Published in final edited form as:

SIAM J Sci Comput. 2010 March 1; 32(2): 947–969. doi:10.1137/090767170.

LOCAL ORTHOGONAL CUTTING METHOD FOR COMPUTING MEDIAL CURVES AND ITS BIOMEDICAL APPLICATIONS

Xiangmin Jiao^{*}, Daniel R. Einstein[†], and Vladimir Dyedov[‡]

^{*}Dept. of Applied Mathematics & Statistics, Stony Brook University, Stony Brook, NY.
jiao@ams.sunysb.edu

[†]Biological Monitoring & Modeling, Pacific Northwest National Laboratory, Richland, WA.
daniel.einstein@pnl.gov

[‡]Dept. of Applied Mathematics & Statistics, Stony Brook University, Stony Brook, NY.
vladimir@ams.sunysb.edu

Abstract

Medial curves have a wide range of applications in geometric modeling and analysis (such as shape matching) and biomedical engineering (such as morphometry and computer assisted surgery). The computation of medial curves poses significant challenges, both in terms of theoretical analysis and practical efficiency and reliability. In this paper, we propose a definition and analysis of medial curves and also describe an efficient and robust method called *local orthogonal cutting (LOC)* for computing medial curves. Our approach is based on three key concepts: a local orthogonal decomposition of objects into substructures, a differential geometry concept called the *interior center of curvature (ICC)*, and integrated stability and consistency tests. These concepts lend themselves to robust numerical techniques and result in an algorithm that is efficient and noise resistant. We illustrate the effectiveness and robustness of our approach with some highly complex, large-scale, noisy biomedical geometries derived from medical images, including lung airways and blood vessels. We also present comparisons of our method with some existing methods.

Keywords

Curves and surfaces; medial curves; orthogonal decomposition; interior center of curvature; stability; biomedical applications

1. Introduction

The subject of this paper is the construction of a 1-D representation (commonly known as the *curve skeleton*) of a 3-D object that is mostly composed of tubular sub-structures. Curve skeletons have a number of variants [6]. We define and compute a special type of curve skeleton, which we refer to as the *medial curves*.

Medial curves and other curve skeletons have a wide range of applications in geometric modeling and analysis, such as shape matching and shape retrieval, which are surveyed thoroughly in [6]. The motivating applications for this work are from the field of biomedical engineering, where medial curves are important for multiscale modeling [11], morphometry [9,32], physiology [22], image analysis [19], and computer assisted surgery [31]. For example,

the medial curves can form the basis of a morphometric analysis of tree-structured airways in the lung [9] or coronary vessels in the heart [12], by providing statistics for segment length, diameter, and bifurcation angle. Since the medial curves embed the topology of the tree and distill its geometry, these data can then be analyzed for fractal relationships that reveal genetic decision making in morphogenesis, pathological remodeling or anatomic variation. Furthermore, in multiscale modeling, these data can form the basis of physiological multiscale boundary conditions — in the form of lower-dimensional differential equations — that represent the biophysics of subsets of the vessel or airway tree at a scale that is not practical to represent as a 3D continuum [11]. In endoscopy or virtual surgery, the medial curves can help to pilot diagnostic devices to a specific location in the airway or vessel tree [31]. In addition, the medial curve can also be used for automatic identification of the outlets in biological geometries for imposing boundary conditions in numerical simulations [?].

From these biomedical applications, we summarize the following key requirements for the medial curve:

Topological and structural correctness: The medial curve should be *homotopic* to the object, and should represent the *structure* of the object by correctly labeling the *ends* and *branches*.

Smoothness: The medial curve should be *smooth* except at branches and distinct sharp turns.

Centeredness: The medial curve should be near the “*centers*” of the object.

In addition, an algorithm for computing medial curves must satisfy the following requirements.

Robustness: The algorithm must handle complex geometries with tens of thousands of branches without user intervention.

Noise resistance: The algorithm should tolerate moderate level of noise, which are unavoidable in image-derived geometries.

Efficiency: The algorithm must be fast and have a small memory footprint (i.e., have close to linear complexity in time and space) to handle large datasets with millions of triangles.

The objective of this paper is to define the medial curve and construct algorithms that satisfy the above requirements. We focus on imaging-derived biomedical geometries. However, the above requirements also apply to some other application areas (such as virtual navigation), to which our methodology will also apply. We assume the input geometry is given by a surface triangulation, such as that obtained by an isosurfacing algorithm (see [26] for a survey of such algorithms). The decision of using surface triangulations (instead of using volumetric data) is for the considerations of accuracy, efficiency, and application requirements, as we explain in more detail in Section 2.

In this paper, we take a novel perspective on defining and computing medial curves, and make a few contributions toward the better understanding, and more robust computation, of medial curves. Our first contribution is a concept called the *interior center of curvature* (ICC) of surfaces. For canonical shapes (such as spheres, cylinders and tori) and more generally the *canal surfaces* [10, Sec. 20.2], the ICCs of the surface give precisely their medial curves. For general surfaces, we define the *average interior center of curvature* (AICC) as approximations of ICC with improved noise resistance, and use AICCs as a starting point for finding the medial curves.

Our second contribution is to define the medial curve based on a *local orthogonal decomposition* of the object into tips, segments, and junctions. We define this decomposition

based on geometric quantities (including AICC and orthogonality) and topological invariance (the Euler characteristic). The medial curve is then the graph connecting the centers of these substructures and of the cutting planes between them.

Our third contribution is a new algorithm for the computation of medial curves, called *local orthogonal cutting (LOC)*. Our algorithm is robust and resistant to noise, and it has nearly linear time and space complexities. We demonstrate the effectiveness of our method for complex biomedical geometries derived from medical images, including lung airways and blood vessels, and present comparisons of our method with some existing methods.

The remainder of the paper is organized as follows. In Section 2, we briefly review some previous work on medial curves and curve skeletons. In Section 3, we introduce a definition of medial curves based on a decomposition of the objects. In Section 4, we present our algorithm for efficient and robust computation of medial curves. In Section 5, we describe the use of the medial curve constructed by our algorithm for some biomedical applications. In Section 6, we present some experimental examples for complex geometries with our method. Finally, Section 7 concludes the paper with a discussion of future research directions.

2. Previous Work

There is a vast literature on extracting curve skeletons or similar 1-D curves from an explicit or an implicit representation of an object. A complete survey is beyond the present scope. Instead, we review a few references that are representative. For clarity, we divide the approaches into two categories: algorithms that are based on the volume, and those based on the surface.

2.1. Volume-Based Approaches

Most volumetric methods begin with a binary voxelized image, derived from MRI (magnetic resonance) or CT (computed tomography) scans. Among these methods, *thinning* is probably the most popular paradigm for computing skeletons, due to its relative simplicity and efficiency. A thinning algorithm typically iteratively erodes the contours in a binary image until a single layer of pixels remain, and these pixels are then connected to form the skeleton. The process is based on Blum's grass-fire analogy of the medial axis [4]. There are many variants of thinning algorithms. See for example [17] for a survey on thinning in 2-D and see [18,27] for algorithms in 3-D. Because the resolution is quantized to the voxel size, the precision of these methods are limited, so the resulting curves in general is not smooth and has limited accuracy. In addition, they are sensitive to the noise in the image. This sensitivity is often manifested as spurious branches, leading to incorrect structures. In [30], smoothness is improved by adding a shrinking step before thinning and sensitivity to noise is alleviated by a pruning step, but centeredness and accuracy of the skeletons may still suffer.

Another class of method is distance transform, which computes distance field for each interior point or voxel center (see e.g. [20]). Ridges of the distance field are detected, yielding an approximate medial surface, which is then pruned. Similar to other voxel-based methods, the accuracies of these methods are constrained by the voxel resolution. In addition, these methods can be sensitive to noise due to the instability of the medial axis. To alleviate the sensitivity to noise and improve accuracy, Cornea et al. proposed to compute the distance field based on a simulated electric potential [5]. However, their method is computationally prohibitive for practical image sizes. Wischgoll et al. proposed a variation of this approach, which positions discrete points on the boundary of the object with subvoxel precision based on intensity gradients in the voxelized image [32]. This approach considerably reduces the computational overhead, but it does not guarantee topological correctness.

