential thinning methods divide the discrete space into several subsets named subfields and, at each subiteration, only voxels belonging to one of the subfields are considered for deletion. A different number of subfields can be used in 3D: two [77], [78], four [76], [79], or eight [13], [104]. For example, in the 2-subfield approach [77], two voxels are in the same subfield if they share an edge.

Fully parallel [24], [39], [76], [82] algorithms consider all boundary points for deletion in a single thinning iteration. In order to maintain topology, the neighborhood that needs to be inspected when deciding whether a voxel is deletable or not must be extended past the immediate 26 neighbors.

Some thinning methods produce a surface-skeleton in the first stage and continue to thin until a one voxel thick curve-skeleton is obtained [24]; others directly produce a curve-skeleton, which usually involves using a different set of templates. Additionally, a 2.5D algorithm aimed specifically at reducing a surface skeleton to a curve-skeleton by thinning is described in [120].

Fig. 4 shows the thinning process on a 2D shape. At each iteration, the boundary points are marked with "B" and, if they are simple points, they are removed before the start of the new iteration. At the last iteration, no points can be removed.

Most thinning algorithms are designed and proven correct for a specific (6, 8, or 26) connectivity (see Section 2.2). The correctness proof deals with preservation of the original objects topology.

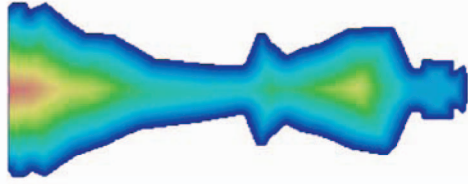


Fig. 5. A color-coded slice of the distance field of a 3D shape.

5.2 Using a Distance Field

The *distance transform* or *distance field* is defined for each interior point P of a 3D object O as the smallest distance from that point to the boundary $B(O)$ of the object:

$$D(P) = \min_{Q \in B(O)} (d(P, Q)), \text{ where } d \text{ is some distance metric.}$$

Various distance functions can be used such as the Euclidean distance or an approximation such as the $\langle 3, 4, 5 \rangle$ chamfer metric [23]. A distance field can also be approximated using fast marching methods [103], [111], [123], [124]. Fig. 5 shows the color-coded distance field values on a slice of a 3D chess piece shape. The color map ranges from blue for small distance field values to red for large values.

The ridges of this distance field correspond to voxels that are locally centered within the object. Most of the methods in this class attempt to find these voxels. These are candidates (from the larger pool of object voxels) for curve-skeleton points. Several approaches are used to find candidate voxels: In **distance ordered thinning** approaches [36], [44], [101], a thinning algorithm uses a priority function computed over the distance field to select candidate voxels for removal; **gradient searching** [17] involves detecting neighborhoods of nonuniform gradient and flagging those points as candidate voxels; **divergence computation** is used in [25] as the priority function and a user-defined threshold blocks the removal of simple points with lower divergence value; **parameter controlled thinning** involves comparisons between the distance field value at a voxel and the average distance field value of its neighbors [43]. In the **geodesic front propagation** algorithm described in [96], candidate voxels are local maxima of the distance field detected on the geodesic front propagating from the root of the 3D object. Thresholding the **bisector angle** [81] or **surface shrinking** along the gradient of the distance field [109] are other methods for producing candidate voxels. In **voxel coding** approaches [138], the distance field, which measures the distance from the boundary, is combined with a distance-from-a-source field to generate a set of candidate voxels. In [103], the distance field is combined with several distance-from-a-source fields generated using fast marching fronts of various speeds.

The set of candidate voxels is usually a fairly large set and the next step is to somehow prune it down to a manageable size. Various criteria can be used in this step to “weed out insignificant extreme points” [129], such as: **sphere coverage** of a path tree [129], **boundary visibility** from candidate voxels [52], **thinning** [123], or **clustering** [118].

After the pruning step, the remaining voxels are usually disconnected and the final step involves reconnecting them in order to produce a set of 1D curves. For connectivity,

most algorithms use **minimum spanning trees** [118], [130], **shortest paths** [52], [123], [129], or other graph algorithms. In [137], an “**LMpath**” defines the connectivity of local maxima clusters, while, in [17], the gradient of the distance field is used to connect new voxels to the already connected set. In [43], a recursive **midpoint subdivision** algorithm identifies the closest candidate in between two already selected voxels to build a 1D path between two user-selected end points.

Most of the distance field-based algorithms have these three steps: 1) find ridge points (local maxima, saddles), 2) prune, and 3) connect.

Some methods may first connect the set of candidate voxels and then prune it by extracting shortest paths from the connected set [17], others explicitly maintain the connectivity while pruning, combining Steps 2 and 3. An alternative is provided by the fixed topology skeleton, which is a set of a fixed number of connected active contours driven by the underlying distance field [47].

A 2.5D method presented in [107] starts from a medial surface representation of 3D objects and, for each surface point, it computes the distance to the boundary of the surface. The candidate voxels (centers of maximal geodesic discs) are connected using gradient guided path growing.

The distance field-based methods can accurately extract the medial surface; however, they cannot extract a curve-skeleton from arbitrary objects without employing additional techniques to prune the medial surface. For example, all voxels on the medial sheet shaded in Fig. 1c have the same distance field value. Therefore, some sort of pruning must be used to simplify it to a set of curves.

The main advantage of these methods is that computation of the distance field is very fast and it is usually needed by the application afterward. Furthermore, for tubular objects, the distance field approach works very well.

5.3 Geometric Methods

Geometric methods apply to objects represented by polygonal meshes or scattered point sets.

A popular approach is to use the **Voronoi diagram** [26], [91], [92], [93] generated by the vertices of the 3D polygonal representation or directly by a set of unorganized points [4], [7], [86]. The Voronoi diagram represents a subdivision of the space into regions that are closer to a generator element (a mesh vertex in the case of a 3D model) than to any other such element. The internal edges and faces of the Voronoi diagram can be used to extract an approximation of the medial surface (skeleton) of the shape. A curve-skeleton can be extracted from this medial surface approximation by pruning it to a 1D structure. In [132], a number of “domain balls” (nonintersecting maximal inscribed balls) are detected, whose centers are later connected into a curve-skeleton using surface connectivity information. In [38], a curve-skeleton is computed by eroding the Voronoi diagram representation using the medial geodesic function as a priority function.

Other methods attempt to directly identify the medial surface of a 3D polyhedral by tracing the seams of the medial surface: [37], [102], [110], [112], [135]

Cores and M-reps [30], [85], [97], [100] are also medial-axis/surface approaches. A core is a locus in a space whose

coordinates are position, radius, and associated orientations. The location of the core represents the middle of the figure and the spread of the core represents the width of the figure. M-reps are a generalization of the Core concept. The M-rep models the medial surface using a “web” of connected atoms. Each atom has a position and additional information describing the shape locally, such as: width, a local figural frame (which implies the figural directions), and an object angle between opposing corresponding positions on the implied boundary.

A similar structure is the **shock scaffold**, which relies on the concept of contact spheres [67], [68] and represents the medial axis/surface by a set of shock curves, defined as the intersection of medial surface sheets (not the curve-skeleton).

The methods described above can be labeled as medial-axis/surface-based. The main disadvantage of medial-axis-based geometric methods is their sensitivity to noise. For example, Amenta et al.’s *power shape* [4] contains a large number of unwanted branches that need to be pruned to extract a simpler skeleton [7]. Furthermore, the above described methods are more computationally intensive than the thinning/distance field-based methods and produce a medial surface, not a curve-skeleton. Nevertheless, we mention these methods here because reducing a surface skeleton to a curve-skeleton is possible using the curve thinning algorithms [107], [120].

There are other geometric methods that avoid the medial axis altogether, and produce a 1D skeleton directly.

Reeb graph-based shape descriptors, with roots in Morse theory, are 1D structures encoding the topology and geometry of the original shape. The Reeb graph captures the topology of a compact manifold by following the evolution of the level sets of a real-valued function defined on the respective manifold. More formally [14]:

Given a real-valued function f , defined on a compact manifold M , $f : M \rightarrow \mathbb{R}$, the Reeb graph is the quotient space of $M \times \mathbb{R}$ defined by the equivalence relation \sim given by:

- $(x_1, f(x_1)) \sim (x_2, f(x_2))$ iff $f(x_1) = f(x_2)$ and
- x_1 and x_2 are in the same connected component of $f^{-1}(f(x_1))$ (or $f^{-1}(f(x_2))$).

In other words, the equivalence classes defined by \sim consist of the connected components of the level sets of f . Nodes in the Reeb graph correspond to the critical points of the function f (i.e., points where the gradient of f is zero) and edges of the graph represent connections between critical points. According to Morse theory, the object changes its topology only in connection to the critical points of f , so the edges of the Reeb graph can be thought of as representing different components of the object and the nodes can be regarded as connections between these different components.

The Reeb graph is not a curve-skeleton: It is not even defined in the same space as the original object. However, an embedding of the Reeb graph into 3D space can be attempted by mapping each edge into a sequence of 3D points defined for example as the centers of the successive level-set contours associated with the respective edge. This defines a curve-skeleton for the original object [65]. Fig. 6 shows an example for a 2D shape.

