
Automated Structured All-Quadrilateral and Hexahedral Meshing of Tubular Surfaces

Guanglei Xiong¹, Suraj Musuvathy², and Tong Fang³

¹ Siemens Corporation, Corporate Research and Technology
`guanglei.xiong@siemens.com`

² Siemens Corporation, Corporate Research and Technology
`suraj.musuvathy@siemens.com`

³ Siemens Corporation, Corporate Research and Technology
`tong.fang@siemens.com`

High-quality quadrilateral and hexahedral meshes of tubular structures are useful for many CAD and CAE applications in medical and industrial environments. In this paper, we propose an automated structured meshing strategy for tubular surfaces that possess multiple branches and n-furcations. From a given triangulated mesh, a centerline is extracted and individual branches are identified based on surface decomposition at the junctions. Each branch of the mesh is optimally parameterized to ensure global correspondences and even spacing of contour lines across branches. For each branch, a coarse quadrilateral mesh is created by tracing iso-parameter lines in longitudinal and circumferential directions. The quadrilateral mesh is then approximated with a Catmull-Clark subdivision surface. An initial hexahedral mesh is obtained by computing branch centroids and creating cells by joining adjacent branch centroids and corresponding surface vertices. High quality hexahedral meshes are obtained by refinement of the initial hexahedral mesh using Catmull-Clark solid subdivision. We show the efficiency and robustness of our approach using several real data sets.

1 Introduction

Tubular structures are commonly found in medical (e.g. trees of blood vessels) and industrial (e.g. network of pipes) environments. Although triangular (tri) and tetrahedral (tet) meshes are common geometric representations of the surfaces and volumes respectively, high-quality structured quadrilateral (quad) and hexahedral (hex) meshes are desirable and sometimes a crucial prerequisite in certain situations. For example in computer-aided design (CAD), high quality quad meshes are beneficial to the conversion to tensor-product NURBS surfaces. In computer-aided engineering (CAE), quad/hex meshes

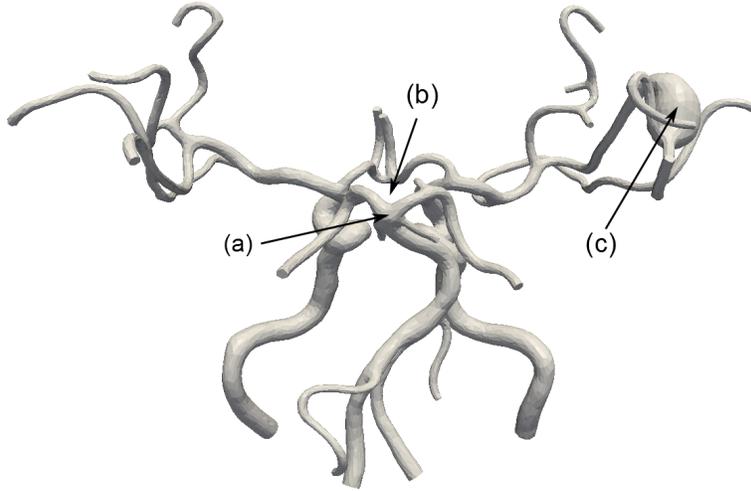


Fig. 1. An example tubular surface reconstructed from CT images of human cerebral arteries, which has (a) a quad-furcation junction; (b) a loop structure (Circle of Willis); and (c) a non-tubular structure (aneurysm).

are preferred in computational solid and fluid mechanics using finite element method (FEM) to tri/tet meshes due to convergence and accuracy improvement [19]. As a generalization of standard FEM, NURBS-based isogeometric analysis [14], which offers great potential in unifying CAD and CAE, also requires hex meshes. However, topological complexity and geometric variations pose unique challenges to structured mesh generation for tubular structures. An example is shown in Fig. 1, which is reconstructed from CT images of human cerebral arteries. It is composed of as many as 41 branches. Some of the branches are long and tortuous. Although most of the junctions are bifurcations, there are occasionally tri-furcations or even quad-furcations. In addition, the surface has a loop and a non-tubular structure.

In this paper, we propose a new strategy to generate structured quad and hex meshes automatically from given triangulated surfaces of tubular shapes. The proposed approach handles arbitrary branching topology by using a novel branch decomposition technique and a new mesh parameterization technique that is consistent across branch junctions. The resulting hex meshes have high quality shapes at the individual element level, with the elements having positive scaled Jacobian values mostly close to the ideal value of 1.0, as well as globally smooth shapes due to Catmull-Clark subdivision. The elements are aligned in a natural manner along the cylindrical axis of the tubular structures. Further, the outer boundary of the generated hex meshes approximates the input triangulated surface within a user specified accuracy. Branch segments are identified and labeled automatically, thereby enabling specification and analysis of the data at the level of individual branches. Fig. 12 shows a hex

mesh computed for the triangulated surface mesh shown in Fig. 1 using the approach presented in this paper.