2.2. Surface-Based Approaches

In this paper, we compute the medial curve from a surface triangulation. Surface-based approaches have two potential advantages over volumetric approaches. First, they are not limited by the voxel resolution. Second, they can potentially be much more efficient, as the number of surface points is usually orders of magnitude smaller than the number of image voxels for large datasets. The most common class of surface-based approaches begins by computing the medial surface from the Voronoi diagrams [2]. The Voronoi-based medial surfaces are then distilled into medial curves or medial skeletons by either clustering points, or by extracting the internal edges of the Voronoi diagram [8]. This approach is quite expensive. In addition, since small perturbations in the original surface can induce large perturbations in the medial axis, this approach is sensitive to noise, and the resulting medial curves are in general not smooth. In addition, approaches based on the medial surfaces can produce structurally incorrect medial curves for many biomedical applications. For example, the stable medial-axis based skeleton of a cylinder is a centered line that has four legs reaching out at 45 degrees to the top and bottom of the cylinder, while the desired solution is a straight, centered line that extends all the way from one end of the cylinder to the other.

Another class of surface-based approaches involves mesh contraction. For example, in a recent approach meshes are operated on directly by shrinking the boundary to a 1D graph, by applying a curvature flow and implicit Laplacian smoothing with global positional constraints [3]. This approach produces medial curves that are homotopic to the object, and it is reasonably efficient. However, the endpoints tend to retract into the geometry, and the medial curve tends not to be well centered.

Another type of method is based on Reeb and Morse graphs. Such methods are typically very efficient and can encode the correct surface topology [28], and they are closely related to many mesh segmentation methods. Typically, level set functions are defined on the surface and stable feature points are defined as the extrema of these so-called mapping functions. Various other mapping functions have been defined for Reeb-graph methods, but it is in general difficult to define mapping functions that yield a geometrically correct skeleton. It is also difficult to transform a topological graph into a geometrically accurate skeleton [29]. Another approach to Reeb-graphs is to identify feature points on the surface before defining mapping functions by locating vertices at the extremities of surface components [14,23]. However, stable feature point identification is challenging by itself and may be sensitive to noise.

Other related surface based approaches examine contours of intersection with the surface rather than contours of a mapping function [24,25]. These methods are based on the observation that tubular networks can be decomposed into tubular and non-tubular regions by examining the intersection curves of inflated spheres. In the first of these, the spheres were centered at the vertices [24]. The study of the evolution of the intersection curves, γ_i , as function of the radius in terms of the number of connected components i , determines the local topology. Where $i = 1$ the neighborhood will be an extremity; where $i = 2$ the neighborhood will be tubular; where $i \geq 3$ the neighborhood will be a branch. Since every point on a medial curve is an endpoint, a segment or a branch point, the connection to medial curves is obvious. However, the sizes of the neighborhoods must be subjected to application specific parameters due to multiple ambiguities. In [25], only tubular sections were extracted, by first finding two intersection curves from spheres inflated at seed vertices, termed medial loops. The medial loops were then swept along the tubular section by collocating spheres located at the centroids of the medial loops rather than the seed vertices.

3. Geometry and Topology of Medial Curves

In this section, we present some basic geometric and topological concepts. We consider a volumetric object Ω embedded in $\mathbb{R}^3$. The *surface* is the boundary of the object, denoted by $\partial\Omega$, so it is by definition a 2-manifold without boundary. We will define the *medial curve* of the object Ω (or equivalently, the *medial curve* of its surface $\partial\Omega$). We assume the surface is given by a triangulation. By convention, surface normals are outward, so the curvature is positive if the object is locally concave and is negative if it is locally convex. Without loss of generality, we assume the object contains a single connected component. For an object with multiple connected components, all of our discussions apply to each connected component of the object.

3.1. Canal Surfaces and Centers of Curvature

To define the medial curve, we first consider a special but important class of surfaces called *canal surfaces* [10, Sect. 20.2], which are closely related to the medial axis and centers of curvatures.

3.1.1. Canal Surfaces and Medial Axis—A canal surface is the envelope of a 1-parameter family of spheres in $\mathbb{R}^3$. The notion of canal surface is quite general: all tubes and most surfaces of revolution are canal surfaces. The curve formed by the centers of these spheres is called the *center curve* of the canal surface [10, page 646]. Figure 3.1 shows some examples of canal surface and their center curves. In this paper, we refer to the center curve as the *medial curve*, as it is a special case of the *medial axis* for general surfaces, as we will explain next.

Although the medial axis is typically explained using the grass-fire analogy (see Section 2), it was originally motivated by the *medial axis transformation* [4], which is the process of computing the medial axis and the local feature size (lfs) and then approximating the surface (or curve) using the envelope of the spheres (or circles) that are centered on the medial axis and have radii equal to the lfs. The local feature size is therefore also called the *radius* of the object. If the medial axis is a curve (namely, the *medial curve*), then the envelope of these spheres is either a canal surface (if the medial curve is a closed curve) or a collection of canal surface patches (if the medial curve has branches). Many geometries, especially biomedical geometries, can be well approximated as a union of canal surfaces (more precisely, the union of the volume bounded by canal surfaces). Therefore, it is important and valuable to first understand canal surfaces and their medial curves. For more complex geometries, we will develop numerical approximations and introduce other measures to address the additional issues such as junctions and open-ends.

3.1.2. Interior Center of Curvature—To help understanding canal surfaces, we introduce some new concepts. The *interior curvature* (IC) is the most negative curvature at a point. We refer to the reciprocal of the absolute value of IC as the *interior radius of curvature* (IRC). The point that has a distance equal to IRC along the inward surface normal of $\mathbf{p}$ will be called the *interior center of curvature* (ICC). More precisely, given a point $\mathbf{p} \in \partial\Omega$, let $\hat{\mathbf{n}}_{\mathbf{p}}$ denote the unit outward surface normal to Γ at $\mathbf{p}$, κ_{p1} and κ_{p2} denote the principal curvatures at $\mathbf{p}$. Then the *interior curvature*, denoted by $\kappa_{\mathbf{p}}^-$, is the more negative value of κ_{p1} and κ_{p2} , i.e.,

$$\kappa_{\mathbf{p}}^- = \begin{cases} \kappa_{p1} & \kappa_{p1} < 0 \text{ and } \kappa_{p1} \leq \kappa_{p2}, \\ \kappa_{p2} & \kappa_{p2} < 0 \text{ and } \kappa_{p2} < \kappa_{p1} \\ -\varepsilon & \kappa_{p1} \geq 0 \text{ and } \kappa_{p2} \geq 0 \end{cases} \quad (3.1)$$

where ε is an infinitesimal number (e.g., 10^{-100} for double-precision floating-point arithmetic) and is introduced to avoid division by zero. The *interior center of curvature* (ICC) at $\mathbf{p}$ is then

$$\mathbf{c}_p = \mathbf{p} + \frac{1}{\kappa_p^-} \widehat{\mathbf{n}}_p. \quad (3.2)$$

The ICC is the center of an *osculating sphere*, which is tangent to the surface at $\mathbf{p}$ and has a radius equal to the IRC.

The above definitions of IRC and ICC seem to be new and are applicable to any smooth surfaces. For a canal surface Γ , the IRC is the local feature size of the surface, and the IRCs form the medial curve of Γ . The osculating sphere intersects a canal surface Γ at a circle, which we refer to as an *intersection circle*. If the radius of the canal surface is constant, then this circle is a principal curve and is orthogonal to Γ (see Theorem 20.14 in [10, page 649]).

3.1.3. Average Interior Center of Curvature—The curvatures are second-order differential quantities, so the ICC are inherently sensitive to noise. To improve noise resistance for surface triangulations, we define the *average interior center of curvature* (AICC) of a surface patch sampled by a discrete point set $P = \{\mathbf{p}_1, \mathbf{p}_2, \dots, \mathbf{p}_m\}$ to be a weighted average of the ICCs at these points, i.e.,

$$\bar{\mathbf{c}} = \frac{\sum_{i=1}^m w_i \mathbf{c}_i}{\sum_{i=1}^m w_i}, \quad (3.3)$$

where $\mathbf{c}_i$ denote the ICC at point $\mathbf{p}_i$. We choose the weight w_i to be $(\kappa_i^-)^2$ times a control area a_i of the vertex i (where the control area is chosen to be one third of the sum of the areas of the incident triangles of $\mathbf{p}_i$). This weighting scheme gives higher priorities to more curved areas and lower priorities to flatter areas. Therefore, it tends to underestimate the average radius of curvature and in turn stabilizes the AICC. For better robustness, we sometimes set the weights to zeros for the points that are “invisible” from a particular viewpoint, as we will explain in Section 4. The AICC of a sphere is exactly the center of the sphere. The AICC of a small patch of a tube is close to the medial axis, and it converges to a point on the medial axis as the size of the patch tends to zero. Therefore, AICC preserves the key features of ICC while allowing better noise resistance.