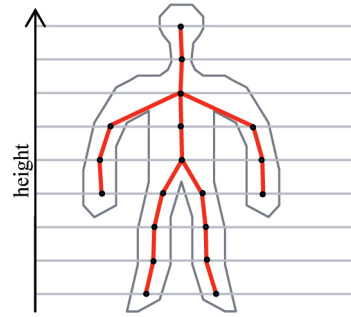


Fig. 6. An embedding of the Reeb graph into the original image space. Each node is taken to be the centroid of its corresponding contour.

Extensions of the Reeb graph to polygonal meshes have been proposed such as the Extended Reeb Graph [8], [14] or the Discrete Reeb Graph [134]. The choice of the real-valued function f distinguishes the various algorithms. The height function used in [8], [134] is sensitive to the object’s orientation. The Euclidean distance from a point in space, usually the barycenter of the mesh (used in [15], [16]), does not depend on the orientation of the object, but it is affected by changes in the underlying mesh as a result of articulated motion. Using the geodesic distance instead of the Euclidean distance solves this problem, but the source point must now be a vertex of the mesh. The shortest path distance to a given source point [65], [127] is sensitive to the selection of the source point.

Alternatively, the integral of the geodesic distance to all the points on the surface [53] is less sensitive to small perturbations due to noise. Using the shortest geodesic distance to a set of curvature maxima on the surface [86] introduces the problem of robustly and automatically detecting relevant curvature maxima on the surface. A similar approach is given in [28], where curve-skeletons generated from multiple extremities are combined into a final one.

Other geometric methods do not rely on a function defined on the manifold to produce a curve-skeleton. Li et al. [70] construct a line segment skeleton by collapsing edges in length order (shortest first). This method is sensitive to the mesh tessellation. Katz and Tal [59] first decompose a mesh surface into segments using clustering and then use this segmentation to construct a skeleton (joints are represented by centered vertices at the boundary between different patches). In [71], the curve-skeleton and the mesh decomposition processes are interrelated. The curve-skeleton of each component is obtained by connecting the principal axis with the centroid of each opening (connection to other components). In turn, the mesh decomposition process decides whether or not to further decompose an existing component by evaluating the quality of its curve-skeleton.

5.4 General-Field Functions

Various types of fields generated by functions other than the distance transform can also be used to extract curve-skeletons. Included in this class are generalized **potential field** functions [3], [32], [35], where the potential at a point interior to the object is determined as a sum of potentials generated by the boundary of the object. In the discrete case

[35], the boundary voxels are considered point charges generating the potential field. The **electrostatic field** function is used in [2], [51] to generate a potential inside the object. The **visible repulsive force** function presented in [73] and [133] is a special case of the generalized potential field function used in [3], [32] and [35]: the Newtonian repulsive force. However, here, the visibility of boundary elements from an interior point is taken into consideration by computing the intersection with a number of sampling rays originating at the current position on the path. In [80], the combination of **radial basis functions** originating at the mesh vertices is used to define a field inside the object.

The curve-skeleton is extracted by detecting the local extremes of the field and connecting them. A force following algorithm can be used for connectivity, using the mesh vertices [133], the convex corners of the mesh [3], [32], the significant corners along equipotential contours [2], or the critical points of the vector field [35] as “seed” points. Another possibility is the use of an active contour to detect the final location of the curves connecting the extremes in the field [80], [132].

Detection of local extremes can be achieved explicitly by looking for critical points over the entire underlying vector field [35] or detecting the local maxima along equipotential contours [2]. The other methods directly use the force-following algorithms starting at other seed points, using the fact that the force-following algorithm stops when it reaches an extreme.

The main advantage of these functions over the distance field is that they can produce nice curves on medial sheets where the distance field is constant. This is because they take into account larger boundary areas, not just the distance to the closest point on the boundary. This also creates an averaging effect that makes these algorithms less sensitive to boundary noise. However, they are much more expensive to compute.

Resolution of the voxel grid also affects the field functions, which tend to be more sensitive to noise in thin regions of the object because significant contributions toward the final field value at an interior point come from fewer boundary voxels.

Fig. 7 shows the vectors of the repulsive force field of a 2D shape.

Another disadvantage of the general field functions is their numerical instability since the computations usually involve first or even second order derivatives.

6 DISCUSSION

The classification proposed in the previous section was based on the first steps of the implementation, but we would like to emphasize that many methods use concepts from two or more different classes to finally produce a curve-skeleton. The distance field based thinning procedures described in [36], [44], [101] initially compute a distance field over the object, but then employ thinning to extract a curve-skeleton, using the distance field as a priority function to select voxels that will be removed next. In [132], a control skeleton named the domain connected graph (DCG) is extracted in several steps involving Voronoi diagrams, Reeb graphs, and repulsive force fields.

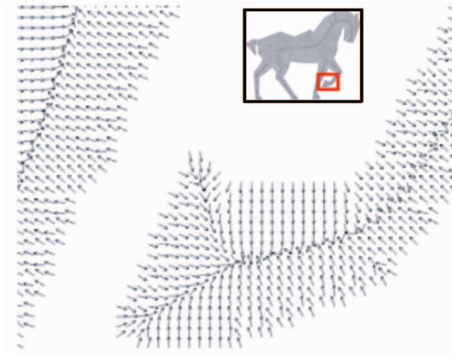


Fig. 7. The repulsive force field of a 2D shape. The inset shows what region of the force field was magnified.

Additionally, note that concepts from different classes are actually closely related. For example, we could relate the single-source distance field (voxel-coding) algorithms [103], [138] with the Reeb-graph approaches [65], [127] since the minimum distance search employed in single-source distance field methods actually involves the traversal of the level sets of a distance function defined on the object (in this case, on the full solid 3D object rather than only its surface). Surface shrinking along the gradient of the distance field [109] is also similar to surface thinning (removing layers from the boundary of the object until no more layers can be removed).

In the remainder of this section, we discuss the described curve-skeletonization methods in terms of the properties presented in Section 3.

6.1 Homotopy

Homotopy is ensured by the thinning methods because only voxels that do not change the object topology (the *simple points*) are removed.

Since distance field methods do not produce a curve-skeleton directly, topology preservation depends on the subsequent pruning and connectivity steps. Clearly, connectivity algorithms based on minimum spanning trees do not preserve topology because they are not able to create loops.

The *power shape* [4] represents a topologically correct approximation of the medial axis. However, care must be taken in the subsequent simplification steps so the final curve-skeleton maintains the topology of the power shape. The *Reeb graph* is an accurate representation of the object topology. However, in the discrete case, an important factor is the sampling frequency used to construct the level sets. If the sampling is too sparse, small topologically relevant features may be lost, while a too dense sampling may take too long to compute. An adaptive sampling solution is proposed in [8]. The *edge-contraction methods* described in [70] specifically maintains the topology of the intermediate shapes. The curve-skeleton extraction method based on mesh decomposition described in [59] cannot preserve topology because the extracted curve-skeleton is always a tree. This approach was taken because the curve-skeleton was used for animation where loops are generally not desirable.

Curve-skeletons produced by algorithms employing force-following on generalized fields could be disconnected even for objects with a single connected component, due to numerical errors affecting the integration steps in regions with insufficient resolution [35]. Using an active contour approach [32] can preserve connectivity of curve-skeleton segments, but the initial connectivity of the sinks based on minimum distance seems arbitrary and may not correspond to the topology of the original object. Furthermore, due to numerical errors and resolution of the voxel grid, potential field algorithms can create loops in the curve-skeleton that do not correspond to a tunnel or hole in the original object [35].

6.2 Invariant under Isometric Transformations

Directional thinning methods are sensitive to the object orientation. The final result (end points, number of branches, and their location) depends on the order in which the different directions are processed. Distance field, Voronoi-based, and general field methods do not depend on object orientation.

Reeb graph-based methods can be sensitive to object orientation depending on the function chosen to extract the level sets. For example, the height function [8], [134] is dependent on orientation, while the distance to the barycenter of the mesh [15], [16] is not.

In all cases involving discrete representations of objects, the finite resolution of the voxel grid produces small errors when objects are transformed. As a result, even though curve-skeletonization algorithms themselves are not sensitive to object orientation, the input data itself is already adversely affected by the transformation. Such small discretization errors show up on the boundary of the transformed object and their effect on the resulting curve-skeleton is similar to that of surface noise.

6.3 Reconstruction

The skeleton (medial surface) of a 3D object captures local symmetries present in the object through different types of elements: Surface patches in the skeleton represent symmetric plate-like regions of the original shape, while individual curves in the skeleton correspond to cylinder-like (tubular) shape regions.

It should be obvious that, regardless of the method used to compute it, a complete and accurate reconstruction of the original object is not possible from the information retained in a curve-skeleton alone when using the simple ball-growing approach. Since the curve-skeleton contains only curve-segments, flat object parts cannot be reconstructed from it. Cylindrical shapes, (i.e., shapes that can be accurately represented by generalized cylinders), represent a special class of objects that can be accurately reconstructed from the curve-skeleton alone. General shapes, however, can only be approximated by a generalized cylinder reconstruction. Clearly, a denser curve-skeleton will generate a better reconstruction [43].

Reconstruction using the ball-growing approach [43] needs distance field information in order to determine the radius of the ball that will be grown from each curve-skeleton point. In this respect, the distance field-based methods have an advantage over the other methods because this information is already available.

