Chapter 7
Vascular Geometry Reconstruction and Grid Generation

Thomas Wischgoll, Daniel R. Einstein, Andrew P. Kuprat, Xiangmin Jiao, and Ghassan S. Kassab

Abstract The geometry of vascular system is an important determinant of blood flow in health and disease. There is a strong geometric component to atherosclerosis in coronary heart disease since lesions are preferentially located at bifurcation points and regions of high curvature. The influence of these local structures on recirculation and deleterious shear stresses and their role in plaque development is widely accepted. Over time, researchers have turned to MR, CT, or biplane images of vascular trees to faithfully capture these features in the flow simulations. Historically, this has taken the form of labor-intensive manual reconstructions from morphometric measurements based on the centerline, whereby small idealized subsets of vascular trees are developed into computational grids. With improved imaging, image processing, and geometric reconstruction algorithms, researchers have begun to develop geometrically accurate computational models directly from the medical images. This chapter provides an overview of contemporary methods for image processing, centerline detection, boundary condition definition, and grid generation of both clinical and research images of cardiovascular structures.

7.1 Introduction

Computational fluid dynamics (CFD) has become an increasingly important component of integration and discovery in cardiovascular research. Although fluid and tissue stresses are not easily measured, they can be predicted through physics-based simulations. This is critical for cardiovascular research because vessel wall shear stress profiles can endothelial function, thrombus formation, and rupture, as well as the growth of aneurysms and atherosclerotic plaque. These and other cardiovascular

T. Wischgoll (✉)
Department of Computer Science and Engineering, Wright State University, Dayton, OH, USA
e-mail: thomas.wischgoll@wright.edu

J.M. Guccione, et al. (eds.), *Computational Cardiovascular Mechanics*,
DOI 10.1007/978-1-4419-0730-1_7,

issues are generally coupled multiphysics problems with a strong geometric component. These geometries are increasingly derived from imaging modalities such as magnetic resonance imaging (MRI) or computed tomography (CT), and are complex and articulated across multiple scales.

Efficient visualization, analysis, and unstructured mesh generation of these multiple geometries in a way that is tuned to the physics of the problem remains challenging. Finite computational resources dictate that computational geometry algorithms must be efficient and that the grids must be optimally adapted to the geometry in order to minimize both computational cost and discretization error. At the same time, cardiovascular biophysical simulations require that the grid be organized both by scale and by intrinsic properties. The arterial wall, for example, is a laminated tissue consisting of three separate layers, each with a separate family of collagen and elastin fibers and smooth muscles. Thus, a grid of the vessel wall must be similarly layered. That same layering persists at all scales of a vascular network. These transitions of scale are mirrored in the blood. The lumen of a coronary arteriole, for example, may be several orders of magnitude smaller than the thickness of a ventricle, and thus there is a need to manage error over a range of meaningful scales. Although the issue of scale clearly exists in both the fluid and the solid domains of physiological problems, an optimal discretization of the fluid and solid will be quite different due to the very different physics that dominate each domain. The most notable difference is that fluid problems tend to have strong gradients at the boundary.

Here, we survey some recent developments for computational grids for vascular CFD or fluid–solid interaction simulations. Specifically, we focus on image processing, centerline detection, and grid generation. Image processing, and particularly image segmentation, is a necessary first step for both centerline detection and grid generation. Given a centerline, some researchers have defined idealized grids based on subsets of arterial trees, wherein each segment of the centerline is associated with a diameter and length, and assembled into a network of tapered tubes. Although these types of grids have yielded valuable insights into cardiovascular flow, our focus is to develop grids directly from the medical image. Nevertheless, the centerline remains an important data structure for morphometric analysis and thus has an important role in the determination of physiological multiscale boundary conditions. Specifically, the centerlines allow for the computation of various quantitative measurements, such as vessel length, vessel radius, and bifurcation angles.

7.2 Image Processing

In order to identify the geometry of the vasculature, typically MRI or CT is used resulting in a volumetric image. A volumetric image consists of voxels aligned along a regular 3-D grid. It is generally not likely that the boundary of the vessels is exactly located at these voxels. A better precision can be achieved by finding the exact location in between a set of voxels. Since an accurate representation of the object boundary is crucial to any further processing of the data, improvement of the precision is an essential step. Different approaches are available depending on the need of the algorithm used to further process the result. Some algorithms for computing the centerline only require an accurate representation of individual points. On the other hand, grid generating algorithms typically require a surface representation of the boundary; i.e., the points need to be connected by some geometric primitive. The following subsections provide examples of both types of algorithms.

7.2.1 Segmentation of the Vessel Boundary

The method described here uses similar techniques as described by Canny's non-maxima suppression [1] but extended to three dimensions. First, the image gradient is computed for every voxel. Using an experimentally determined threshold, all voxels with a gradient length below this threshold are neglected. The advantage of this gradient-based thresholding is that it is less sensitive to the selected threshold compared to intensity-based segmentation algorithms. This is particularly important for smaller vessels (1 voxel in diameter or less) that can be missed due to partial volume effects when using intensity segmentation.

In order to achieve sub-voxel precision, the gradients of the voxels exceeding the threshold are compared to their neighbors to identify local maxima along the gradient. In 3-D, the direct neighborhood of a single voxel generally consists of 26 voxels forming a cube that surrounds the current voxel. In order to find the local maximum along the current gradient, the gradients of the neighboring voxels in positive and negative directions have to be determined. When using 2-D images, nearest-neighbor interpolation of these gradients [2] may work but yield incorrect results in a 3-D volumetric image. Therefore, the gradients on the boundary of the cube formed by the neighboring voxels are interpolated linearly to determine a better approximation of the desired gradients.