3.1.4. AICC-Centered Minimum Containing Sphere—Besides ICC, the osculating sphere is an important concept for canal surfaces. An osculating sphere intersects a canal surface at a circle. Such an intersection is unstable to compute in the presence of numerical errors. For better stability, we approximate an osculating sphere by a sphere that is centered at the AICC with a slightly larger radius, such that this enlarged sphere would intersect the surface at a finite area. We refer to the enlarged sphere as the *AICC-centered minimum containing sphere* (MCS) of the area, and it will be useful in our computation of medial curves later.

3.2. Local Orthogonal Decomposition

Our objective now is to define the medial curves for surfaces that are mostly composed of canal surfaces. We propose a decomposition of an objects into three different types of sub-structures, and the definition of medial curves will naturally follow from this decomposition.

3.2.1. Decomposition of Objects—Suppose a given object is mostly composed of a set of canal surfaces, referred to as the *component canal surfaces* (CCS). Imagine moving a planar knife to cut the object through the intersection circles of the CCSes. We refer to the cross-

section of the knife with the object as a *cut*. In general, the cuts do not intersect with each other, and any point in the object belongs to at most one cut. If the medial axis γ of one CCS is not closed, then the cut corresponding to an open-end of γ cuts off a *tip* of the object. At an intersection of multiple CCSes, the cutting planes are ill-defined, but such an intersection may be identified by the well-defined cuts from their adjacent CCSes.

In summary, the above process decomposes an object into three types of sub-structures: the *segments*, each of which is composed of continuous cuts and bounded by two cuts, the *tips*, each of which is incident on one cut, and the *junctions*, each of which is incident on three or more cuts. The cuts also decompose the surface of the object into surface patches, and for convenience we refer to these surface patches as tips, segments, and junctions as well. Figure 3.2 shows an illustration of these three types of local structures and their corresponding cuts.

Given this above decomposition, we define the *medial curve* of the surface to be the union of the medial curves of the segments as well as the edges that connect the “junction centers” with the end-points of the medial curves of their adjacent segments. This decomposition also determines the topology of the medial curve: the *end-points* of medial curve correspond to the tips, and the *bifurcations* and *multi-furcations* of the medial curve correspond to the junctions. The key of computing the medial curve is then the computation of such a decomposition of the object, which we describe in the next subsection.

Note that for a given object, there are a constant number of tips and junctions, but there can be an arbitrary number of cuts. For efficiency, it is desirable to have the lengths of the segments to be proportional to the local diameter of the object and potentially be dependent on the curvature, so that the medial curve of each segment can be approximated by a line segment, and the number of edges in the medial curve is independent of the resolution of the surface triangulation.

3.2.2. Approximation by Orthogonal Cuts—Computing the exact intersection circles is a difficult and potentially unstable process. For simplicity, we assume the radii (i.e., the local feature sizes) of the component canal surfaces do not change rapidly, so that the cutting plane are nearly orthogonal to the surface and the ICCs are approximately at the centroid of the cutting plane. Thereafter, we can locate the cuts by computing cutting planes that are as orthogonal to the surface as possible. This concept of near orthogonality will lend itself to robust numerical optimization techniques.

A plane is uniquely determined by a point and a normal direction. We will determine the points based on the AICC and the centroid. For the normal direction, we measure orthogonality using two different types of quantities: *angle-based* and *eigenvalue-based*. Given a plane P , let $\hat{n}_P$ denote the unit normal to P , and $\hat{n}_\tau$ denote the unit face normal of a triangle τ . The dihedral angle between P and τ is $\arccos |\hat{n}_P^T \hat{n}_\tau|$, which ideally should be 90° but can be as small as 0° . We therefore define the angle-based measure as the minimum dihedral angle between P and the triangles intersecting P .

The minimum dihedral angle is useful for measuring the quality of a given plane, but by itself it cannot facilitate optimizing orthogonality. For the latter purpose, we introduce an eigenvalue-based measure. Consider the matrix

$$N = \sum_{\tau} w_{\tau} \hat{n}_{\tau} \hat{n}_{\tau}^T, \quad (3.4)$$

which we refer to as the *normal covariance matrix*. If the plane P intersects a cylinder nearly orthogonally, then all the normals $\hat{n}_t$ are coplanar, and N is a rank-two matrix. Let σ_2 and σ_3 denote the two smaller eigenvalues of N , and then $\rho = \sigma_3/\sigma_2$ is then a measure of how close the surface is to a cylinder locally. In general, ρ is between 0 and 1, and the smaller the better. If ρ is sufficiently small, then the eigenvector of N corresponding to σ_3 may provide a better normal for the plane.

The angle-based and eigenvalue-based measures complement each other in algorithm design. The former is a measure of a given plane, whereas the latter is a measure for potentially a better plane, and the corresponding eigenvectors can facilitate optimizing the normals of the plane. Both of these measures are numerical in nature, so some tolerances are needed, which we will discuss in Section 4.2.2.

3.3. Topological Invariants

Besides geometric measures, important information about the local structure of the object can be obtained by analyzing the topology of a surface patch P . In particular, we classify P based on whether it is connected, whether it is closed, and whether it has tunnels. To capture the first two criteria, we consider the number of connected components of P and the number of connect components of the border curves of P and a simple counting. The definitions of these quantities are trivial, and their computations require only a breadth-first search. For the third criterion, we must consider the Euler characteristic of P . Given a polygonal surface (such as a triangulated surface), the *Euler characteristic* is the alternating sum:

$$\chi \equiv \#\text{vertices} - \#\text{edges} + \#\text{faces}. \quad (3.5)$$

For a tip, segment, or junction, the surface patch is an open surface. The Euler characteristic is 1 for the surface of a tip, 0 for a segment, and a negative number for a junction. Therefore, χ gives an alternative means for classifying tip, segment, and junction. In the next section, we present an algorithm for decomposing an object using the AICC, orthogonality, and Euler characteristic.

4. Computation of Medial Curves

In the preceding section, we defined the medial curves based on a decomposition of an object into tips, segments, and junctions, and introduced measures for these local structures and for the cuts between them. We hereafter present an algorithm, called *local orthogonal cutting (LOC)*, based on these intuitive definitions. At a high level, our algorithm has the following five steps: 1) compute the differential quantities, 2) identify stable tips, 3) identify other patches, 4) fine-tune cuts and segments, and 5) reconstruct the medial curve. We describe each of these steps in the following subsections and present the analysis, data structures, and some implementation issues in Section 4.6.

4.1. Computing Differential Quantities

The identification of candidate cuts relies on the concept of interior center of curvature. Thus, we begin by computing the geometric quantities of the surface, including the normals and principal curvatures at the vertices as well as the normals and areas of the triangles. The surfaces in our applications are inherently noisy, as they are typically derived from medical images. For stability and noise resistance, we compute these vertex quantities by applying a quadratic fitting over the two-ring neighborhood of each vertex (i.e., the set of vertices that are within two edges away from the vertex) and solve the fitting using the weighted least squares framework in [13]. From the normals and principal curvatures, we obtain the interior curvature (IC) and the

interior center of curvature (ICC) using Eqs. (3.1) and (3.2), and then compute the AICC of a face incident on each vertex using Eq. (3.3) for improved noise resistance.