6.4 Thinness (1D)

Thinning algorithms can either directly produce a curve-skeleton (by using curve-thinning templates) or further thin a surface skeleton to a 1D representation. Parallel thinning algorithms, which remove all simple points at once, may not be able to achieve 1D skeletons due to homotopy constraints. An illustrative 2D example is the case of a rectangle whose width is an even number of voxels: In the last step of the thinning process, the middle section of the skeleton will be a curve of width 2. Although all points of this curve are simple points, removing them would completely remove the middle section. At this stage, no other simple points can be removed and the skeleton is not 1D. Directional thinning methods do not have this disadvantage: one row of voxels in the middle section will be removed when processing the up-down direction (for example) and the second row will be preserved in subsequent steps.

Distance field methods and Voronoi-based geometric methods do not produce a 1D skeleton directly. Both require significant postprocessing to reduce the candidate voxels (distance field) or the medial surface (Voronoi-based methods) to a curve-skeleton.

The Reeb graph-based geometric methods and the mesh decomposition-based method of [59] directly produce 1D straightline skeleton segments by connecting level-set or component junction centroids, although the representation is no longer voxel-based.

Thinness is an implicit property of the general-field methods that use force-following or active contours to generate 1D skeleton branches.

6.5 Centeredness

Thinning and general field methods do not guarantee centeredness. In the case of directional thinning, this would depend on the order in which the different directions are applied. In the case of general field methods, since they are taking into account a larger surface area than the two closest points, centeredness is usually compromised.

Methods using a distance field can better achieve centeredness because of the centeredness information included in the distance field. However, once clustering and spanning trees are used, centeredness may be lost (see, for example, [129]). Distance field methods using a distance to source (for example, geodesic field propagation) do not generate centered curve-skeletons. The problem is described in detail in [96] and the solution provided there combines the distance-to-source field with a global distance-to-boundary field to achieve centeredness.

Geometric methods directly computing contact points (points where the maximal inscribed spheres touch the surface) [68] can also achieve centeredness since these points can be incorporated more easily into the pruning steps. Voronoi-based methods are dependent on the sampling density of the object's surface: A denser sampling produces a more centered curve-skeleton [4], but the running time increases. Centeredness of level-set [127] and mesh decomposition-based [59] curve-skeletons is poor because centroids are directly connected with straightline segments, regardless of the configuration of the object between these points. Centeredness problems arise especially in regions where the topology of the object changes

between successive level sets. It is also influenced by the resolution (the distance between two successive level sets).

Resolution affects any centeredness measurement in the discrete domain. Using the same example of a shape whose width is an even number of voxels, if this shape is reduced to a 1D skeleton, at the grid's resolution, the curve-skeleton must be one voxel closer to one of the sides than to the other.

6.6 Reliability

Reliability is an application specific property important for virtual navigation (each boundary point must be visible from at least one curve-skeleton location). For this reason, this property is only guaranteed by those algorithms developed specifically for that purpose (see, for example, [52], [61]).

Regardless of the algorithm used to compute the curve-skeleton, reliability can be easily checked using the visibility test for each boundary location on the original object. If necessary, more branches can be added to the curve-skeleton in a postprocessing step.

6.7 Junction Detection and Component-Wise Differentiation

The ability to distinguish the different components of the curve-skeleton depends on the ability to detect the junction points, i.e., the points where two or more curves meet. From this decomposition, one can infer the corresponding part structure of the original object.

Some thinning algorithms directly classify the skeleton points as junctions, either during thinning [120] or as a postprocessing step [28]. Distance field methods must test for joints after significant pruning and clustering [129]. However, junction placement for these classes of methods is sensitive to noise.

From the geometric algorithm class, level-set methods directly identify the joints as the centroids of level-sets. Joint locations depend on the function used to define the level sets. Similarly, the joints of the mesh decomposition-based curve-skeleton, identified as the mesh component junctions, depend on the coarseness of the decomposition. The edge-contraction method can easily identify junctions by inspecting the vertices of the skeleton as a postprocessing step.

General-field methods can identify joints directly before extracting the curve-skeleton by locating the critical points of the underlying vector field [35], during extraction as the local extrema where the force-following algorithm stops [32], or the points where a previously visited location is encountered. Joint placement depends on the function used to define the underlying field.

6.8 Connectivity

Connectivity is usually checked by all the algorithms. Some algorithms (e.g., thinning, level-set based geometric methods) explicitly maintain connectivity during computation, while other methods check and enforce connectivity in a postprocessing step.

6.9 Robustness

Thinning, distance field, and Voronoi-based geometric methods are sensitive to noise, generating unnecessary

branches in the skeleton as a result. Several methods have been proposed to filter the resulting skeletons [7], [36].

Level-set-based geometric methods are affected by noise in different ways. While the location of the level-set centroids should not be affected by noise because of the averaging effect involved in computing the centroid, noise can adversely affect the number of contours in a level set, thus generating undesirable branches in the resulting skeleton.

General field approaches are less susceptible to noise because of the large amount of averaging included in the underlying computation. These methods are more sensitive to resolution because thin regions in the objects can cause numerical instabilities in the computations.

Many of the algorithms described in the literature are usually illustrated with only a few examples and are not tested on a large database of general 3D objects (for example, the Princeton Shape Benchmark [113], a database of 1,814 3D models). Thus, it is unclear how robust and general these algorithms are with respect to the choice of their parameters.

6.10 Smoothness

Due to their discrete nature, thinning algorithms do not produce smooth curve-skeletons. Boundary irregularities propagate all the way to the curve-skeleton during the thinning process.

Distance field methods have the same disadvantage because there is no averaging involved in the computation of the distance field values. However, the subsequent pruning and reconnection steps can incorporate some smoothing constraints. Voronoi-based geometric methods behave in a similar way.

Level-set geometric methods introduce smoothing in the computation of level-set centroids, but topology changes in the evolution of level sets are not handled in a way that preserves smoothness. The same is true for methods based on mesh decomposition.

In the case of general field methods, extensive averaging is employed during the computation of the vector field and the effect is improved smoothness of the extracted curve-skeleton.

Although some algorithms may produce smoother curve-skeletons than others, smoothing can be performed in a postprocessing step, regardless of the extraction algorithm used to compute the initial curve-skeleton.

6.11 Hierarchy

Hierarchy (the ability to create a family of curve-skeletons of increased complexity) is not achievable using thinning algorithms because, when processing a voxel, there are only two choices: keep it or remove it. A curve-skeleton is obtained only after the last iteration of the algorithm.

Distance field methods can produce a hierarchy of curve-skeletons by varying the number of candidate voxels selected in the pruning step. In order to obtain a strict hierarchy (i.e., the curve-skeleton at one level is included in the next level curve-skeleton), the reconnection step must take into account the previous level curve-skeleton. Similarly to distance field, Voronoi-based geometric methods can produce hierarchic curve-skeletons by pruning the

surface skeleton at different thresholds and reconnecting the candidate voxels into a hierarchy of curve-skeletons. In the case of mesh decomposition-based methods, if the decomposition process is a hierarchical one [59], the produced curve-skeletons will also form a hierarchy. Reeb graph-based geometric methods are not hierarchic.

General field methods produce hierarchic curve-skeletons by varying the number of seed points used to construct individual curve-skeleton segments. Additional segments added to a curve-skeleton do not affect the existing segments, creating a strict hierarchy [35].

6.12 Efficiency

Thinning is practically a linear process in the number of object voxels. Most of the voxels of the input object are removed when they are first processed (they are simple points). The nonsimple points are processed again at every subsequent thinning step until they are finally removed or the algorithm terminates. An exact complexity analysis of such algorithms is difficult since they are data dependent.

The Euclidean distance field of a 3D object can be computed in linear time using the algorithm of Saito and Toriwaki [106]. The subsequent steps of filtering and reconnecting the curve-skeleton may, however, have a higher complexity (but, they usually operate on a greatly reduced set of voxels).

Computation of the Voronoi diagram of a set of n points in 3D is $O(n^2)$ in the worst case, although, in practice, it is almost linear [4]. As in the distance field case, additional processing is necessary to get a curve-skeleton, but the number of input elements is reduced. The computational complexity of the level-set-based geometric methods depends on the kind of function computed over the object. The height function can be computed in linear time, while the exact integral of the geodesic function is $O(n^2)$ [53] (n is the number of mesh vertices). The following steps, detecting connected components in each level set and computing their centroids, are linear in n .

The complexity of potential field computation is $O(n^2)$ [35], where n is the number of object voxels. These methods are computationally more intensive than distance field methods because they take into consideration larger boundary areas, not just the closest point.

6.13 Handle Point Sets

Objects represented by a set of points on the boundary can be converted to a voxelized point representation by mapping each sample point to the closest voxel. Note that this transformation is different from voxelizing a polygon mesh. In the latter case, the interior of the mesh is completely filled with object voxels, while, in the former case, the interior and the full boundary of the object are still unknown. Thinning algorithms cannot operate on such a representation since they need to know the complete interior and boundary of the object.

Distance fields can be computed using the known boundary points as sources, but the field will extend outside the object since the distinction inside/outside is not known. The resulting curve-skeleton will also have branches outside, as well as inside the object, and it may be difficult to distinguish between them.