Once the neighboring gradients in positive and negative direction of the current gradient are computed, they are compared to find the local maxima. Thus, if the length of the current gradient is larger than the length of both of its neighbors, the local maximum can be calculated similar to the 2-D case. When interpolated quadratically, the three gradients together form a parabolic curve along the direction of the current gradient. In general, the current gradient is larger than the interpolated neighbors since only local maxima are considered in this step. Hence, the local maximum can be identified by determining the zero of the first derivative of the parabolic curve. The determination of all local maxima within the volumetric image in this fashion then results in a more accurate and smoother approximation of the object boundary with sub-voxel precision. Once all points on the boundary are extracted from the volumetric image using this gradient approach with sub-voxel precision, the resulting point cloud can be further processed to identify the centerlines.

7.2.2 Segmentation Under Topological Control

In order to create a volume grid that is faithful to the medical image, it is necessary to produce a triangulated isosurface from a segmentation of the data. An important consideration is to produce such an isosurface while preserving correct vessel topology. From a topological point of view, an arterial tree (excluding the capillary bed) is homeomorphic with a sphere. Due to finite resolution, isosurfacing algorithms such as Marching Cubes [3] are unable to determine whether voxels that connect only by a corner or by an edge should truly be connected. This ambiguity can give rise to multiple handles that corrupt segmentations. Therefore segmentations must be performed under topological control [4].

To segment the data, a fuzzy connected-threshold algorithm is applied to the image in order to convert the series of grayscale images into a binary volume. Connectedness is restricted to face connectivity to prevent ambiguous representations of the surface between vessels and background. Face connectivity is accomplished both by restricting the region growing algorithm to faces and by a post-segmentation connectivity check that reassigns voxels found to possess vertex or edge connectivity.

Subsequently, loops are removed to bring the face-connected segmentation into proper topology using an automated approach based on skeletonization, loop detection, loop cutting, and clean-up. A breadth-first search of branches in the skeleton is applied, starting at the top of the coronary ostia. To find the optimal cutting location within the loop, a test cut is performed separately for each skeleton voxel belonging to the loop. Cuts are then affected at the region of minimum cross-sectional area and maximum path length from the ostia.

To extract the isosurface from the segmented image, we apply the Marching Tetrahedra variant of the popular Marching Cubes algorithm (Fig. 7.1). This produces a closed triangulated surface, devoid of boundary patches at the inlets and outlets and whose surface density is a function of resolution of the underlying data.

7.3 Centerline Detection

Numerous algorithms for extracting centerlines from volumetric data sets are available. An overview of the various techniques can be found in the paper by Cornea et al. [5]. Some methods begin with all voxels of a volumetric image and use a thinning technique to shrink down the object to a single line [6, 7, 8, 9, 10, 11–13, 14]. Ideally, the topology of the object should be preserved as proposed by Lobregt et al. [15] which is the basic technique used in commercial software systems, such as Analyze^{TM}. Luboz et al. [3] used a thinning-based technique to determine vessel radii and lengths from a CT scan. A smoothing filter was employed to eliminate the jaggedness of the thinning process and the results were validated using a silicon phantom. A standard deviation of 0.4 mm between the computed and the actual

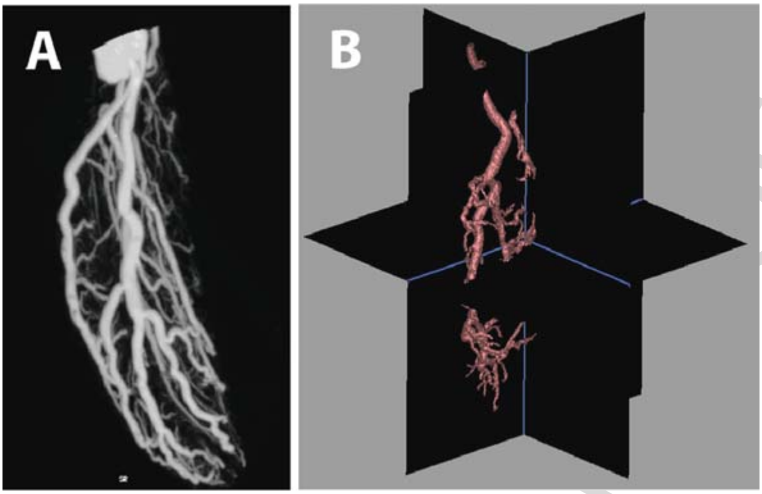


Fig. 7.1 Maximum intensity projection of mouse coronary vasculature (**a**). Segmentation and isosurface extraction (**b**)

measurements was reported for a scan with a resolution of 0.6 mm. The disadvantage of thinning algorithms is that they can only be applied to volumetric data sets and the centerlines are described at voxel-precision resulting in somewhat jagged lines, which do not allow accurate measurements of branch angles.

Other approaches use the distance transform or distance field in order to obtain centerlines. For example, fast marching methods [16, 17] can be employed to compute the distance field. Voxels representing the centerlines of the object are identified by finding ridges in the distance field. The resulting candidates must then be pruned first. The resulting values are connected using a path connection or minimum span tree algorithm [18, 19, 20]. The distance field can also be combined with a distance-from-source field to compute a skeleton [21]. Similar to thinning approaches, these methods are voxel-based and tend to generate the same jagged centerlines. This implies that a centerline can deviate from its original location by up to half a voxel due to the numerical representation.

A more recent method by Cornea et al. [31] computes the distance field based on a potential similar to an electrical charge and then uses a 3-D topological analysis to determine the centerlines. Typically, this approach is very accurate. The computations of the centerlines for a CT-scanned volumetric image of a typical size, such as $512\times512\times200$, would take several months, however, which renders it impractical.

Techniques based on Voronoi diagrams [22, 23] define a medial axis using the Voronoi points. Since this approach usually does not result in a single line but rather a surface-shaped object, the points need to be clustered and connected in order to obtain centerlines. Voronoi-based methods can be applied to volumetric images as well as point sets. These methods usually tend to extract medial surfaces rather than single centerlines. Hence, clustering of the resulting points is required which may introduce numerical errors.