Algorithm 1

Identify stable tips.

```

1: for each convex seed vertex  $v$ , in ascending order of IRC
2:   if  $v$  has been visited; then continue
3:    $S \leftarrow$  AICC-centered MCS of a face incident on  $v$ ;
4:    $\chi_{old} \leftarrow 1$ ;
5:   for  $i = 1, 2, \dots$ 
6:     obtain patch  $\Gamma$  in  $S$  and cut  $P$ ;
7:     fill in small holes in  $\Gamma$ ;
8:      $\chi \leftarrow$  Euler characteristic of  $\Gamma$ ;
9:     if  $\chi \neq 1$  and  $\chi_{old} = 1$ 
10:       save non-disk patch;
11:     elseif  $\chi \neq 1$ 
12:       resort to saved tip; break;
13:     else
14:       if  $P$  is highly orthogonal to  $\partial\Omega$ 
15:         save patch as tip; break;
16:       elseif  $P$  is more orthogonal than last saved
17:         save  $P$  and patch as tip for resort;
18:       end if
19:     end if
20:      $S \leftarrow$  MCS of  $\Gamma$  centered at its AICC;
21:      $\chi_{old} \leftarrow \chi$ ;
22:   end for
23:   if any tip was saved
24:     mark triangles intersecting or below  $P$  by tip ID;
25:     mark vertices of the above triangles as visited;
26:     associate cut with a seed face incident on border edge;
27:   else
28:     mark vertices in saved non-disk patch as visited;
29:   end if
30: end for

```

4.2. Identifying Stable Tips

As defined in Section 3, a tip is a topological disk in the surface. However, not every disk corresponds to a tip. The objective of this step is to determine the disks that are *stable tips*, which are critical for the stability of the medial curve. Our goal is to avoid false positiveness while being as accurate as possible. Due to its complexity, we outline a high-level pseudocode of this step in Algorithm 1. The procedure has some key aspects, which we describe in more detail as follows.

4.2.1. Selecting and Sorting Seed Vertices—To identify stable tips, we select a dense sampling of mesh vertices as seeds, based on their IRCs. We need at least one seed per tip. A tip must have a vertex that is convex. Therefore, we require each seed vertex to be convex, i.e., whose principal curvatures are both negative. To reduce the number of seeds, we impose an upper bound on the IRC of seed vertices. A tight upper bound would be the local radius of the object in the neighborhood near the vertex, but estimating the local radii is overly complex and expensive for our purpose. Instead, we set the upper bound to be half of the minimum dimension of the bounding box of the object.

The ordering of seed vertices is also important. With an arbitrary ordering, a large feature might subsume a smaller one and in turn cause small but important tips to be missing. Thus, we sort the seed vertices in ascending order of IRC (cf. line 1 of Algorithm 1), so that smaller features would be processed first. We go through the seed vertices one by one, and skip those that have been marked as visited in earlier iterations (cf. lines 2, 25 and 28). In the following we describe the sub-steps for each individual seed vertex.

4.2.2. Growing Candidate Patch—Starting from a seed vertex, we grow a candidate patch to seek a stable tip. We obtain a candidate patch starting from an incident face of the seed vertex, and grow the patch through the aid of the MCS of the candidate patch centered at its AICC (cf. lines 3, 6 and 20). To increase the growth rate, we enlarge the MCS slightly (typically by 5%). To guard against strongly bended areas, we include a simple “visibility condition” in the computation of AICC (lines 3 and 20). Specifically, let $\mathbf{o}$ denote the AICC of the surface patch in the previous iteration. Given a vertex $\mathbf{p}$ in the surface patch, let $\hat{\mathbf{n}}$ denote the outward unit normal at $\mathbf{p}$. We set the weight for $\mathbf{p}$ to 0 in (3.3) if $\hat{\mathbf{n}}^T(\mathbf{p} - \mathbf{o}) \leq 0$.

In line 6 of Algorithm 1, we obtain the triangles that intersect or fall within the sphere and that at the same time are not yet assigned to any tip. These triangles are found efficiently through a breadth-first search on the dual graph of the mesh. To reduce sensitivity of the topology to small perturbations, we fill in small holes in the candidate patch (cf. line 6), including the holes that each correspond to a single face, two adjacent faces along an edge, or the one-ring neighbor of a vertex (or more concisely, the *star* of a face, edge, or vertex). From the candidate patch, we also obtain a candidate cutting plane P that is nearly orthogonal to the surface. The determination of the cutting plane is fairly involved, which we defer to Section 4.2.3.

We continue to grow a candidate patch until it reaches some *steady state* topologically and geometrically, so that a small perturbation would less likely affect the qualitative behavior of the medial curve. Topologically, we consider a patch to be non-disk only if it is non-disk for at least two consecutive iterations (cf. lines 9–12). This persistency requirement avoids the situation where a non-disk is introduced due to a small perturbation during an intermediate step. A variable χ_{old} was introduced to facilitate this persistency checking (cf. lines 4, 9, and 21). Geometrically, we consider the orthogonality of the cut, measured by the minimum dihedral angle as defined in Section 3.2.2. Our basic idea is to look for the patch with the most orthogonal cut (cf. lines 16–17). For better efficiency, we consider the plane as *highly orthogonal* if the minimum angle is close to 90° (say $\geq 75^\circ$). A topological-disk patch with a highly orthogonal cut is accepted as a tip immediately (cf. lines 14–15). For robustness, we consider a cut as *not orthogonal* if the minimum angle is smaller than 45° , and we impose an upper bound on the number of iterations of the growth (cf. line 5). In practice, we found an upper bound of 20 to 30 works well. After identifying a tip, we label the triangles that intersect or are below the cut within the candidate patch by a unique integer ID for the tip (cf. lines 24). In addition, we associate the cut with a seed face that is incident on the border edge of the surface patch (cf. lines 26). This seed face will be used in the next step for determining the connectivity of the tips with adjacent patches.

4.2.3. Computing and Optimizing Cut—To determine stable tips, the most involved and critical sub-step is the determination of the cut for a given candidate patch, as described in more detail as follows. This procedure is employed in line 6 of Algorithm 1. It will also be a basic building block in later steps of our overall algorithm.

Given a candidate patch Γ that is a topological disk, we determine a cutting plane in an iterative manner starting from its border curve. A cutting plane is uniquely determined by its normal direction and a point, which we refer to as an *anchor point* of the plane. To determine a candidate cut, we first determine a normal direction using a principal component analysis of the border curve $\partial\Gamma$. In particular, let $\mathbf{c} = \int_{\partial\Gamma} \mathbf{x} ds / \int_{\partial\Gamma} ds$ be the centroid of the border curve, where s denotes a parametrization of $\partial\Gamma$. The matrix

$$\mathbf{A} = \int_{\partial\Gamma} (\mathbf{x} - \mathbf{c})(\mathbf{x} - \mathbf{c})^T ds \quad (4.1)$$

is the covariance matrix of the curve. We obtain an initial normal direction $\mathbf{n}_0$ to the plane as the third eigenvector of $\mathbf{A}$ (i.e., the eigenvector corresponding to the smallest eigenvalue of $\mathbf{A}$). Let $\mathbf{s}$ denotes the area-weighted average of the centroids of the triangles incident on $\partial\Gamma$. We choose the sign of $\mathbf{n}_0$ to satisfy $(\mathbf{c} - \mathbf{s})^T \mathbf{n}_0 > 0$, so that $\mathbf{n}_0$ points outwards with respect to the region between Γ and the plane.

Algorithm 2

Optimize position and orientation of cut starting from base cut $P_0 = (\mathbf{p}_0, \mathbf{n}_0)$ and desired center $\mathbf{o}$.

```

1:  $\mathbf{p} \leftarrow \mathbf{o}$ ;
2: for  $i = 1, 2, \dots$ 
3:    $P \leftarrow (\mathbf{p}, \mathbf{n}_0)$ ;
4:   for  $j = 1, 2, \dots$ 
5:     Compute  $\tilde{\mathbf{n}}$  from normal covariance matrix of  $P \cap \Gamma$ ;
6:      $\tilde{P} \leftarrow (\mathbf{p}, \tilde{\mathbf{n}})$ ;
7:     if  $\tilde{P}$  is safe and is more orthogonal than  $P$ 
8:        $P \leftarrow \tilde{P}$ ;
9:     else
10:      break;
11:   end if
12: end for
13: if  $P$  is orthogonal
14:   return  $P$ ;
15: end if
16:  $\mathbf{p} \leftarrow (\mathbf{p} + \mathbf{p}_0)/2$ ;
17: end for
18: return base cut  $P_0$ ;
```

After obtaining the normal, we select an anchor point for the plane so that all the triangles incident on the border curve are just above the plane. This condition ensures that the surface patches corresponding to different tips would not overlap at any vertex. It will be important in ensuring the remainder of the surface can be decomposed into valid patches, as we will explain

shortly. We consider the plane as the *base cut* if it is orthogonal to the surface, and in addition if the AICC is below the plane. If any of these conditions is violated, we cannot extract a cut and instead would continue to grow the patch.