TABLE 1
Summary of Properties Achievable
by the Various Algorithm Classes

	Thinning	Distance Field	Geometric	General Field
Homotopic	Y		Y	N
Transf. Invariance		Y		Y
Reconstruction	N		N	N
Thin				Y
Centered				
Reliable				
Junction Detection			Y	Y
Connected	Y			
Robust	N	N	N	Y
Smooth				Y
Hierarchic	N			Y
Efficiency	Y	Y	Y	N
Handle Point Sets	N		Y	Y

Voronoi-based geometric methods directly work with objects represented by a set of samples on the boundary [4]. Level-set-based geometric methods can also handle such representations as demonstrated in [127]. Mesh decomposition-based geometric methods, however, cannot handle this case.

Point samples on the object's boundary can be used as sources for the general field methods. As for the distance field, the field will also extend outside the object. However, the curve-skeleton segments outside the objects can be easily identified and removed if using a force-following algorithm since they will be touching the bounding box of the volume (see [35]).

6.14 Discussion Summary

In Table 1 we list, for each algorithm class, the properties it can achieve, as discussed in this section. A "Y" signifies that the algorithm class guarantees that particular property. An "N" is used if the algorithm class cannot achieve a given property. Finally, an empty space is used if a property can be achieved by some, but not all, of the algorithms in a class or if a property is difficult, but not impossible, to achieve using this class of algorithms (as per the detailed discussion above).

7 IMPLEMENTATION

In Section 5, four classes for the curve-skeletonization algorithms were described. The classes are divided into a "core" part and a "postcore" step, which is necessary to prune, cluster, connect, or smooth the curve-skeleton. In this section, we describe the results of comparing one algorithm from each class on several test objects, including one "real" object (a colon data set) and one object with noise (the chess piece). Because many of the algorithms described in the literature are difficult to implement (typically, not all of the implementation details are given, e.g., specific thresholds, epsilon values, and cluster parameters), we have only implemented the "core" part of the algorithms (i.e., the first step(s)).

For the distance field and the Voronoi diagram-based geometric methods, a curve-skeleton is more difficult to obtain directly. The first step of these methods generates a structure closer to a medial surface, so additional processing is required to obtain a 1D curve-skeleton. To illustrate, we

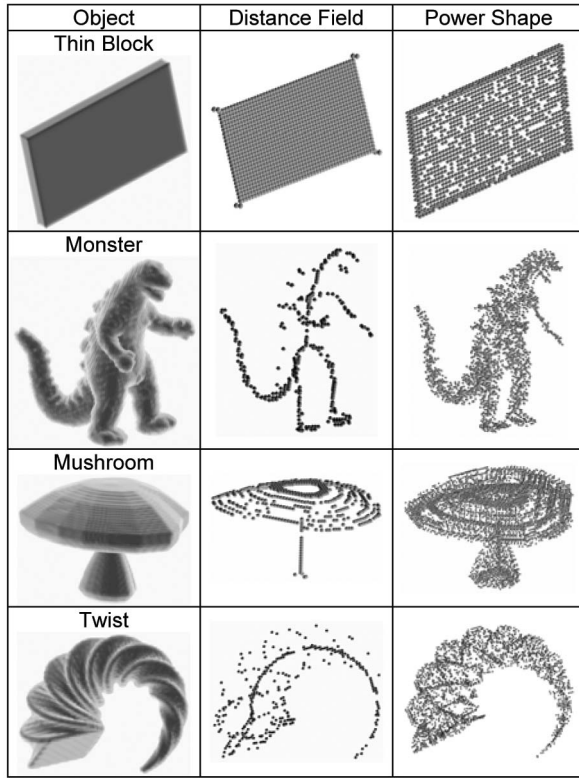


Fig. 8. The candidate voxels selected from a distance field using [43] and the Voronoi diagram-based power shape [4] give a sense of how much additional postprocessing is required to get a 1D curve-skeleton.

used the parameter controlled filtering of the distance function described by Gagvani and Silver in [43] for distance field-based methods and Amenta et al.'s implementation of the *power shape* (Voronoi diagram-based approximation of the skeleton) [4]. Note that, for the *power shape* algorithm, only the surface voxels were given as input to the program and the results shown in the figure are the inside poles determined by the algorithm [4]. A comparison of the resulting structures for several objects is shown in Fig. 8. The purpose of this comparison is to give the reader a sense of how much additional processing would be required in order to extract a curve-skeleton from these structures. Note that the additional processing has to rely on some other information not given by the underlying method used in the first step. For example, pruning the medial surface patch of the box cannot take advantage of the distance field values since they are the same for all voxels on it.

In our previous work [34], we compared the different algorithms using the implementations described above for the distance field and the geometric classes. However, we felt that the comparison was not fair because these implementations did not extract a 1D curve-skeleton directly, while the thinning and potential field methods did. In order to make the comparison fairer, in this paper we attempt to modify the distance field and geometric implementations to extract a 1D curve-skeleton.

In the distance field case, we extended the implementation with two additional steps performed by most algorithms in this class (see Section 5.2): pruning the candidate voxels and connecting them. Specifically, we used clustering to reduce the number of candidate voxels and a

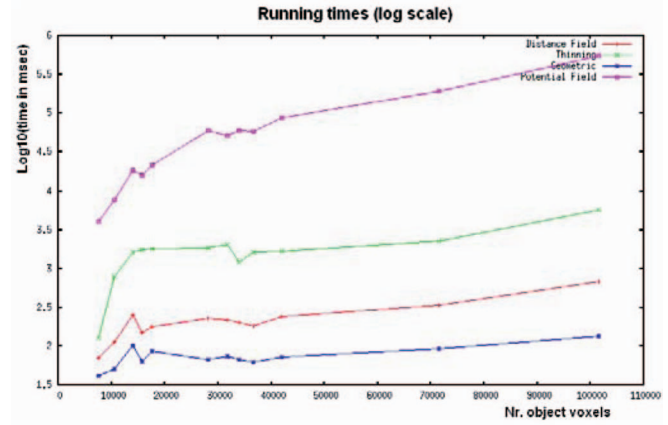


Fig. 9. Running time (log scale) versus number of object voxels.

minimum spanning tree algorithm to connect the centroids of the clusters.

To represent the geometric methods, we changed our implementation to a Reeb graph-based one, which does produce a curve-skeleton in the first step of processing. We have chosen the height function because of its straightforward implementation. We detect connected components in each level-set of the height function, select the centroid of each connected component as its representative, and add it to the curve-skeleton. When the topology of the level sets changes, we connect the centroids of neighboring components in two adjacent level sets with a straight line.

From the thinning class, we implemented the 12-subiteration curve thinning algorithm described by Palágyi and Kuba in [95].

We implemented the potential field method described in [35] to illustrate the general field methods. Since this is a hierarchical method, which produces curve-skeletons of various complexities, for this comparison, we only used the first level of the hierarchy (level 0), i.e., the curve-skeleton generated in the first step by connecting the critical points of the vector field. Both the thinning method and the potential field method directly produce a 1D curve-skeleton.

The results of our implementations on a set of test objects are shown in Fig. 10. Note that, for these results, no attempt was made to tweak the parameters of the programs in order to obtain a better curve-skeleton for one object or another. The distance-field based curve-skeleton was obtained using a thinness threshold of 0.6 for all objects (see [43]). This threshold was selected for one object (the horse) to remove most of the extra branches, but still maintain most of the characteristics of the horse shape. Then, this value was used for all the other objects. For the potential field implementation, we used a field strength parameter value of 6 since, in our experience, this value produced good curve-skeletons, and no additional seed points besides the critical points were used. For the Reeb graph-based geometric implementation, the height function sampled the objects in the “up” direction (increasing Y values) at every discrete value of Y. There were no parameters for the thinning implementation.

Most of the test objects, except the colon and the knight chess piece, were voxelized from models downloaded from the Princeton Shape Benchmark Database [113]. The grid resolution was chosen such that its largest dimension is