For extracting centerlines from volumetric images, geometry-based approaches are preferable over voxel-based approaches. Due to the discrete nature of a voxel of the volumetric image, the location of the centerline can have an error of half a voxel. Geometry-based methods do not have this shortcoming. Nordsletten et al. [24] determined normal vectors based on an iso-surface computed using the volumetric image. These normal vectors are projected inward. The resulting point cloud is then collected and connected by a snake algorithm.

The method described in the following subsections follows an algorithm developed by Wischgoll et al. [20]. The major advantage of this approach lies in the demonstrated accuracy based on actual validations between computed vessel diameters and optical measurements for porcine hearts. This algorithm consists of several steps. Since the object is given as a volumetric CT-scanned image, the object boundary is extracted as previously described. A vector field is then computed that is orthogonal to the object boundary surface. Once the vector field is computed, the centerlines can be determined by applying a topological analysis to this vector field. As a last step, gaps between segments of the centerlines can be closed automatically and vessel diameters can be computed. The following subsections explain these steps in detail.

7.3.1 Vector Field

The proposed method computes the centerlines by applying a topological analysis to a vector field that is determined based on the geometric configuration of the object of which the centerlines are to be determined. The vector field is computed at the identified points on the vessel boundary in such a way that the vectors are orthogonal to the vessel boundary surface. Based on these vectors, the vector field inside the vessels is computed using linear interpolation.

Since the vasculature is given as a volumetric data set, the image gradients can be used to define these vectors on the boundary surface. These image gradients are previously determined as they are needed for extracting the boundary. Since the points are only moved along the direction of the image gradient when determining the sub-voxel precision, this image gradient is still orthogonal to the boundary surface and therefore represents a good approximation for the desired vector field.

7.3.2 Determination of the Centerlines

In order to determine the centerlines of the object, a tetrahedrization of all points on the object boundary is computed first. For this, Si's [18] fast implementation of a Delaunay tetrahedrization algorithm is used. Tetrahedra outside of the vessels are removed based on the gradient vectors. Note that this step also closes small gaps that may exist since tetrahedra covering these gaps will still have vectors attached to the vertices which point inward. Since vectors are known for each vertex

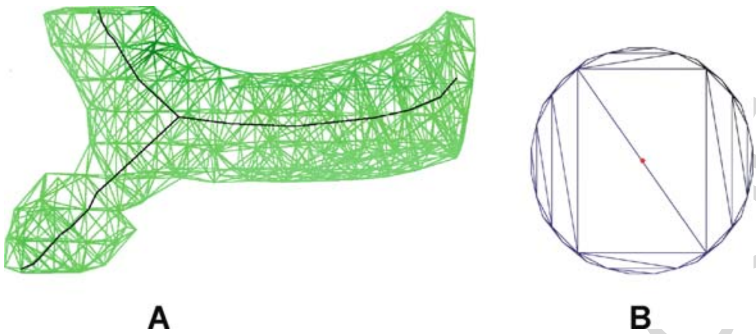


Fig. 7.2 A bifurcation for a small vessel (3 voxels in diameter). The extracted centerline is shown along with the respective tetrahedrization (**a**); Single slice through the tetrahedrization of the phantom data set. The point on the centerline is identified in the center of the image (**b**)

of every tetrahedron, the complete vector field can be computed using this tetrahedrization by linear interpolation within each tetrahedron. This vector field is then used to identify points of the centerlines which are then connected with each other. Figure 7.2a shows an example of the tetrahedrization with outside tetrahedra removed as previously described for a small vessel with a diameter of about 3 voxels. Based on this tetrahedrization and associated vector field, the centerlines can be identified.

In order to perform a topological analysis on the faces of the tetrahedra, the vector field has to be projected onto those faces first. Since tri-linear interpolation is used within the tetrahedra, it is sufficient to project the vectors at the vertices onto each face and then interpolate linearly within the face using these newly computed vectors. Based on the resulting vector field, a topological analysis can be performed on each face of every tetrahedron.

Points on the centerlines can be identified by computing the singularities within the vector field interpolated within every face of the tetrahedrization. For example, for a perfectly cylindrical object, the vector boundary points directly at the center of the cylinder. When examining the resulting vector field at a cross-section of the cylinder, a focus singularity is located at the center of the cylinder within this cross-section. The location of this focus singularity resembles a point on the centerline of the cylinder. Hence, a singularity of type node, focus, or spiral within a face of a tetrahedron indicates a point of the centerline. Since not all objects are cylindrical in shape and given the numerical errors and tolerances, points on the centerlines can be identified from singularities that resemble focus and spiral singularities. Figure 7.2b illustrates an example for a cylindrical object for which a cross-section (a slice perpendicular to the object) is shown. There are two large triangles that connect two opposite sides of the object. Based on these triangles, which resemble faces of tetrahedra of the tetrahedrization, the center point (shown in red) can be identified based on the topological analysis within these triangles.

Obviously, only faces that are close to being a cross-section of the object should be considered to identify points on the centerlines. To determine such cross-sectional faces, the vectors at the vertices can be used. If the vectors at the vertices, which are orthogonal to the object boundary, are approximately coplanar with the face, then this face describes a cross-section of the object. As a test, the scalar product between the normal vector of the face and the vector at all three vertices can be used. If the result is smaller than a user-defined threshold, this face is used to determine points on the centerlines. If we compute the singularity on one of these faces, then we obtain a point which is part of the centerlines. Note that since linear interpolation is used within the face, only a single singularity can be present in each face. In case of bifurcations, there will be two neighboring tetrahedra which contain a singularity, one for each branch. Additionally, this approach disregards boundary points from noise voxels. In order for a set of boundary points to be considered, they need to have gradient vectors that point towards the center from at least three different directions. Hence, boundary points based on noise voxels are automatically neglected.