After determining a base cut, we further optimize its location and orthogonality. Algorithm 2 shows the pseudocode for this procedure. Ideally, the cut should contain the AICC, but there may not exist a plane passing through the AICC while being orthogonal to the surface. For robustness, we instead search for an orthogonal cut that is as close to the AICC as possible. This requires adjusting both the position as well as the orientation of the cut. We achieve this in a double loop: In the outer loop, we perform a bisection procedure to adjust the position, and in the inner loop we adjust the orthogonality.

We first describe the inner loop. Our key idea is to use the normal covariance matrix to obtain an improved normal direction. In particular, let γ denote the intersection curve of a candidate plane P with the surface patch Γ . The normal covariance matrix is $N = \int_{\gamma} \hat{n} \hat{n}^T ds$, computed as in (3.4). Consider the eigenvalue decomposition of N . If the smallest eigenvalue of N is far smaller than the other two (in practice, $\lambda_3 \leq 0.5\lambda_2$), let $\tilde{n}$ be the third eigenvector of N with a sign such that $\tilde{n}^T n_0 > 0$. Let $\tilde{P}$ denote a plane centered at the same anchor point as P but with normal $\tilde{n}$. We check whether $\tilde{P}$ intersects only the interior triangles of Γ (i.e., the triangles not incident on $\partial\Gamma$), and whether the triangles intersecting and below $\tilde{P}$ constitute a disk. If $\tilde{P}$ satisfies both of these conditions, it is then considered *safe*. We accept $\tilde{P}$ as an improved plane if it is safe and it is more orthogonal to the surface than P . We repeat the inner loop for up to four iterations to obtain the most orthogonal plane.

Algorithm 3

Identification of remaining patches.

```

1: for each seed vertex  $v$ , in ascending order of IRC
2:   while any incident face  $\sigma$  of  $v$  is not assigned
3:      $S \leftarrow$  MCS of  $\sigma$  centered at its AICC;
4:     obtain patch  $\Gamma$  in  $S$ ;
5:     for  $i = 1, 2, \dots$ 
6:        $\chi \leftarrow$  Euler characteristic of  $\Gamma$ ;
7:       if  $\chi = 1$ 
8:         identify cut or grow AICC-centered MCS;
9:       else
10:        for each border curve  $\gamma$  of  $\Gamma$ 
11:          identify cut or grow sphere at  $\gamma$ ;
12:        end for
13:        if  $\chi < 0$ 
14:          move cut inwards toward junction center;
15:        end if
16:        end for
17:      end while
18:      if patch has more than three cuts
19:        attempt to decompose junction;
20:      end if
21:      mark triangles of the patch by patch ID;

```

```

22:   associate each cut of patch with a seed face;
23: end for

```

In the outer loop, we start by trying to place the anchor point at the AICC and improving the orthogonality by invoking the inner loop. Let p denote the anchor point of the current trial plane. If the trial plane is not orthogonal, we then move the anchor point of the trial plane to the mid-point between p and the anchor point of the base cut. This process is repeated for a few (typically up to four) iterations. If none of the trial planes is orthogonal enough, we then accept the base plane.

4.3. Identifying Other Patches

After finding stable tips, we then identify the other patches in the remainder of the surface. These other patches include segments, junctions, and potentially some tips that were designated as unstable in the previous step. This step shares some similarities with the previous one. However, there are also some key differences in terms of seed selection, growth of patches, and stopping criteria. In addition, in this step we must address the additional complexity of having potentially too many branches in a junction. We hereafter describe the algorithm and focus on these key issues. Algorithm 3 outlines a high-level pseudocode for this step.

4.3.1. Selecting and Sorting Seeds—We search for the patches starting from a series of seed vertices. A junction or segment may be free of convex points, but it cannot be composed of only concave points. Therefore, the seed vertices must include not only those at convex areas but also at saddle points. We select the candidate seed vertices as those whose IRCs are smaller than half of the smallest dimension of the bounding box of the object, and sort them in ascending order of the IRC (cf. line 1 of Algorithm 3). We go through the seed vertices one by one, and skip a vertex if all of its incident faces have been assigned to a patch (a tip, segment, or junction). Checking incident faces (instead of checking only vertices) is necessary, because some triangles may not have been assigned to any patch even after all vertices have been assigned.

4.3.2. Growing Candidate Patch—Similar to the previous step, we grow a candidate patch starting from an unassigned face incident on a seed vertex, by locating the triangles that intersect or fall within its AICC-centered MCS and at the same time are not yet assigned to another patch. Although the candidate patch starts as a topological disk, it would inevitably grow into a non-disk at segments and junctions, for which we must consider the potential interactions of the different cuts of a patch. Therefore, we grow a candidate patch differently depending on its topology, as we will describe shortly. Note that it is possible for a candidate patch to change its topology during the growth.

Given a patch, we define a border curve of the patch as a single closed curve composed of border edges of the patch. For each patch, we continue to grow it until each border curve has a corresponding cut. In general, each cut has two adjacent patches. For each border curve, its corresponding cut is therefore either matched with a cut that was identified in an adjacent patch earlier, or is created as a new cut and to be matched by another patch later. After locating a patch, we assign an integer ID to it, mark all the faces in it by the patch ID, and associate each newly created cut with a seed face that is incident on a border edge of its corresponding border curve. This seed face will be used for matching the cut later.

Given a candidate patch that is topologically a disk, we grow it as in Algorithm 1 by using the AICC-centered MCS of the candidate patch. A minor difference is that when filling small holes, we consider not only those within the candidate patch but also those between it and previously identified patches. Since we have identified the tips that are stable by themselves, in this step

we accept a disk-patch as a tip only if its border curve γ is matched with an existing cut. We match γ with a cut P in an adjacent patch Γ if every border edge of γ is incident on a triangle assigned to Γ , and P is the cut in Γ whose seed face is incident on γ .

When a candidate patch grows into a non-disk, we then change to grow the patch along each of its border curves separately to identify a cut for each of its border curves. For a patch with two border curves, if one of its border curves γ does not match any existing cut, we attempt to construct a cutting plane from γ , measure its orthogonality of the candidate cut in terms of the minimum dihedral angle, and improve its orthogonality based on the normal covariance matrix. This process is the same as that for stable tips in the previous step. In addition, we emphasize that we place the cut so that all the triangles incident on the border curve are above the cutting plane, so that the remainder of the surface patch would not have any isolated triangles and can be further decomposed into valid sub-structures. If the candidate cut is sufficiently orthogonal, we accept it as the cut for the curve. If an orthogonal plane cannot be obtained, we grow the candidate patch outward at γ . Specifically, we construct a sphere centered at the centroid of γ and containing the triangles incident on γ , and then locate the triangles that fall within or intersect the sphere and are not yet assigned to a different patch, accomplished by a breadth-first search from the triangles incident on γ . We add these triangles into the candidate patch and then re-attempt to match a cut or determine a new cut for the patch. We continue to grow the patch until a cut is identified for the border curve or the number of border curves has changed.

For a candidate patch with more than two border curves, it is desirable for the bounding cuts to be as close to the junction as possible while maintaining the orthogonality of the cuts, so that the junction itself would be more likely to be convex and be closer to a ball-shape. For this reason, we relax the tolerance of orthogonality for junctions. For each border curve γ , we attempt to match a cut, determine a new cut, or grow the patch at γ as above. Starting from each of the identified cuts (either matched or newly created), we try to move it closer to the junction using a bisection procedure. In particular, we try to move the cut toward the centroid of the junction by half of its distance and then optimize its orthogonality. If the plane is not orthogonal enough, we move it back closer to the border curve; otherwise we move the plane closer to the centroid. This process is repeated for a few (typically four) iterations. We identify the cuts one by one and accept the patch as a junction if a cut is identified for each border curve.