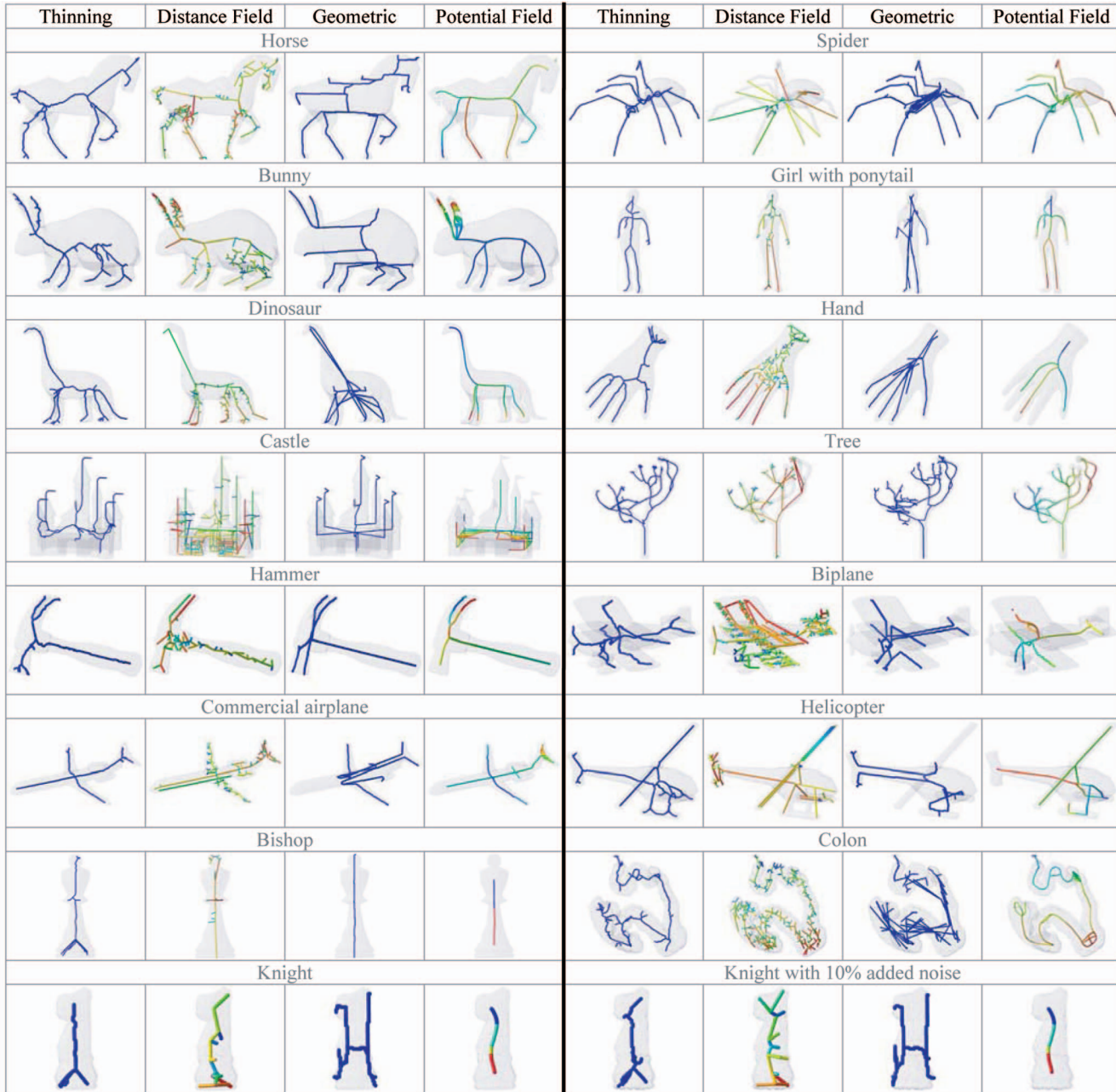


Fig. 10. Skeletons of various objects obtained using different curve-skeletonization algorithms. The comparisons are on the first steps of all of the algorithms. Note: Our implementations are not fully featured and the resulting curve-skeletons may be improved. See text for details about each implementation. All of the objects and algorithm implementations are also available for download at [139].

300 voxels and the aspect ratio of the object is maintained. The colon data set has a resolution of $204 \times 132 \times 260$ and the knight chess piece is $40 \times 39 \times 87$. The source code for our implementations and all models used in the experimental section can be downloaded from [139].

From these results, it is clear that the potential field method yields the cleanest and smoothest curve-skeleton at the initial stage. This is due to the “global averaging” effect of the potential field. For objects with thin, flat regions, like the ears of the bunny, the algorithm identifies too many critical points, creating a large number of curve-skeleton branches in that area, very similar to a medial surface. This is because the algorithm is resolution dependent, a disadvantage identified in [35]. Also note that we are using

only the first level (level 0) of the curve-skeleton hierarchy presented in [35]. While this simple curve-skeleton is good enough for some objects, for others, certain parts are not represented in the curve-skeleton at all: The tail of the dinosaur and two of the hand fingers are missing, for example. By using different values for the field strength parameter (fixed at 6 for the purpose of this comparison for all objects) or with the use of additional seed points (detected automatically), the resulting curve-skeleton can be improved.

The thinning algorithm performs well in all cases, but the resulting curve-skeletons are not smooth since all irregularities of the surface propagate to the curve-skeleton through the erosion process. This also makes it more sensitive to

noise. Furthermore, this noise propagation problem affects the location of the joints, which can be identified in a postprocessing step because the resulting curve-skeletons are one voxel thick in all directions. As there are no parameters to set for this algorithm, one can only improve the curve-skeleton in a postprocessing step, using smoothing and possibly branch pruning to remove extraneous branches.

The distance field-based algorithm produces curve-skeletons with a large number of extra branches. They are generated by the relatively large number of candidate voxels selected using the thinness heuristic. The centroids of the clusters are connected using straight lines, so the centeredness property is not always satisfied. In fact, some curve-skeleton segments can go outside the object entirely as for the spider and the dinosaur. Clearly, the resulting curve-skeleton can be improved by using more sophisticated algorithms for selecting candidate voxels or for connecting them, but they come at a cost: more effort from the implementer and possibly longer running times.

The Reeb graph-based geometric algorithm produces strange results for some objects because the objects were not always oriented in the natural “up-down” direction as they are presented in the figure. For example, for the dinosaur and commercial airplane shapes, the slicing direction corresponds to the front-to-back direction in the figure. This aspect can be improved upon by using different slicing directions for each object. A better alternative is to use an orientation independent function to define the level sets, such as the geodesic distance. Since the centroids of adjacent connected components are connected using straight lines, the centeredness property is compromised, especially when the topology of the level sets changes. As in the distance field case, some segments are outside the object, as we can see for the dinosaur or the commercial airplane objects. This problem is difficult to fix regardless of the function used to define the level sets.

Although the implementations directly produce a curve-skeleton for each class of methods, the comparison is still not completely fair. We cannot claim that each of our implementations is fully representative for an entire class of algorithms as many improvements to the resulting curve-skeletons can surely be made. However, we believe that general characteristics of each class of methods transpire from these results, as discussed in this section.

In terms of running time, the potential field method is the slowest. In Fig. 9, we show a comparison of the running times (in milliseconds, on a logarithmic scale) recorded for each method as a function of the total number of object voxels (note that no optimizations were done for the implementations).

As mentioned in the beginning of the paper, the various algorithms presented here operate either on discrete voxelized data sets or on continuous polygonal representations of 3D object. Since conversion between these representations can be done using well-known algorithms (voxelization or surface extraction), the availability of one representation or the other should not be a deciding factor in choosing the curve-skeletonization algorithm.

8 SUMMARY

A plethora of different curve-skeletonization algorithms have been described in the literature. In this paper, we have given an overview of the visualization applications that use curve-skeletons and have distilled a list of curve-skeleton properties necessary for these applications. We have then classified the algorithms for computing the curve-skeleton based on their implementations and have discussed to what extent each methodology achieves the different curve-skeleton properties. To illustrate this, we implemented a “core” curve-skeletonization algorithm from each class and compared the results on the same set of test objects.

ACKNOWLEDGMENTS

This work is supported in part by US National Science Foundation grants NSF 0118760 and NSF EIA-0205178. The authors would like to thank Dr. Raman Balasubramanian and Xiaosong Yuan for their initial help. They would also like to thank the reviewers for their useful comments and suggestions. The source code and all models used in the experimental section can be downloaded from [139].

REFERENCES

- [1] 3D Studio Max, <http://www.autodesk.com/3dsmax>, 2006.
- [2] G.H. Abdel-Hamid and Y.-H. Yang, “Multiresolution Skeletonization: An Electrostatic Field-Based Approach,” *Proc. IEEE Int’l Conf. Image Processing*, pp. 949-953, 1994.
- [3] N. Ahuja and J. Chuang, “Shape Representation Using a Generalized Potential Field Model,” *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 19, no. 2, pp. 169-176, Feb. 1997.
- [4] N. Amenta, S. Choi, and R. Kolluri, “The Power Crust,” *Proc. Sixth ACM Symp. Solid Modeling*, pp. 249-260, 2001.
- [5] N. Amenta, S. Choi, and R. Kolluri, “The Power Crust, Unions of Balls, and the Medial Axis Transform,” *Computational Geometry: Theory and Applications*, vol. 19, nos. 2-3, pp. 127-153, 2001.
- [6] D. Attali, J.-D. Boissonnat, and H. Edelsbrunner, “Stability and Computation of the Medial Axis—A State-of-the-Art Report,” *Math. Foundations of Scientific Visualization, Computer Graphics, and Massive Data Exploration*, Springer-Verlag, 2004.
- [7] D. Attali and J.-O. Lachaud, “Delaunay Conforming Iso-Surface, Skeleton Extraction and Noise Removal,” *Computational Geometry*, vol. 19, nos. 2-3, pp. 175-189, 2001.
- [8] M. Attene, S. Biasotti, and M. Spagnuolo, “Shape Understanding by Contour-Driven Retiling,” *The Visual Computer*, vol. 19, nos. 2-3, p. 127, 2003.
- [9] S.R. Aylward and E. Bullitt, “Initialization, Noise, Singularities, and Scale in Height Ridge Traversal for Tubular Object Centerline Extraction,” *IEEE Trans. Medical Imaging*, vol. 21, no. 2, 2002.
- [10] S.R. Aylward, J. Jomier, S. Weeks, and E. Bullitt, “Registration and Analysis of Vascular Images,” *Int’l J. Computer Vision*, vol. 55, nos. 2-3, 2003.
- [11] D.C. Banks and B.A. Singer, “Vortex Tubes in Turbulent Flows: Identification, Representation, Reconstruction,” *Proc. IEEE Visualization Conf.*, 1994.
- [12] D. Bartz, W. Straßer, M. Skalej, and D. Welte, “Interactive Exploration of Extra- and Intracranial Blood Vessels,” *Proc. IEEE Visualization Conf.*, 1999.
- [13] G. Bertrand and Z. Aktouf, “A Three-Dimensional Thinning Algorithm Using Subfields,” *Vision Geometry III*, vol. 2356, pp. 113-124, SPIE, 1994.
- [14] S. Biasotti, B. Falcidieno, and M. Spagnuolo, “Extended Reeb Graphs for Surface Understanding and Description,” *Proc. Int’l Conf. Discrete Geometry for Computer Imagery*, p. 185, 2000.
- [15] S. Biasotti, S. Marini, M. Mortara, and G. Patané, “An Overview on Properties and Efficacy of Topological Skeletons in Shape Modeling,” *Proc. Int’l Conf. Shape Modeling and Applications*, 2003.