After computing the center points, vessel diameters are computed for each center point and all points within the vicinity are identified. From this set of points, only the ones that are within the slice of the vessel used to determine the center point are selected to describe the boundary. The radius is then computed as the average of the distances between the center points and the points on the boundary of the vessel slice.

Once individual points of the centerlines (including the corresponding vessel diameters) are computed by identifying the focus and spiral singularities within the faces of the tetrahedra, this set of points must be connected in order to retrieve all centerlines. Since the tetrahedrization describes the topology of the object, the connectivity information of the tetrahedra can be used. Thus, identified points of the centerlines of neighboring tetrahedra are connected with each other forming the centerlines. In some cases, gaps will remain due to the choice of thresholds which can be closed using the method described in the next section.

7.3.3 Geometric Reconstruction

Based on the centerlines extracted from the volumetric image, various measurements can be extracted, such as vessel radius or bifurcation angles. A comparison of the computed radii, which were measured as the distance between centerline and vessel wall, and optical measurements of the radii for the main trunk of five porcine hearts show an excellent accuracy with an average error of 0.7% and rms error of 1.1% of the radii. Using the centerline and radii information, conic cylinders can be formed to represent the individual vessel segment. By representing every segment in this way, the vascular tree can be reconstructed. Figure 7.3 shows an example of such a geometric reconstruction of a porcine heart.

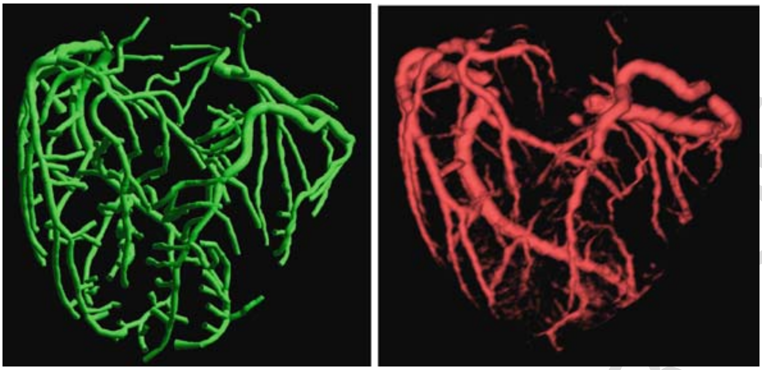


Fig. 7.3 Geometric reconstruction of the vascular tree (*left*) down to the scan resolution based on the centerline and radii information extracted from a CT-scanned porcine heart (*right*). (Wischgoll et al. [20] by permission)

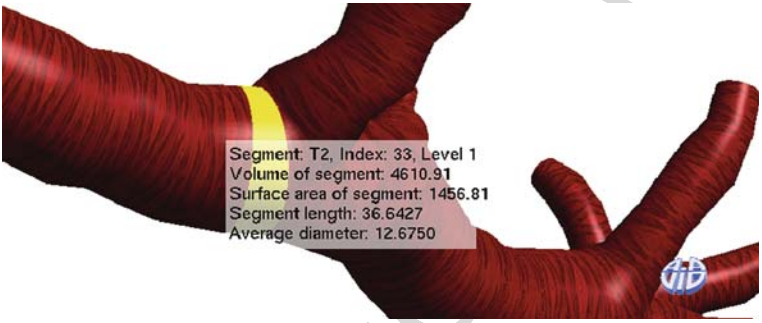


Fig. 7.4 Interactive visualization of vasculature based on the geometric reconstruction showing quantitative measurements

Since the vasculature is represented as geometry, the visualization software not only facilities the gathering of statistical information about the morphometry but it also allows a user to perform various measurements, such as distances or bifurcation angles. By interactively selecting individual vessel segments, for example, the rendering of the geometric reconstruction is overlaid with quantitative measurements, including segment volume and surface area as depicted in Fig. 7.4.

7.4 Grid Generation

With surfaces derived from imaging data, the organization and density of the original surface triangles depend on the resolution of the digital data. The characteristic dimension of the surface triangles is on the order of 1 voxel. Simply generating a volume grid from the original surface could result in grossly under-resolving the computed field where the surface density is close to that of the local feature size or conversely over-resolving the computed field where the surface density is much finer than that of the local feature size. These issues lead to a consideration of the local feature size as an important criterion for sizing and gradation control of the surface that is complementary to criteria that attempt to preserve surface features, topology, and curvature. Moreover, the local feature size in vessel geometry is related to the local diameter. Thus, a measure of the local feature size can also provide a guide for organizing elements radially in layers. This approach has the advantage of creating elements that are mostly parallel to the wall, which reduces discretization error in flows that are predominately axial. At the same time, it essentially decouples strategies for controlling grid density in the normal and tangential directions. It also directly embeds a local understanding of scale into the grid, since the local diameter is related to the local scale.

A robust and computationally efficient metric for local scale is the so-called gradient-limited feature size (GLFS) [25]. Unlike other measures of the local feature size, the GLFS (see Fig. 7.5) can be defined directly on a triangulated surface mesh without a background grid and without referencing the medial axis. Thus, determination of the GLFS is not only computationally efficient, but also robust in the sense that it is Lipchitz continuous and does not change unreasonably under perturbation of the surface mesh. Grids that are organized according to GLFS, such that roughly the same number of layers of elements can be found at all resolved scales, are said to be *scale-invariant*. Scale-invariance is critical in grids of vascular trees because it assures that discretization error at the smallest scale does not unduly affect solution error at the highest scale. In other words, the discretization error is equilibrated at all resolved scales. Combined with the GLFS, the idea of scale-invariance enables the automatic generation of quality anisotropic unstructured grids, while keeping the overall computational cost of the problem tractable. This approach has been adopted in two complementary scale-invariant gridding algorithms for quality layered tetrahedra [25] and quality hybrid prismatic-tetrahedral grids [8]. These algorithms have been implemented in two software frameworks, Lagrit-PNNL and MeshMagic. The defined GLFS in these two algorithms serves three functions: (1) as a field for tangential adaptation of the surface grid, (2) as a metric for creating layered tetrahedra, and (3) as a speed function for construction of a prismatic boundary layer by application of the Generalized Huygens' Principle [26]. Below we define the GLFS and outline these algorithms with examples.