4.3.3. Decomposing Multi-Cut Junctions—When growing a patch, it is probable for a junction to have more than three border curves. Some of these junctions may be real multi-furcations in the geometry, while other are artifacts. These artifacts are especially likely to occur if the geometry has regions where junctions are close to each other. To resolve this issue, we attempt to decompose any multi-cut junctions into bifurcations. In particular, if a candidate junction with more than three cuts is obtained, we sort its cuts by their maximum radii (computed as the maximum distance from the points on the curve to the centroid) in ascending order, and then try to recompute a patch from the seed face of each of these cuts. If we reach at the same number of cuts starting from different seeds, we accept the multi-cut junction as a real multi-furcation. We accept a recomputed patch if it is a segment or a bifurcation. If a recomputed patch is neither a segment nor a bifurcation but has fewer cuts, we then try to recompute for this smaller patch recursively.

In practice, multi-furcations typically have few cuts. For better efficiency, we set an upper bound on the number of cuts of a junction. We stop the growth of a patch if its number of cuts is larger than the upper bound and resort to recomputing the junction from these cuts as above. In practice, an upper bound 10 works well for a variety of geometries in our test. This upper bound is rarely reached even without the decomposition.

The two steps (steps 2 and 3) described in Section 4.2 and 4.3 identify an initial decomposition of the object into tips, segments, and junctions. This initial decomposition determines the topology of the medial curve. In step 3 we preserved all the stable tips identified in step 2. For better efficiency, it is possible to merge part of step 3 into step 2. In particular, we can identify the segments and junctions along with the stable tips in step 2, but allow stable tips to take over the triangles assigned to segments and junctions and remove the segments and junctions that overlap with tips. Then, in step 3 we need to identify only the remaining patches from faces that are not yet assigned to any patch. However, such a reorganization complicates the logic of the algorithm, so we sacrificed efficiency slightly for clarity in our algorithm.

4.4. Fine-Tuning Cuts

The preceding steps identified the local sub-structures mostly independently of each other, except for the care taken in matching the cuts and ensuring topological consistencies. Those steps could not optimize the cuts with respect to each other, because some adjacent patches might still be unknown. Therefore, the lengths of adjacent segments may differ drastically. In addition, our algorithm might have pushed some cuts too deep into the junction in order to keep the junctions as compact as possible. It is therefore desirable to have an additional step to fine-tune the cuts.

We first adjust the location of the cuts at junctions. The objective is to location the cuts that are as orthogonal to the surface as possible while being not too far from the junction. We use a bisection algorithm to move each cut iteratively toward the adjacent cut to locate the point that optimizes orthogonality, and use a more strict tolerance for orthogonality than in Section 4.3. After optimizing the cuts at junctions, we then perform an iterative Laplacian smoothing to redistribute the cuts between segments and optimize their orthogonality, while preserving the cuts at the tips and junctions.

4.5. Reconstructing Medial Curve

After decomposing an object into tips, segments, and junctions, we reconstruct its medial curves based on the connectivity of these patches. Specifically, we create a vertex in the medial curve for each cut center and for the center of each segment and junction, and assign consecutive integer IDs to the vertices. After creating the vertices, we create an edge in the medial curve to connect each cut center with the center of each of its incident segments or junctions, and identify each edge by a pair of vertex IDs. We choose the cut center to be the centroid (i.e., the center of mass) of the cross section of the cutting plane with the surface. The centroid is defined as $\int_C x dx dy / \int_C dx dy$, which we compute by decomposing the cross section into triangles apexed at a point in the cross section and then computing the signed-area-weighted average of the centroids of these triangles. For each segment, we compute its center as the average of the cut centers. For each junction, we compute its center as the volume centroid of the junction, and compute it as a signed-volume-weighted average of the centroids of the pyramids that are apexed at the same point and based at the cuts or based at the parts of the faces that fall within the patch. The centroid of a pyramid is simply three fourths of the centroid of the base plus one fourth of the apex.

4.6. Implementation Details

Our algorithm requires only the vertex coordinates and element connectivity of the triangles as input. Internally, our algorithm requires some data structures to store the mesh, the cuts, the patches, and their correlations with each other. We hereafter describe these data structures.

For the mesh, we need a data structure that can facilitate neighbor-to-neighbor traversal. We use the well-known *half-edge data structure* (also known as doubly-connected edge list) [7, §2.2], and in particular an array-based version described in [1]. At a high-level, each edge in

general has two opposite, directed half-edges in its two incident triangles. Our data structure identifies each half-edge in a face by a face ID and a local edge ID within the face, encoded in a single integer, and stores the mapping from each half-edge to its opposite half-edge in a 2-D array of size $m \times 3$, where m is the number of triangles. Together with the element connectivity and a mapping from each vertex to one of its incident half-edges, we have a compact and complete half-edge data structure. This data structure can be constructed efficiently in linear time. We group these arrays along with other geometric quantities (including normals, curvatures, and surface areas) into an object (a.k.a. a structure).

A key output of our algorithm is the medial curve. Similar to the input mesh, we store the medial curve simply by a list of vertex coordinates and the element connectivity of the edges. For later processing, a data structure is also needed for neighbor-to-neighbor traversal of the curve. We define a vertex-based data structure for storing the medial curve, analogous to the array-based half-edge data structure. Specifically, we refer to a vertex within each edge as a “half-vertex” and identify it by the edge ID and a local vertex ID within the edge, encoded in a single integer. Since the medial curve in general is not a manifold, a vertex can have more than two incident edges. We construct a cyclic mapping for the half-vertices corresponding to the same vertex (instead of a mappings between opposite half-vertices in a manifold mesh). This mapping is stored in a 2-D $n \times 2$ array, where n is the number of edges in the medial curve. We refer to this as an *extended half-vertex data structure* for non-manifold curves.

In our algorithm, we assign an integer ID for each cut and patch, and store the data for them in arrays grouped with an aggregate object. We assign patch IDs in the order of being identified. Note that each cut is incident on two patches. We consider the patch that is identified first as the *owner patch* of the cut. We assign IDs to the cuts consecutively in ascending order of their owner patches. For each cut, we store its centroid, its normal outward to its owner patch, as well as the IDs of its incident patches. For each patch, we store the cuts owned by it consecutively and also store a list of the IDs of the matched patches. These data allow the complete reconstruction of the medial curve as well as the computation of the statistics of the medial curve for applications. For best efficiency, our algorithm preallocates buffer spaces used by the algorithm and group them into a single object.

We have implemented our algorithm in MATLAB, which is feasible because our data structures are all array-based, without using any pointers. The use of the high-level programming language MATLAB and its interactive visualization environment eases implementing and debugging of the complex logic of our algorithm. We then converted our MATLAB code into efficient and portable ANSI C code using the Agility MCS software (www.agilityds.com; acquired by Mathworks in early 2009).

5. Customizing for Biological Applications

The algorithm we described in the preceding section was intended to be general; it takes into account only the geometry of the object itself. However, applications sometimes have specific needs for the medial curve, so it is desirable to extend or customize our algorithm to leverage the application-specific knowledge. In this section, we describe the use of the medial curve constructed by our algorithm for biomedical applications, with morphometry as an example.

Morphometry is concerned with studying variation and change in the size and shape of biological objects, such as lung airways. Important measures for morphology include the local radius of the object, angles of bifurcations, and distance between junctions. Statistical analysis of these measures can reveal patterns of genetic decision making during growth or can be used to create lower-dimensional models of transport.

Most biological trees are conduits for flow; resistance to flow is related to the local radius to the fourth power, and is thus a critical morphometric measure. However, the “local radius” is a vague concept. A standard practice is to evaluate the shortest distance from a point on the medial curve to the surface [9], but such an approach is sensitive to the position of the medial curve. Instead, we compute the *hydraulic diameter*, $D_h = 4A/U$, where A denotes the area of a cross section, and U is the perimeter of the cross section. The hydraulic diameter is relatively insensitive to the perturbation to the cut, as long as the cut is nearly orthogonal. We compute these quantities at vertices of the medial curve corresponding to the cut centers and then interpolate them to segment centers.

For junctions, the local radius is not as meaningful as it is for segments, largely because it is ambiguous. Rather, the most important quantity for junctions is the *bifurcation angles*, especially for tree structures such as lung airways and blood vessels. Like local radius, the bifurcation angle is also vaguely defined. A standard practice is to define it as the enclosed angle between two daughter segments.

To compute the bifurcation angles, it is necessary to determine the upstream and downstream directions at each point, especially for tree structures such as lung airways and blood vessels. This requires identifying a *root node* of the tree. To achieve this, we first group the edges of the medial curve between two junction centers into a *super-segments*. Then, we compute the average cut areas of the super-segments that are incident on tips. The tip incident on the super-element with the largest hydraulic diameter is labeled as the root. Besides computing the average cut area, we also compute the length of each super-segment, as the sum of the edge lengths.