- [16] S. Biasotti, S. Marini, M. Mortara, G. Patanè, M. Spagnuolo, and B. Falcidieno, "3D Shape Matching through Topological Structures," *Lecture Notes in Computer Science*, vol. 2886, p. 23, 2003.
- [17] I. Bitter, A.E. Kaufman, and M. Sato, "Penalized-Distance Volumetric Skeleton Algorithm," *IEEE Trans. Visualization and Computer Graphics*, vol. 7, no. 3, July-Sept. 2001.
- [18] R.L. Blanding, G.M. Turkiyyah, D.W. Storti, and M.A. Ganter, "Skeleton-Based Three-Dimensional Geometric Morphing," *Computational Geometry*, vol. 15, pp. 129-148, 2000.
- [19] J. Bloomenthal, "Medial Based Vertex Deformation," *Proc. SIGGRAPH/Eurographics Symp. Computer Animation*, pp. 147-151, 2002.
- [20] J. Bloomenthal and C. Lim, "Skeletal Methods of Shape Manipulation," *Proc. Int'l Shape Modeling Conf.*, 1999.
- [21] H. Blum, "A Transformation for Extraction New Descriptors of Shape," *Models for the Perception of Speech and Visual Form*, MIT Press, 1967.
- [22] H. Blum and R.N. Nagel, "Shape Description Using Weighted Symmetric Axis Features," *Pattern Recognition*, vol. 10, no. 3, p. 167, 1978.
- [23] G. Borgefors, "On Digital Distance Transforms in Three Dimensions," *Computer Vision and Image Understanding*, vol. 64, no. 3, pp. 368-376, 1996.
- [24] G. Borgefors, I. Nyström, and G. Sanniti di Baja, "Computing Skeletons in Three Dimensions," *Pattern Recognition*, vol. 32, no. 7, 1999.
- [25] S. Bouix and K. Siddiqi, "Divergence-Based Medial Surfaces," *Proc. European Conf. Computer Vision (ECCV)*, pp. 603-618, 2000.
- [26] J.W. Brandt and V.R. Alazi, "Continuous Skeleton Computation by Voronoi Diagram," *CVGIP: Image Understanding*, vol. 55, pp. 329-338, 1992.
- [27] A. Brennecke and T. Isenberg, "3D Shape Matching Using Skeleton Graphs," *Simulation and Visualization*, pp. 299-310, 2004.
- [28] G.J. Brostow, I. Essa, D. Steedly, and V. Kwatra, "Novel Skeletal Representation for Articulated Creatures," *Proc. European Conf. Computer Vision*, 2004.
- [29] D. Brunner and G. Brunnett, "Mesh Segmentation Using the Object Skeleton Graph," *Proc. Int'l Conf. Computer Graphics and Imaging*, pp. 48-55, 2004.
- [30] C.A. Burbeck and S.M. Pizer, "Object Representation by Cores: Identifying and Representing Primitive Spatial Regions," *Vision Research*, vol. 35, no. 13, p. 1917, 1995.
- [31] S.W. Choi and H.P. Seidel, "Linear One-Sided Stability of MAT for Weakly Injective 3D Domain," *Proc. ACM Symp. Solid Modeling and Applications*, 2002.
- [32] J. Chuang, C. Tsai, and M.-C. Ko, "Skeletonization of Three-Dimensional Object Using Generalized Potential Field," *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 22, no. 11, p. 1241, Nov. 2000.
- [33] N.D. Cornea, M.F. Demirci, D. Silver, A. Shokoufandeh, S.J. Dickinson, and P.B. Kantor, "3D Object Retrieval Using Many-to-Many Matching of Curve-Skeletons," *Proc. Int'l Shape Modeling Conf.*, 2005.
- [34] N.D. Cornea, D. Silver, and P. Min, "Curve-Skeleton Applications," *Proc. IEEE Visualization Conf.*, p. 95, 2005.
- [35] N. Cornea, D. Silver, X. Yuan, and R. Balasubramanian, "Computing Hierarchical Curve-Skeletons of 3D Objects," *The Visual Computer*, vol. 21, no. 11, pp. 945-955, 2005.
- [36] M. Couprie and R. Zrou, "Discrete Bisector Function and Euclidean Skeleton," *Lecture Notes in Computer Science*, vol. 3429, 2005.
- [37] T. Culver, J. Keyser, and D. Manocha, "Exact Computation of the Medial Axis of a Polyhedron," *Computer Aided Geometric Design*, vol. 21, no. 1, pp. 65-98, 2004.
- [38] T.K. Dey and J. Sun, "Defining and Computing Curve-Skeletons with Medial Geodesic Function," *Proc. Eurographics Symp. Geometry Proc.*, 2006.
- [39] U. Eckhardt and G. Maderlechner, "Invariant Thinning," *Pattern Recognition and Artificial Intelligence*, no. 7, pp. 1115-1144, 1993.
- [40] A.F. Frangi, W.J. Niessen, R.M. Hoogeveen, T. van Walsum, and M.A. Viergever, "Model-Based Quantitation of 3D Magnetic Resonance Angiographic Images," *IEEE Trans. Medical Imaging*, vol. 18, no. 10, p. 946, 1999.
- [41] Y. Fridman, S.M. Pizer, S. Aylward, and E. Bullitt, "Extracting Branching Tubular Object Geometry via Cores," *Medical Image Analysis*, vol. 8, no. 3, pp. 169-176, 2004.
- [42] D.S. Fritsch, S.M. Pizer, B.S. Morse, D.H. Eberly, and A. Liu, "The Multiscale Medial Axis and Its Applications in Image Registration," *Pattern Recognition Letters*, vol. 15, pp. 445-452, 1994.
- [43] N. Gagvani and D. Silver, "Parameter Controlled Volume Thinning," *Graphical Models and Image Processing*, vol. 61, no. 3, pp. 149-164, 1999.
- [44] N. Gagvani and D. Silver, "Animating Volumetric Models," *Academic Press Professional*, vol. 63, no. 6, pp. 443-458, 2001.
- [45] N. Gagvani and D. Silver, "Shape-Based Volumetric Collision Detection," *Proc. IEEE Symp. Volume Visualization*, 2000.
- [46] P. Giblin and B.B. Kimia, "A Formal Classification of 3D Medial Axis Points and Their Local Geometry," *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, pp. 566-573, 2000.
- [47] P. Golland and W.E.L. Grimson, "Fixed Topology Skeletons," *Computer Vision and Pattern Recognition*, 2000.
- [48] W. Gong and G. Bertrand, "A Simple Parallel 3D Thinning Algorithm," *Proc. IEEE Conf. Pattern Recognition*, pp. 188-190, 1990.
- [49] B. Gorte and N. Pfeifer, "Structuring Laser-Scanned Trees Using 3D Mathematical Morphology," *Proc. XXth Congress of Int'l Archives of Photogrammetry*, p. 929, 2004.
- [50] J. Greer, "Computer Skeletonization and Automatic Electron Density Map Analysis," *Methods in Enzymology*, vol. 115, pp. 206-224, 1985.
- [51] T. Grigorishin and Y.H. Yang, "Skeletonization: An Electrostatic Field-Based Approach," *Pattern Analysis and Applications*, vol. 1, pp. 163-177, 1998.
- [52] T. He, L. Hong, D. Chen, and Z. Liang, "Reliable Path for Virtual Endoscopy: Ensuring Complete Examination of Human Organs," *IEEE Trans. Visualization and Computer Graphics*, vol. 7, no. 4, pp. 333-342, Oct.-Dec. 2001.
- [53] M. Hilaga, Y. Shinagawa, T. Kohmura, and T. Kunii, "Topology Matching for Fully Automatic Similarity Estimation of 3D Shapes," *Proc. SIGGRAPH*, 2001.
- [54] L. Hong, S. Muraki, A. Kaufman, D. Bartz, and T. He, "Virtual Voyage: Interactive Navigation in the Human Colon," *Proc. SIGGRAPH*, 1997.
- [55] S. Hornus, A. Angelidis, and M.-P. Cani, "Implicit Modeling Using Subdivision Curves," *The Visual Computer*, vol. 19, nos. 2-3, p. 94, 2003.
- [56] P.P. Jonker, "Skeletons in N Dimensions Using Shape Primitives," *Pattern Recognition Letters*, vol. 23, no. 6, p. 677, 2002.
- [57] A. Kanitsar, D. Fleischmann, R. Wegenkittl, P. Felkel, and M.E. Gröller, "CPR: Curved Planar Reformation," *Proc. IEEE Visualization Conf.*, 2002.
- [58] A. Kanitsar, R. Wegenkittl, D. Fleischmann, and M.E. Gröller, "Advanced Curved Planar Reformation: Flattening of Vascular Structures," *Proc. IEEE Visualization Conf.*, 2003.
- [59] S. Katz and A. Tal, "Hierarchical Mesh Decomposition Using Fuzzy Clustering and Cuts," *Proc. ACM SIGGRAPH*, 2003.
- [60] R.A. Katz and S.M. Pizer, "Untangling the Blum Medial Axis Transform," *Int'l J. Computer Vision*, vol. 55, nos. 2-3, pp. 139-153, 2003.
- [61] D.-G. Kang and J.B. Ra, "A New Path Planning Algorithm for Maximizing Visibility in Computed Tomography Colonography," *IEEE Trans. Medical Imaging*, vol. 24, no. 8, p. 957, 2005.
- [62] T.Y. Kong and A. Rosenfeld, "Digital Topology: Introduction and Survey," *Computer Vision, Graphics, and Image Processing*, vol. 48, no. 3, pp. 357-393, 1989.
- [63] T.Y. Kong, A.W. Roscoe, and A. Rosenfeld, "Concepts of Digital Topology," *Topology and Its Applications*, vol. 46, no. 3, pp. 219-262, Elsevier, 1992.
- [64] F. Lazarus and A. Verroust, "Metamorphosis of Cylinder-Like Objects," *J. Visualization and Computer Animation*, vol. 8, no. 3, pp. 131-146, 1998.
- [65] F. Lazarus and A. Verroust, "Level Set Diagrams of Polyhedral Objects," *Proc. ACM Solid Modeling*, 1999.
- [66] T. Lee and R.L. Kashyap, "Building Skeleton Models via 3D Medial Surface/Axis Thinning Algorithms," *CVGIP: Graphical Models and Image Processing*, vol. 56, no. 6, pp. 462-478, Nov. 1994.
- [67] F.F. Leymarie and B.B. Kimia, "Computation of the Shock Scaffold for Unorganized Point Clouds in 3D," *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, 2003.
- [68] F.F. Leymarie, "3D Shape Representation via Shock Flows," PhD thesis, Brown Univ., May 2003.
- [69] H. Li, "An Integrated Approach to Protein Backbone Modeling," MS thesis, Queen's Univ., Kingston, Ontario, Canada, Oct. 2002.