7.4.1 Definition of GLFS

Let S be an oriented closed triangulated surface, which is derived from the isosurface of imaging data by the Marching Cubes algorithm. For a vascular tree, S will consists of a single connected component with genus 0. However, this is not an intrinsic limitation of the approach. As illustrated in Fig. 7.6, we modify S to produce a high-quality surface mesh S' by performing the operations of smoothing, refinement, and de-refinement while limiting perturbations to a small fraction of a voxel.

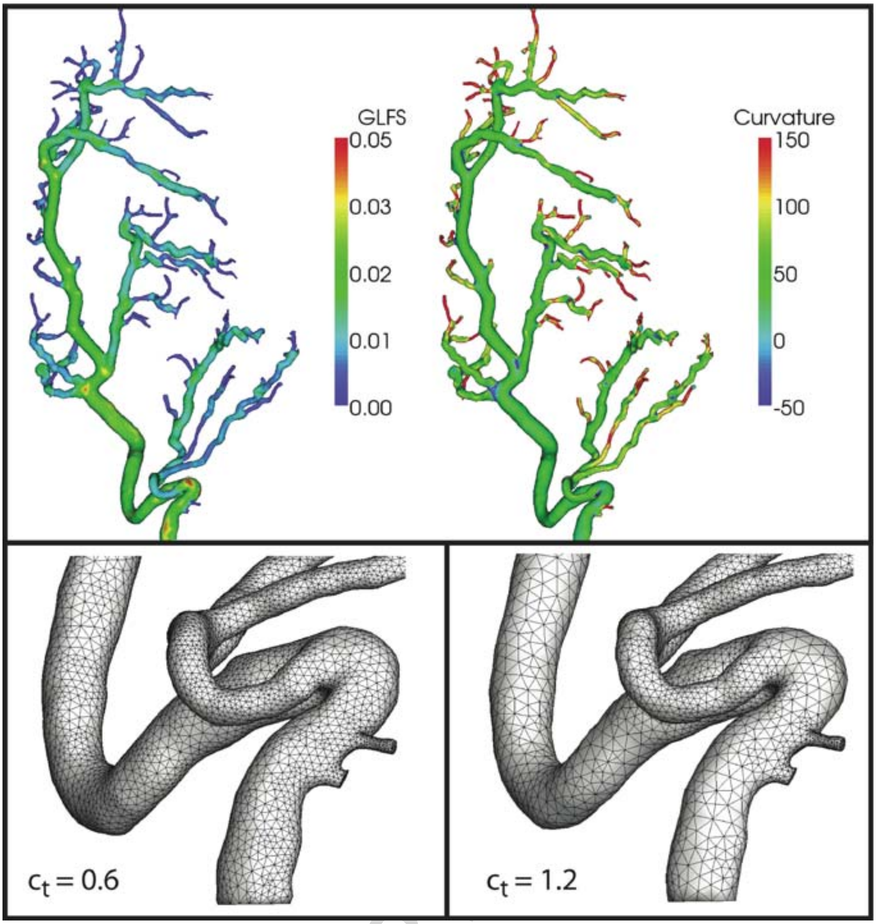


Fig. 7.5 GLFS and first principal curvature (*top panel*) defined on a mouse coronary arterial tree from computed tomography. Efficient computation of these sizing fields was performed in less than 5 s for this geometry on a laptop. Based on the GLFS modulated by the curvature, the original surface mesh from Marching Cubes is selectively refined and de-refined. The bottom panel shows the tangential adaption of the triangulated surface mesh for c_t values of 0.6 (152282 triangles) and 1.2 (129366 triangles). The curvature field for linear values of c_t prevents further de-refinement of the surface grid. For certain applications, it may make sense to convolve the GLFS with a non-linear function that weights higher or lower scales. These operations are supported in Lagrit-PNNL and MeshMagic

For any point $\boldsymbol{x}$ of S, we define *the raw feature size* or *local diameter* $F[\boldsymbol{x}]$ as the length of the line segment formed by first shooting a ray from $\boldsymbol{x}$ in the direction of $\hat{\mathbf{n}}\,[\mathbf{x}]$, the inward normal at $\boldsymbol{x}$, and then truncating the ray at its first intersection with S; that is

$$F[\mathbf{x}] \equiv \min\left\{\lambda > 0 \left| x + \lambda\hat{\mathbf{n}}[\mathbf{x}]\right| \in S\right\}. \tag{7.1}$$

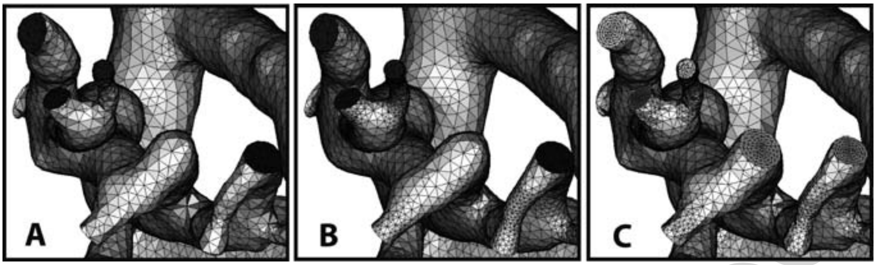


Fig. 7.6 Elaboration of closed surface mesh. Truncation produces valid triangulations and optimally orthogonal planes (**a**). Those triangulations may then be adapted to the physics of the problem (**b**) in order to produce a quality layered tetrahedral [25] grid (**c**)

Since S is closed, with a robust normal $\hat{\mathbf{n}}$ [1] the ray proceeding from $\mathbf{x}$ in the direction $\hat{\mathbf{n}}$ will intersect S at least once, and hence $F[\boldsymbol{x}]$ is well-defined. Similarly, we also perform an outward interrogation of the geometry to compute another raw feature size field F_{out} using $\hat{\mathbf{n}}_{\text{out}} = -\hat{\mathbf{n}}_{\text{in}}$. This outwards value is finite in some areas (e.g., at concave parts of S) and is applied only to the adaptation of surface meshes, where it is necessary to respect a minimum sampling frequency for Delaunay methods.