From the above discussions, it seems obvious that the local orthogonal decomposition and the medial curve computed by our method are well suited for morphometry, although our method was not specifically designed for this application. However, this match is not a coincidence, because our method and morphometry share similar mathematical and physical principles, such as orthogonality, stability and centeredness. For this reason, we believe our method has a broad range of scientific applications other than morphometry, especially those whose analysis is based on similar mathematical principles.

6. Experimental Results

In this section, we report some experimental results with our method for complex, image-derived biomedical geometries and compare our method against some others. The medial-curve algorithms are largely independent of the image processing process. In our setting, we used a fuzzy-connected threshold approach for segmentation [?] and a variant of the well-known marching cubes algorithm for the isosurface extraction [26]. To assess the robustness and noise resistance, we will report results for raw, unsmoothed surface meshes from isosurfacing, smoothed meshes with a volume-conserving smoothing algorithm [15], as well as those after adaptive remeshing using the method in [16].

6.1. Results for Smoothed Geometries

We first present results for three complex biomedical geometries. All these geometries are smoothed, as they typical are in real applications. Table 6.1 shows the statistics of the datasets and summarizes our results. Our first test geometry is a mouse coronary artery network, derived from the CT dataset courtesy of Dr. Ghassan Kassab [21]. Figure 6.1A shows the geometry, superimposed on the medial curve computed by our algorithm; see row “Coronary” in Table 6.1 for the statistics of the mesh and the medical curve. Since this geometry is much simpler than the other two (with fewer than 100 tips and junctions), we could visually inspect the accuracy of the result. Our inspection indicated that the structure of the medial curve was 100%

accurate (i.e., fully agreed with the intuition of a trained biologist). This accuracy is remarkable, as evident from the comparative study that we will show in Section 6.3. Our algorithm took 0.97 seconds for this case, including 0.36 seconds to identify the tips, on a Linux computer with a 3GHz Intel Duo Core processor and 2GB memory.

Our second test geometry is a monkey pulmonary airway tree (see Figure 6.1B and row “Monkey Lung” in Table 6.1), derived from the MRI dataset courtesy of Dr. Kevin Minard. The voxelated image consisted of 256 transverse image slices, each depicting a square field of view 10cm on a side. The resolution in each slice was $391\mu\text{m}$, and each slice was $391\mu\text{m}$ thick. Our algorithm processed the dataset in about 27.89 seconds, and identified 6,214 tips and 7,121 junctions. The medial curve is homotopic to the object, as guaranteed by our algorithm. Visual inspection at sampled locations indicates that the curve was smooth, well centered, and structurally correct almost everywhere.

Our third test geometry is a high-resolution rat pulmonary airway tree (see Figure 6.1C and row “Rat Lung” in Table 6.1), derived from the CT dataset courtesy of Drs. Timothy Cox, Richard Jacobs and James Carson. The voxelated image consisted of 1024 slices, and the resolution of each slice was $35\mu\text{m}$, and each slice was $35\mu\text{m}$ thick. The surface mesh had about 8 million triangles. Our algorithm identified 17,416 tips and 12,525 junctions in about six minutes. As for the monkey lung, visual inspection at sampled locations indicates that the curve was smooth and structurally correct almost everywhere.

6.2. Assessment of Sensitivity to Noise

To further assess the noise resistance of our method, we apply our algorithm to a series of meshes for the mouse coronary artery, including an unsmoothed triangulation of the isosurface from marching cubes, a smoothed triangulation, as well as three meshes that were registered to the gradient-limited feature size as described in [16] with three different levels of refinement and derefinement. Table 6.2 summarizes the numbers of vertices, triangles, identified tips, and identified junctions of these cases.

The unsmoothed mesh is a challenging test, as the noise in this dataset is high-frequency and has a magnitude of up to one voxel. Figure 6.2 shows the tips, junctions and medial curve for this dataset. Our method identified the same number of tips with and without smoothing, although it produced different number of junctions (because some high-degree junctions in the unsmoothed mesh were broken into low-degree junctions). This result shows that our method is relatively insensitive to noise. For the adaptively refined meshes, our method slightly fewer tips and junctions (by about 7%), probably because the additional smoothing involved in the mesh adaptation process eliminated some short branches and their corresponding tips.

6.3. Comparison with Other Methods

We compare our method against the publicly available implementations of two other surface-based approaches [8,3]. The method of Dey and Sun in [8] is based on an analysis of a medial geodesic function on the Voronoi-based medial surface [8], and the method of Au et al. in [3] is based on mesh contraction. Table 6.3 shows the numbers of identified tips and junctions as well as the overall runtimes for our method and those two methods for the smoothed and unsmoothed coronary geometries shown in Figure 6.1A and Figure 6.2. Figure 6.3 shows close-up views of the results from these methods. Note that the result of our method for the smoothed geometry has been verified manually, so we use it as the reference solution. The Dey-and-Sun’s approach in [8] produced nearly twice as many tips and junctions as our method. From Figure 6.3A and 6.3C, it is obvious that there were many short, spurious branches. Besides spurious branches, Dey-and-Sun’s approach also missed several actual branches as can be seen in Figure 6.3C. In addition, the method in [8] produced far more tips and junctions for the

unsmoothed mesh than for the smoothed mesh, and the medial curve tends to be highly jagged, showing its sensitivity to noise both topologically and geometrically. In terms of computational cost, the runtime of Dey-and-Sun's method was more than three orders of magnitude slower than ours for this mesh. This ratio grows substantially as the size of the mesh increases, and their code never terminated for the meshes with one million or more faces in our test.

In contrast to [8], the mesh contraction method of Au et al. in [3] appears to be relatively insensitive to noise. However, as shown in Figure 6.3E, their method tends to miss tips and branches. Furthermore, the medial curve from their method was not well centered, with some vertices outside of the object (see Figure 6.3F). In terms of running time, their method is more than 30 times slower than ours. From these results, we conclude that for biomedical geometries, which are the interests of our applications, our algorithm delivers a better combination of structural accuracy, smoothness, centeredness, robustness, noise resistance, and efficiency, compared to the other state-of-the-art methods.

7. Conclusion and Discussions

In this paper, we presented a new method for computing the medial curves of surface meshes composed of mostly tubular sub-structures. We defined the medial curves based on a decomposition of objects into three different types of substructures, including tips, junctions, and segments, which naturally map to substructures of the medial curve. To facilitate the identification of these substructures, we introduced a new geometric concepts called the interior centers of curvature (ICC), which constitute the medial axis of canal surfaces, and the average interior centers of curvature (AICC), which inherits the key features of the ICC while being more noise resistant. We then proposed robust numerical techniques, including eigenvalue analysis, weighted least squares approximations, and minimization algorithms, to enhance the noise resistance. We have illustrated the effectiveness and robustness of our approach with some highly complex, large-scale, noisy biomedical geometries derived from medical images, and presented a comparison of our method with some existing methods.

Our method is not without limitation. As mentioned earlier, we have primarily focused on objects composed of tubular sub-structures, and omitted objects with intrinsic ambiguities and disk-like substructures. However, our approach, in particular the concept of the interior centers of curvatures, should be useful in identifying the boundary of medial surfaces of disk-like substructures. As a future research direction, we plan to extend our method to aid the construction of medial surfaces and the estimation of local feature sizes of surfaces, which have broader applications such as mesh generation and domain decomposition. As another future research direction, we plan to apply our algorithm to applications including morphometry and multiscale analysis of lung airways, pulmonary arteries and coronary arteries.

Acknowledgments

The first author was supported by the National Science Foundation under award DMS-0809285. The work of the second author was supported by the National Heart and Blood Institute under award 1RO1HL073598-01A1 and by the National Institute of Environmental Health Sciences under award P01 ES011617. The authors would also like to acknowledge Dr. Ghassan Kassab for graciously providing the coronary CT data, Dr. Kevin Minard for the lung MR data, Drs Timothy Cox and Richard Jacob for the CT data of the rat pulmonary airway tree, and Dr. James Carson for the segmentation of the data.