- [70] X. Li, T.W. Woon, and Z. Huang, "Decomposing Polygon Meshes for Interactive Applications," *Proc. ACM Symp. Interactive 3D Graphics*, pp. 35-42, 2001.
- [71] J.-H. Lien and N.M. Amato, "Simultaneous Shape Decomposition and Skeletonization," Technical Report TR05-015, Parasol Lab, Dept. of Computer Science, Texas A&M Univ., Dec. 2005.
- [72] A. Lieutier, "Any Open Bounded Subset of \mathbb{R}^n Has the Same Homotopy Type than Its Medial Axis," *Proc. ACM Shape Modeling Int'l Conf.*, 2003.
- [73] P. Liu, F. Wu, W. Ma, R. Liang, and M. Ouhyoung, "Automatic Animation Skeleton Construction Using Repulsive Force Field," *Proc. 11th Pacific Conf. Computer Graphics and Applications*, 2003.
- [74] C. Lohou and G. Bertrand, "A 3D 12-Subiteration Thinning Algorithm Based on P-Simple Points," *Discrete Applied Math.*, vol. 139, pp. 171-195, 2004.
- [75] W.E. Lorensen and H.E. Cline, "Marching Cubes: A High Resolution 3D Surface Construction Algorithm," *Proc. ACM SIGGRAPH*, vol. 21, no. 4, pp. 163-169, 1987.
- [76] C.M. Ma and M. Sonka, "A Fully Parallel 3D Thinning Algorithm and Its Applications," *Computer Vision and Image Understanding*, vol. 64, no. 3, pp. 420-433, 1996.
- [77] C.-M. Ma and S.-Y. Wan, "A Medial-Surface Oriented 3D Two-Subfield Thinning Algorithm," *Pattern Recognition Letters*, vol. 22, p. 1439, 2001.
- [78] C.-M. Ma, S.-Y. Wan, and H.-K. Chang, "Extracting Medial Curves on 3D Images," *Pattern Recognition Letters*, vol. 23, p. 895, 2002.
- [79] C.-M. Ma, S.-Y. Wan, and J.-D. Lee, "Three-Dimensional Topology Preserving Reduction on the 4-Subfields," *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 24, no. 12, p. 1594, Dec. 2002.
- [80] W. Ma, F. Wu, and M. Ouhyoung, "Skeleton Extraction of 3D Objects with Radial Basis Functions," *Proc. IEEE Int'l Conf. Shape Modeling and Applications*, 2003.
- [81] G. Malandain and S. Fernandez-Vidal, *Euclidean Skeletons, Image and Vision Computing*, vol. 16, pp. 317-327, 1998.
- [82] A. Manzanera, T. Bernard, F. Preteux, and B. Longuet, "A Unified Mathematical Framework for a Compact and Fully Parallel n-D Skeletonization Procedure," *Proc. Vision Geometry VIII*, pp. 57-68, 1999.
- [83] Maya, Autodesk, <http://www.autodesk.com/maya>, 2006.
- [84] D.G. Morgenthaler, "Three-Dimensional Simple Points: Serial Erosion, Parallel Thinning and Skeletonization," TR-1005, Computer Vision Lab., Univ. of Maryland, 1981.
- [85] B.S. Morse, S.M. Pizer, D.T. Puff, and C. Gu, "Zoom-Invariant Vision of Figural Shape: Effects on Cores of Image Disturbances," *Computer Vision and Image Understanding*, vol. 69, no. 1, p. 72, 1998.
- [86] M. Mortara and G. Patané, "Shape-Covering for Skeleton Extraction," *Int'l J. Shape Modeling*, vol. 8, no. 2, p. 139, 2002.
- [87] M. Näf, O. Kubler, R. Kikinis, M.E. Shenton, and G. Szekely, "Characterization and Recognition of 3D Organ Shape in Medical Image Analysis Using Skeletonization," *Proc. IEEE Workshop Math. Methods in Biomedical Image Analysis*, pp. 139-150, 1996.
- [88] C.W. Niblack, P.B. Gibbons, and D.W. Capson, "Generating Skeletons and Centerlines from the Distance Transform," *Graphical Models and Image Processing*, vol. 54, no. 5, pp. 420-437, 1992.
- [89] I. Nyström, G. Sanniti di Baja, and S. Svensson, "Representing Volumetric Vascular Structures Using Curve Skeletons," *Proc. Int'l Conf. Image Analysis and Processing*, p. 495, 2001.
- [90] I. Nyström and Ö. Smedby, "Skeletonization of Volumetric Vascular Images—Distance Information Utilized for Visualization," *J. Combinatorial Optimization*, vol. 5, pp. 27-41, 2001.
- [91] R. Ogniewicz, "A Multiscale MAT from Voronoi Diagrams: The Skeleton-Space and Its Application to Shape Description and Decomposition," *Aspects of Visual Form Processing*, World Scientific, p. 430, 1994.
- [92] R. Ogniewicz and M. Ilg, "Voronoi Skeletons: Theory and Applications," *Proc. Conf. Computer Vision and Pattern Recognition*, p. 63, 1992.
- [93] R. Ogniewicz and O. Kübler, "Hierarchic Voronoi Skeletons," *Pattern Recognition*, vol. 28, no. 3, p. 343, 1995.
- [94] K. Palágyi and A. Kuba, "Directional 3D Thinning Using 8 Subiterations," *Lecture Notes in Computer Science*, vol. 1568, pp. 325-336, 1999.
- [95] K. Palágyi and A. Kuba, "A Parallel 3D 12-Subiteration Thinning Algorithm," *Graphical Models and Image Processing*, vol. 61, no. 4, pp. 199-221, 1999.
- [96] D. Perchet, C.I. Fetita, and F. Preteux, "Advanced Navigation Tools for Virtual Bronchoscopy," *Proc. SPIE Conf. Image Processing: Algorithms and Systems III*, 2004.
- [97] S.M. Pizer, D. Eberly, and D.S. Fritsch, "Zoom-Invariant Vision of Figural Shape: The Mathematics of Cores," *Computer Vision and Image Understanding*, vol. 69, no. 1, p. 55, 1998.
- [98] S.M. Pizer, G. Gerig, S. Joshi, and S. Aylward, "Multiscale Medial Shape-Based Analysis of Image Objects," *Proc. IEEE*, vol. 91, no. 10, pp. 1670-1679, 2003.
- [99] S.M. Pizer, D. Fritsch, P. Yushkevich, V. Johnson, and E. Chaney, "Segmentation, Registration, and Measurement of Shape Variation via Image Object Shape," *IEEE Trans. Medical Imaging*, vol. 18, pp. 851-865, 1999.
- [100] S.M. Pizer, A.L. Thall, and D.T. Chen, "M-Reps: A New Object Representation for Graphics," Technical Report TR99-030, 17, Univ. of North Carolina, Chapel Hill, 1999.
- [101] C. Pudney, "Distance-Ordered Homotopic Thinning: A Skeletonization Algorithm for 3D Digital Images," *Computer Vision and Image Understanding*, vol. 72, no. 3, pp. 404-413, 1998.
- [102] J.M. Reddy and G.M. Turkiyyah, "Computation of 3D Skeletons Using a Generalized Delaunay Triangulation Technique," *Computer-Aided Design*, vol. 27, no. 9, pp. 677-694, 1995.
- [103] M. Sabry Hassouna and A.A. Farag, "Robust Centerline Extraction Framework Using Level Sets," *Proc. Conf. Computer Vision and Pattern Recognition*, pp. 458-465, 2005.
- [104] P.K. Saha, B.B. Chaudhuri, and D. Dutta Majumder, "A New Shape Preserving Parallel Thinning Algorithm for 3D Digital Images," *Pattern Recognition*, vol. 30, no. 12, pp. 1939-1955, 1997.
- [105] P.K. Saha and B.B. Chaudhuri, "3D Digital Topology under Binary Transformation with Applications," *Computer Vision and Image Understanding*, vol. 63, no. 3, pp. 418-429, 1996.
- [106] T. Saito and J. Toriwaki, "New Algorithms for Euclidean Distance Transformation of an n-Dimensional Digitized Picture with Applications," *Pattern Recognition*, vol. 27, pp. 1551-1565, 1994.
- [107] G. Sanniti di Baja and S. Svensson, "A New Shape Descriptor for Surfaces in 3D Images," *Pattern Recognition Letters*, vol. 23, p. 703, 2002.
- [108] K. Santilli, K. Bemis, D. Silver, J. Dastur, and P. Rona, "Generating Realistic Images from Hydrothermal Plume Data," *Proc. IEEE Visualization Conf.*, 2004.
- [109] H. Schirmacher, M. Zöckler, D. Stalling, and H. Hege, "Boundary Surface Shrinking—A Continuous Approach to 3D Center Line Extraction," *Proc. Image and Multidimensional Signal Processing Symp.*, pp. 25-28, 1998.
- [110] M.P.P. Schlicher, E. Bouts, and P.W. Verbeek, "Fast Analytical Medial-Axis Localization in Convex Polyhedra," *Proc. Int'l Conf. Pattern Recognition*, pp. 55-61, 1996.
- [111] J.A. Sethian, "Fast Marching Methods," *SIAM Rev.*, vol. 41, no. 2, p. 199, 1999.
- [112] E.C. Sherbrooke, N.M. Patrikalakis, and E. Brisson, "Computation of the Medial Axis Transform of 3-D Polyhedra," *Proc. Int'l Conf. Shape Modeling and Applications*, 1995.
- [113] P. Shilane, P. Min, M. Kazhdan, and T. Funkhouser, "The Princeton Shape Benchmark," *Proc. Shape Modeling Int'l Conf.*, 2004.
- [114] D. Silver and N. Gagvani, "Unwinding the Colon," *Proc. Medicine Meets Virtual Reality (MMVR)*, 2002.
- [115] E. Sorantin et al., "Spiral-CT-Based Assessment of Tracheal Stenoses Using 3D-Skeletonization," *IEEE Trans. Medical Imaging*, vol. 21, no. 3, p. 263, 2002.
- [116] M. Sramek and A.E. Kaufman, "Alias-Free Voxelization of Geometric Objects," *IEEE Trans. Visualization and Computer Graphics*, vol. 5, no. 3, pp. 251-267, July-Sept. 1999.
- [117] M. Straka, M. Cervenansky, A. La Cruz, A. Kochl, M. Sramek, E. Grollier, and D. Fleischmann, "The VesselGlyph: Focus & Context Visualization in CT-Angiography," *Proc. IEEE Visualization Conf.*, 2004.
- [118] H. Sundar, D. Silver, N. Gagvani, and S. Dickinson, "Skeleton Based Shape Matching and Retrieval," *Proc. Shape Modeling Int'l Conf.*, 2003.
- [119] K. Suresh, "Automating the CAD/CAE Dimensional Reduction Process," *Proc. ACM Symp. Solid Modeling and Applications*, 2003.
- [120] S. Svensson, I. Nyström, and G. Sanniti di Baja, "Curve-Skeletonization of Surface-Like Objects in 3D Images Guided by Voxel Classification," *Pattern Recognition Letters*, pp. 1419-1426, 2002.