The raw feature size computed by ray tracing is bounded, but it is sensitive to abrupt changes in the geometry. To address this, we first impose user-specified lower and upper bound to the feature size, denoted by $L_{\min}$ and $L_{\max}$, respectively. Thereafter, we compute a new feature size $f[\boldsymbol{x}]$ by modifying $F[\boldsymbol{x}]$, so that the spatial gradient is relatively insensitive to these changes in S. We accomplish this by performing a gradient-limiting procedure [25]. First, we initialize $f[\boldsymbol{x}]$ to $F[\boldsymbol{x}]$. Given a bound G on the surface gradient of $f[\boldsymbol{x}]$, the algorithm places the directed edges that violate the gradient limit into a max-priority queue, ranked by the key

$$f[x_1] - (f[x_2] + G|x_1 - x_2|)\,. \tag{7.2}$$

which measures how much the gradient violates the gradient limit for a directed edge x_1x_2 on S. Let x_ix_j be the directed edge with the highest priority in the queue. We relax $f[\boldsymbol{x}_i]$ to satisfy the gradient limit, recompute the gradient violation for the edges incident on $\boldsymbol{x}_\text{i}$, and update the priority queue accordingly. The process continues until the queue is empty. For computational efficiency, ray-triangle intersections are queried within an axis-aligned bounding box (AABB) tree [27] that contains at its leaf nodes the bounding box for each triangle. This algorithm has a complexity of $O(N \log N)$, where N is the number of triangles in S. Figure 7.5a shows $f[\boldsymbol{x}]$ for a coronary arterial tree from micro-CT.

7.4.2 Layered Anisotropic Tetrahedra

Once the surface mesh has adapted to some function of the GLFS with edge lengths on the surface equal to about $c_t f[x_i]$, where c_t is a user definable parameter, it is

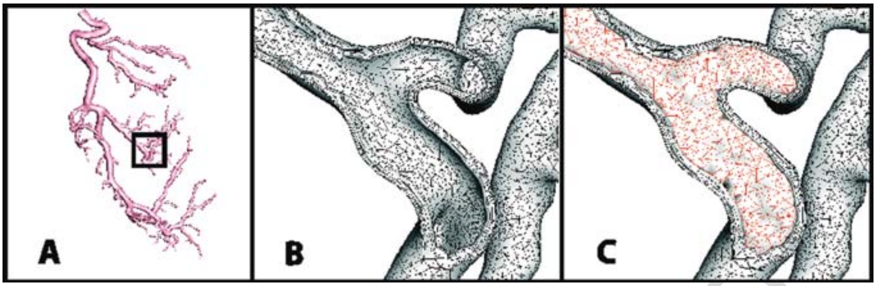


Fig. 7.7 Hybrid prismatic/tetrahedral grid better resolve strong gradients, shear stresses, and particle dynamics at the wall while reducing the overall element count. *Panel A* shows the orientation of the cut-away plane. *Panel B* shows the layer of prisms at the wall. *Panel C* shows the tetrahedra and prisms together

possible to construct either a layered tetrahedral volume grid (Fig. 7.6c) or a layered hybrid prismatic/tetrahedral grid (Fig. 7.7c), depending on the solver.

To create a layered tetrahedral grid, points are cast along "seeding rays" from each point $\mathbf{x}_i$ on the surface S' in the direction $\hat{\mathbf{n}}[\mathbf{x}_i]$. If M is the target number of layers across the cross-section of the geometry, then points $\mathbf{x}_i^m, 0 \le m \le \frac{M}{2}$, are distributed – equally or according to some desired ratio spacing – between $\mathbf{x}_i^0 \equiv \mathbf{x}_i$ on the surface and $\mathbf{x}_i + \frac{1}{2}f[\mathbf{x}_i]\hat{\mathbf{n}}[\mathbf{x}_i]$. Due to gradient-limiting, $f[\mathbf{x}_i] \le F[\mathbf{x}_i]$. In areas where there is greater inequality, extra 'filler' points $\mathbf{x}_i^{M/2+1},\ldots,\mathbf{x}_i^{m_i}$ are inserted between $\mathbf{x}_i + \frac{1}{2}f[\mathbf{x}_i]\hat{\mathbf{n}}[\mathbf{x}_i]$ and $\mathbf{x}_i + \frac{1}{2}F[\mathbf{x}_i]\hat{\mathbf{n}}[\mathbf{x}_i]$. The presence of these filler points guarantees that points are distributed over the whole geometry [25], but with possible overlap, and possibly undesirable proximity to portions of the surface that are nearly grazed by the seeding rays. Consequently, a filtering operation eliminates duplicate points that lie within a fraction of $L_{\min}$ of each other. Finally a Delaunay algorithm connects these points with the restriction that the filler points are not inserted if Delaunay point insertion would connect them to any point $\mathbf{x}_i$ on S'. Tetrahedra that contain no interior points (points $\mathbf{x}_i^m, m \ge 1$) are removed. Finally, the tetrahedral grid is improved with layer-aware, edge-flipping operations and a "crushing algorithm" that inserts nodes on the opposed diagonals of slivers and then merges them, eliminating the slivers. We note that the surface edge lengths $c_t f[\mathbf{x}_i]$ are independent of layer thicknesses.