REFERENCES

1. Alumbaugh T, Jiao X. Compact array-based mesh data structures. Proc. 14th Int. Meshing Roundtable 2005:485–504.
2. Amenta, N.; Choi, S.; Kolluri, RK. The power crust; Proc. sixth ACM Symposium on Solid Modeling and Applications; 2001. p. 249-266.

3. Au OK-C, Tai C-L, Chu H-K, Cohen-Or D, Lee T-Y. Skeleton extraction by mesh contraction. *ACM Trans. Graph* 2008;27
4. Blum, H. *Models for the Perception of Speech and Visual Form*. Cambridge, MA: MIT Press; 1967. A transformation for extracting new descriptors of shape; p. 362-380.
5. Cornea N, Silver D, Yuan X, Balasubramanian R. Computing hierarchical curve-skeletons of 3d objects. *Vis. Comput. (Germany)* 2005;21:945-955.
6. Cornea ND, Silver D, Min P. Curve-skeleton properties, applications, and algorithms. *IEEE Trans. Vis. Comput. Graph* 2007;13:530-548. [PubMed: 17356219]
7. de Berg, M.; Cheong, O.; van Kreveld, M.; Overmars, M. *Computational Geometry: Algorithms and Applications*. 3rd ed.. Springer; 2008.
8. Dey T, Sun J. Defining and computing curve-skeletons with medial geodesic function. *Proc. Eurographics Symp. Geometry Proc* 2006:143-152.
9. Einstein DR, Neradilak B, Pollisar N, Minard KR, Wallis C, Fanucchi M, Carson JP, Kuprat AP, Kabilan S, Jacob RE, Corley RA. An automated self-similarity analysis of the pulmonary tree of the sprague-dawley rat. *Anat Rec* 2008;291:1628-1648.
10. Gray, A.; Abbena, E.; Salamon, S. *Modern Differential Geometry of Curves and Surfaces with Mathematica*. 3rd ed.. Chapman & Hall/CRC; 2006.
11. Huo Y, Kassab GS. A hybrid one-dimensional/Womersley model of pulsatile blood flow in the entire coronary arterial tree. *Am J Physiol Heart Circ Physiol* 2007;292:H2623-H2633. [PubMed: 17208998]
12. Huo Y, Kassab GS. A scaling law of vascular volume. *Biophys J* 2009;96:347-353. [PubMed: 19167288]
13. Jiao X, Zha H. Consistent computation of first- and second-order differential quantities for surface meshes. *Prof. ACM Solid and Physical Modeling Symposium* 2008:159-170.
14. Katz S, Leifman G, Tal A. Mesh segmentation using feature point and core extraction. *Vis. Comput* 2005;21:865-875.
15. Kuprat A, Khamayseh A, George D, Larkey L. Volume conserving smoothing for piecewise linear curves, surfaces, and triple lines. *J. Comput. Phys* 2001;172:99-118.
16. Kuprat AP, Einstein DR. An anisotropic scale-invariant unstructured mesh generator suitable for volumetric imaging data. *J. Comput. Phys* 2009;228:619-640. [PubMed: 19784397]
17. Lam L, Lee S-W, Suen CY. Thinning methodologies—a comprehensive survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 1992;14:869-885.
18. Lee T-C, Kashyap RL, Chu C-N. Building skeleton models via 3-d medial surface/axis thinning algorithms. *CVGIP: Graph. Models Image Process* 1994;56:462-478.
19. Maddah M, Afzali-Kusha A, Soltanian-Zadeh H. Efficient center-line extraction for quantification of vessels in confocal microscopy images. *Med Phys* 2003;30:204-211. [PubMed: 12607838]
20. Maddah M, Soltanian-Zadeh H, Afzali-Kusha A. Snake modeling and distance transform approach to vascular centerline extraction and quantification. *Comput. Med. Imaging Graph. (UK)* 2003;27:503-512.
21. Molloy S, Kassab GS, Zhou Y. Quantification of coronary artery lumen volume by digital angiography: In-vivo validation. *Circulation* 2001;104:2351. [PubMed: 11696477]
22. Molthen RC, Karau KL, DCA. Quantitative models of the rat pulmonary arterial tree morphometry applied to hypoxia-induced arterial remodeling. *J Appl Physiol* 2004;97:2372-2384. [PubMed: 15333611]
23. Mortara M, Patane G. Affine-invariant skeleton of 3d shapes. *Shape Modeling International* 2002:245-252.
24. Mortara M, Patane G, Spagnuolo M, Falcidieno B, Rossignac J. Blowing bubbles for the multi-scale analysis and decomposition of triangle meshes. *Algorithmica, Special Issues on Shape Algorithms* 2004;38:227-248.
25. Mortara M, Patane G, Spagnuolo M, Falcidieno B, Rossignac J. Plumber: a method for a multi-scale decomposition of 3d shapes into tubular primitives and bodies. *ACM Symposium on Solid Modeling and Applications*. 2004

26. Newman T, Yi H. A survey of the marching cubes algorithm. *Computers & Graphics* 2006;30:854–879.
27. Palágyi K, Kuba A. A 3d 6-subiteration thinning algorithm for extracting medial lines. *Pattern Recognition Letters* 1998;19:613–627.
28. Pascucci V, Scorzelli G, Bremer P-T, Mascarenhas A. Robust on-line computation of reeb graphs: simplicity and speed. *ACM Trans. Graph* 2007;26:58.
29. Tierny J, Vandeborre J-P, Daoudi M. Enhancing 3d mesh topological skeletons with discrete contour constrictions. *The Visual Computer* 2008;24:155–172.
30. Wang Y-S, Lee T-Y. Curve-skeleton extraction using iterative least squares optimization. *IEEE Trans. Vis. Comput. Graph* 2008;14:926–936. [PubMed: 18467765]
31. Whittaker DR, Dwyer J, Fillinger MF. Prediction of altered endograft path during endovascular abdominal aortic aneurysm repair with the gore excluder. *J Vasc Surg* 2005;41:575–583. [PubMed: 15874919]
32. Wischgoll T, Choy JS, Ritman EL, Kassab GS. Validation of image-based method for extraction of coronary morphometry. *Ann Biomed Eng* 2008;36:356–368. [PubMed: 18228141]

FIG. 3.1. Examples of medial curves (red) of a cylinder, a torus, and a surface generated by a sphere of radius $0.3 + 0.15 \sin(3t)$ moving on a unit circle.

FIG. 3.2. Illustration of tip, segment, and junction, along with orthogonal cuts (blue planes). Red curves show medial curves.

FIG. 6.1. Medial curve of three biological surfaces. **A)** A mouse coronary artery from CT data. **B)** A monkey pulmonary airway from MRI data. **C)** A high-resolution rat pulmonary airway from CT data.

FIG. 6.2.
Noisy dataset of mouse coronary artery and the computed medial curve.

FIG. 6.3.

Comparison of our results against two other methods for the smoothed and unsmoothed coronary geometry. Artifacts in the results are indicated by black arrows. Panels A–D contrast the medial-geodesic method with our method; Panels E–H contrast the mesh contraction method with our method.

TABLE 6.1

Results for three test geometries in Figure 6.1.

Case	Input Mesh		Elements		Run Times (sec)		
	vertices	triangles	tips	junctions	tips	others	total
Coronary	57,008	114,012	96	85	0.36	0.60	0.97
Monkey Lung	895,915	1,791,826	6,214	7,121	13.3	14.6	27.9
Rat Lung	3,975,949	7,951,894	17,416	12,525	55.8	312.7	368.5

TABLE 6.2

Results for different meshes for the mouse coronary.

Case	Input Mesh		Elements	
	vertices	triangles	tips	junctions
Unsmoothed	57,008	114,012	96	83
Smoothed	57,008	114,012	96	85
Adaptive 1	76,143	152,282	92	81
Adaptive 2	71,130	142,256	91	83
Adaptive 3	64,685	129,366	91	79

TABLE 6.3

Comparison of our method with alternative surface-based methods for the mouse coronary artery. For Au et al., numbers of tips and junctions were counted visually, as their code did not allow saving the result. All times are in seconds

Case	Ours			Dey and Sun [8]			Au et al. [3]		
	tips	junctions	time	tips	junctions	time	tips	junctions	time
Smooth	96	85	0.97	146	142	1294.3	84	79	35.3
Unsmoothed	96	83	1.10	163	161	1439.3	87	81	32.6