- [121] S. Svensson, C. Arcelli, and G. Sanniti di Baja, "Finding Cavities and Tunnels in 3D Complex Objects," *Proc. Int'l Conf. Image Analysis and Processing*, pp. 342-347, 2003.
- [122] A. Tal and S. Katz, "Hierarchical Mesh Decomposition Using Fuzzy Clustering and Cuts," *ACM Trans. Graphics*, vol. 22, no. 3, pp. 954-961, 2003.
- [123] A. Telea and A. Vilanova, "A Robust Level-Set Algorithm for Centerline Extraction," *Proc. Eurographics/IEEE Symp. Data Visualization*, 2003.
- [124] A. Telea and J.J. van Wijk, "An Augmented Fast Marching Method for Computing Skeletons and Centerlines," *Proc. Symp. Data Vision*, 2002.
- [125] C. Theobalt, E. de Aguiar, M.A. Magnor, H. Theisel, and H.-P. Seidel, "Marker-Free Kinematic Skeleton Estimation from Sequences of Volume Data," *Virtual Reality Software and Technology*, p. 57, 2004.
- [126] Y.F. Tsao and K.S. Fu, "A Parallel Thinning Algorithm for 3D Pictures," *Computer Vision, Graphics, and Image Processing*, vol. 17, pp. 315-331, 1981.
- [127] A. Verroust and F. Lazarus, "Extracting Skeletal Curves from 3D Scattered Data," *The Visual Computer*, vol. 16, pp. 15-25, 2000.
- [128] B. Vrolijk, F. Reinders, and F.H. Post, "Feature Tracking With Skeleton Graphs," *Data Visualization: The State of the Art*, pp. 37-52, Kluwer Academic, 2003.
- [129] L. Wade and R.E. Parent, "Automated Generation of Control Skeletons for Use in Animation," *The Visual Computer*, vol. 18, no. 2, pp. 97-110, 2002.
- [130] M. Wan, F. Dacheville, and A. Kaufman, "Distance-Field Based Skeletons for Virtual Navigation," *Proc. IEEE Visualization Conf.*, 2001.
- [131] R. Webster, M. Harris, R. Shenk, J. Blumenstock, J. Gerber, C. Billman, A. Benson, and R. Haluck, "Using an Approximation to the Euclidean Skeleton for Efficient Collision Detection and Tissue Deformations in Surgical Simulators," *Medicine Meets Virtual Reality*, vol. 13, IOS Press, 2005.
- [132] F.-C. Wu, W.-C. Ma, R.-H. Liang, B.-Y. Chen, and M. Ouhyoung, "Domain Connected Graph: The Skeleton of a Closed 3D Shape for Animation," *The Visual Computer*, vol. 22, no. 2, p. 117, 2006.
- [133] F. Wu, W.-C. Ma, P. Liou, R. Liang, and M. Ouhyoung, "Skeleton Extraction of 3D Objects with Visible Repulsive Force," *Proc. Eurographics Symp. Geometry Processing*, 2003.
- [134] Y. Xiao, P. Siebert, and N. Werghi, "A Discrete Reeb Graph Approach for the Segmentation of Human Body Scans," *Proc. Int'l Conf. 3D Digital Imaging and Modeling*, p. 378, 2003.
- [135] Y. Yang, O. Brock, and R.N. Moll, "Efficient and Robust Computation of an Approximated Medial Axis," *Proc. Solid Modeling and Applications Conf.*, 2004.
- [136] Y. Zhao, H. Ong, T. Tan, and Y. Xiao, "Intuitive Interfaces for Animation: Interactive Control of Component-Based Morphing," *Proc. ACM SIGGRAPH*, 2003.
- [137] Y. Zhou, A. Kaufman, and A.W. Toga, "Three-Dimensional Skeleton and Centerline Generation Based on an Approximate Minimum Distance Field," *The Visual Computer*, vol. 14, pp. 303-314, 1998.
- [138] Y. Zhou and A.W. Toga, "Efficient Skeletonization of Volumetric Objects," *IEEE Trans. Visualization and Computer Graphics*, vol. 5, no. 3, pp. 196-209, July-Sept. 1999.
- [139] Online resource: <http://www.caip.rutgers.edu/~cornea/CurveSkeletons>, 2006.



Nicu D. Cornea received the BSc degree in computer engineering from the Technical University of Cluj-Napoca, Romania, in 2001 and the MSc degree in electrical and computer engineering from Rutgers University, New Jersey. Currently, he is a PhD student in the Department of Electrical and Computer Engineering at Rutgers University. His research interests include volume visualization and graphics.



Deborah Silver received the BS degree from Columbia University, School of Engineering and the PhD degree from Princeton University in computer science. She is a professor in the Department of Electrical and Computer Engineering at Rutgers, The State University of New Jersey. Her research interests include visualization, volume graphics, and computer graphics. She is a member of the IEEE.



Patrick Min received the MSc degree in computer science from Leiden University, The Netherlands, in 1993 and the PhD degree in computer science from Princeton University, New Jersey, in 2004. Next, he was a postdoctoral researcher at Utrecht University, The Netherlands. Currently, he is an assistant professor of computer science and mathematics at the American University of Rome. His research interests are in 3D modeling, 3D shape semantics, and 3D model retrieval. He is a member of the Eurographics Association. For more information, see <http://www.google.com/search?q=patrick+min>.

► For more information on this or any other computing topic, please visit our Digital Library at www.computer.org/publications/dlib.