7.4.3 Hybrid Prismatic/Tetrahedral Grids

In the case of hybrid prismatic/tetrahedral grids, we similarly begin with an adaption of the surface to $c_t f[\mathbf{x}_i]$. Instead of casting and reconnecting points, however, our method advances a surface layer by solving the Lagrangian evolution equation,

$$\frac{\partial x}{\partial t} = f(\mathbf{x},t)\ \hat{\mathbf{n}}, \tag{7.3}$$

where t denotes time, $\hat{\mathbf{n}}$ denotes the unit surface normal, and $f(\boldsymbol{x}\ ,t)$ denotes the GLFS, as defined above.

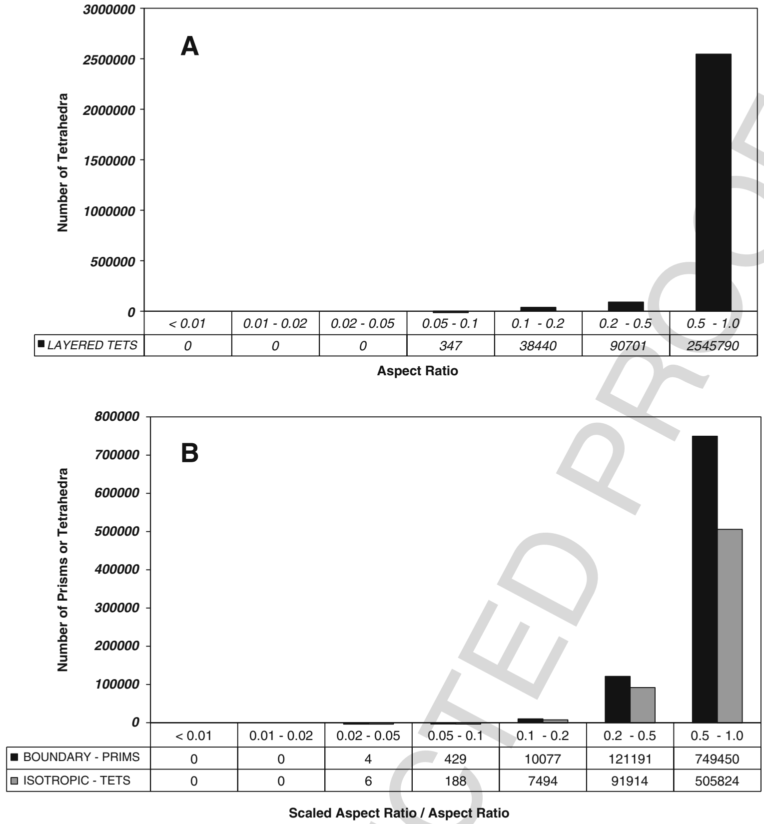
Generating a layer of prisms reduces to marching the vertices in time by discretizing Eq. (7.3). To avoid "swallowtails" [28] in strongly concave regions and in regions with large curvatures, we apply the *face offsetting method* in [29], which is based on a geometric construction called the *generalized Huygens' principle* and numerical techniques of least-squares approximation and eigenvalue analysis. A comprehensive exposition of the approach is given in [8]. Here we simply note that unlike previous approaches, which propagate vertices along some vertex normals, this algorithm propagates faces and reconstructs the vertices. Mesh quality is achieved by applying a novel prismatic variational smoothing procedure to improve base triangle shapes and edge orthogonality. Following face-offsetting, we tetrahedralize the interior with a boundary constrained Delaunay method [30].

7.4.4 Element Quality

Discretization error can have two sources: (1) insufficient grid density to resolve computed gradients, and (2) "badly" shaped elements. What exactly constitutes a badly shaped element is somewhat application dependent. It is generally accepted that an isotropic element, i.e. an element with nearly equal internal angles and approximately equal edge lengths, is "good" and a highly skewed element is "bad". However, for certain classes of problems such as CFD, isotropic elements may be neither necessary nor particularly appropriate. Nevertheless, the accuracy or speed of some applications can be compromised by just a few bad elements, so it is important to be able to judge element quality by some standard measure. In Fig. 7.8, we present the quality statistics of the layered tetrahedral grid shown in Fig. 7.6, and the hybrid prism/tet grid shown in Fig. 7.7. For tetrahedra, we report the aspect ratio which is proportional to the ratio of the inscribed radius to the length of the longest edge. For prisms, we report instead the so-called scaled aspect ratio [8], whose definition is somewhat more involved. In effect, the scaled aspect ratio combines the measures of triangle shapes and edge orthogonality. Both quality metrics vary between 0 and 1, where 1 is optimal.

7.5 Summary

There is no doubt that patient-specific treatment requires the tools to quantify standard patient images using conventional imaging (CT, MRI, etc.). This chapter presents validated image segmentation in conjunction with mesh generation algorithms to create mathematical models of patient vasculature. These mathematical models can then be coupled with physics-based simulations to provide the desired diagnostic or prognostic indices. This approach will clearly impact patient



	< 0.01	0.01 - 0.02	0.02 - 0.05	0.05 - 0.1	0.1 - 0.2	0.2 - 0.5	0.5 - 1.0
LAYERED TETS	0	0	0	347	38440	90701	2545790

	< 0.01	0.01 - 0.02	0.02 - 0.05	0.05 - 0.1	0.1 - 0.2	0.2 - 0.5	0.5 - 1.0
BOUNDARY - PRIMS	0	0	4	429	10077	121191	749450
ISOTROPIC - TETS	0	0	6	188	7494	91914	505824

Fig. 7.8 Element quality statistics for a layered tetrahedral grid of the mouse coronary geometry (**a**), and for the hybrid prism/tetrahedral grid (**b**), shown in black and grey bars, respectively. Both grids were produced with $c_t = 0.6$. For grid (**a**) the number of layers M was set to 8

management medically and surgically, particularly for heart failure where interventions affect the vasculature of the heart.

Acknowledgments This research was supported in part by Wright State University; the Ohio Board of Regents; the National Heart and Blood Institute HL055554-11, HL-084529, and HL073598; the National Institute of Environmental Health Sciences P01ES011617; and by the National Science Foundation DMS-0809285.

References

1. Canny JF. A computational approach to edge detection. *IEEE Trans Pattern Anal Mach Intell*. 1986;PAMI-8(6):679–98.
2. Jain R, Kasturi R, Schunck BG. Machine vision. New York: McGraw-Hill, Inc., 1995.
3. Lorensen WE, Cline HE. Marching cubes: a high resolution 3D surface construction algorithm. *Comput Graph*. 1987;21(4).

4. Carson JP, Einstein DR, Minard KR, Fanucchi MV, Wallis CD, Corley RA. Lung airway cast segmentation with proper topology, suitable for computational fluid-dynamic simulations. *Comput Med Imaging Graph.* (submitted).
5. Cornea ND, Silver D, Min P. Curve-skeleton applications. Proceedings of IEEE visualization, 2005, pp. 95–102.
6. Bertrand G, Aktouf Z. A three-dimensional thinning algorithm using subfields. *Vis Geom III*. 1994;2356:113–24.
7. Brunner D, Brunnett G. Mesh segmentation using the object skeleton graph. Proceedings of IASTED international conference on computer graphics and imaging, 2004, pp. 48–55.
8. Dyedov V, Einstein DR, Jiao X, Kuprat AP, Carson JP, del Pin F. Variational generation of prismatic boundary-layer meshes for biomedical computing. *Int J Numer Methods Eng*. 2009;79(8):907–945.
9. Lee T, Kashyap RL, Chu CN. Building skeleton models via 3-D medial surface/axis thinning algorithms. *CVGIP: Graph Model Image Process*. 1994;56(6):462–78.
10. Lohou C, Bertrand G. A 3D 12-subiteration thinning algorithm based on P-simple points. *Discrete Appl Math*. 2004;139:171–95.
11. Palágyi K, Kuba A. Directional 3D thinning using 8 subiterations. Proceedings of discrete geometry for computer imagery, *Lect Notes Comput Sci*. 1999;1568:325–36.
12. Palágyi K, Kuba A. A parallel 3D 12-subiteration thinning algorithm. *Graph Model Image Proc*. 1999;61(4):199–221.
13. Saha PK, Chaudhuri BB, Dutta Majumder D. A new shape preserving parallel thinning algorithm for 3D digital images. *Pattern Recognit*. 1997;30(12):1939–55.
14. Tsao YF, Fu KS. A parallel thinning algorithm for 3-D pictures. *Comput Graph Image Process*. 1981;17:315–31.
15. Lobregt S, Verbeek PW, Groen FCA. Three-dimensional skeletonization: principle and algorithm. *IEEE Trans Pattern Anal Mach Intell*. 1980;2(1):75–7.
16. Sethian JA. Fast marching methods. *SIAM Rev*. 1999;41(2):199–235.
17. Telea A, Vilanova A. A robust level-set algorithm for centerline extraction. Eurographics/IEEE symposium on data visualization, 2003, pp. 185–94.
18. Si H.TetGen. A quality tetrahedral mesh generator and three-dimensional Delaunay triangulator. WIAS Technical Report No. 9, 2004.
19. Wan M, Dachille F, Kaufman A. Distance-field based skeletons for virtual navigation. Proceedings of IEEE visualization, 2001, pp. 239–45.
20. Wischgoll T, Choy JS, Ritman ES, Kassab GS. Validation of image-based extraction method for morphometry of coronary arteries. *Ann Biomed Eng*. 2008;36(3):356–68.
21. Zhou Y, Toga AW. Efficient skeletonization of volumetric objects. *IEEE Trans Vis Comput Graph*. 1999;5(3):196–209.
22. Amenta N, Choi S, Kolluri R.The power crust. Proceedings of 6th ACM symposium on solid modeling, 2001, pp. 249–60.
23. Dey TK, Goswami S. Tight Cocone: a water-tight surface reconstructor. Proceedings of 8th ACM symposium. Solid modeling applications, 127–34. *Journal version in J Comput Inform Sci Eng*. 2003;30:302–7.
24. Nordsletten DA, Blackett S, Bentley MD, Ritman EL, Smith NP. Structural morphology of renal vasculature. *Am J Physiol Heart Circ Physiol*. 2006;291(1):H296–309.
25. Kuprat-AP, Einstein-DR. An anisotropic scale-invariant unstructured mesh generator suitable for volumetric imaging data. *J Comput Phys*. 2009;228:619–40.
26. Jiao X, Zha H. Consistent computation of first- and second-order differential quantities for surface meshes. In ACM solid and physical modeling symposium, 2008.
27. Khamayseh A, Hansen G. Use of the spatial kD-tree in computational physics applications. *Commun Comput Phys*. 2007;2:545–76.
28. Sethian JA. Level set methods and fast marching methods: evolving interfaces in computational geometry, fluid mechanics, computer vision, and materials science. Cambridge: Cambridge University Press, 1999.

29. Jiao X. Face offsetting: a unified approach for explicit moving interfaces. *J Comput Phys*. 2007;220:612–625.
30. Si H. Adaptive tetrahedral mesh generation by constrained Delaunay refinement. *Int J Numer Methods Eng*. 2008;856–80.
31. Cornea ND, Silver D, Yuan X, Balasubramanian R. Computing hierarchical curve-skeletons of 3D objects. *Vis Comput*. 2005;21(11):945–55.
32. Luboz V, Wu X, Krissian K, Westin CF, Kikinis R, Cotin S, Dawson S. A segmentation and reconstruction technique for 3D vascular structures. MICCAI 2005, *Lect Notes Comput Sci*. 2005;3749:43–50.