2 Related Work

Many algorithms, mainly categorized as grid-based and advancing front, have been proposed for unstructured quad/hex meshing and have attained varying amounts of success during the past two decades. We refer to Shepherd [22] for a comprehensive survey. However, more structured, ideally cylindrical axis aligned, meshes are usually favorable for tubular structures. The advantages are two-fold. One is that it requires fewer elements to capture the surface details with structured quads/hexes. The other is that it simplifies the subsequent operations, such as mesh refinement, boundary layer meshing, and conversion to surface or volumetric NURBS. Works that aim at generating structured meshes for the bifurcation geometry are fewer in the current literature. Antiga et al. [1] generated hex meshes for carotid bifurcations by applying the Cooper tool [4] to the decomposed geometry. Verma et al. [24] decomposed bifurcation geometry by solving three sets of Laplace equations on the tet meshes and swept a template grid to create hex meshes [16]. Santis et al. [8] fit longitudinal splines from surface slicing and then used them to guide sweeping. Makris et al. [21] partitioned a bifurcation into two blocks, one for each branch, and union them to obtain a structured hex mesh.

Extending from a single bifurcation, recent developments have shifted to more complicated geometry. Zhang et al. [25] designed templates for handling various branching configurations and successfully constructed hex meshes and solid NURBS using sweeping and surface fitting for geometry with multiple branches. Yet, the need to select and orient the template for every junction in patient-specific cases limits the procedure from full automation. Santis et al. [9] created multiple block-structures and projected them to the surface. Hex meshes for multi-branch geometry were generated by isotropic refinements. However, it is necessary to manually adjust the initial placements of block-structures, which is often cumbersome and can take a significant amount of time.

Although these methods enable the structured hex meshing, no fully automated method to mesh complicated geometry like in Fig. 1 is available today to our knowledge. Quad/Hex mesh generation, particularly for tubular structures, remains a major bottleneck in CAD and CAE applications.

3 Overview

In this paper, we propose a new strategy to generate structured quad and hex meshes automatically from pre-existing triangulated surfaces. Our basic

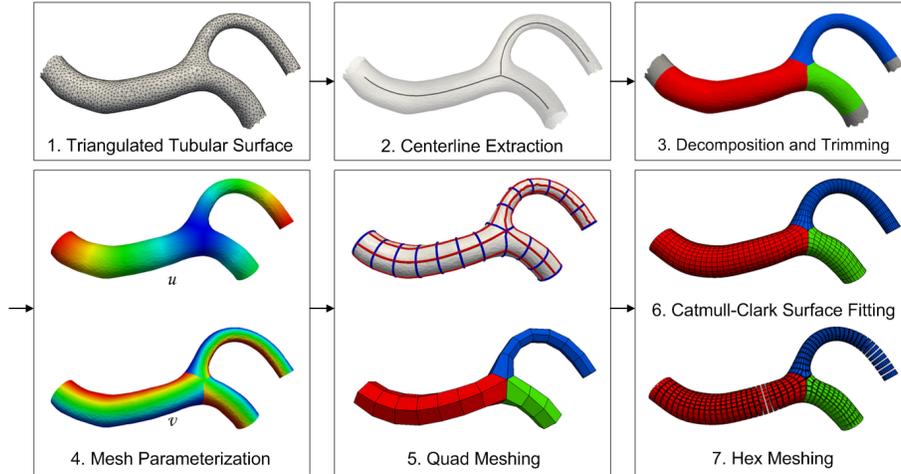


Fig. 2. The proposed meshing workflow demonstrated using a simple bifurcation.

idea stems from a divide and conquer paradigm. Fig. 2 shows the proposed meshing workflow. At a high level, it includes the following steps:

1. *Triangulated Tubular Surface*: The input mesh is required to be manifold. We define a branch by the region between two junctions. A junction is where three or more branches meet.
2. *Centerline Extraction*: We extract the centerline using mesh contraction from the input surface. Based on the centerline structure, the topological composition of the tubular surface is obtained.
3. *Decomposition and Trimming*: Guided by the topological information, the random walker algorithm is used to subdivide the surface at the junctions. In order to remove irregularities at the ends, we trim the surface with planar cuts at the open (non-junction) ends. Both decomposition and trimming lead to boundary curves that finally partition the whole surface into generalized cylinders, one for each individual branch. Boundary curves are also split into segments based on branch neighborhood information.
4. *Mesh Parameterization*: We parameterize each branch surface using Laplacian-based harmonic functions. In a natural way, one parameter is in the longitudinal direction and the other is in circumferential direction. To ensure global correspondences and even spacing of contour lines across branches, the selection of parameter values is optimized using constrained integer programming.
5. *Quad Meshing*: We trace the iso-parameter lines in both longitudinal and circumferential directions. An initial quad mesh is constructed by linking the crossing nodes of the iso-parameter lines.

6. *Catmull-Clark (CC) Surface Fitting*: The parameterized quad mesh is approximated by a CC subdivision surface. A high quality quad mesh is generated by sampling the CC (limit) surface.
7. *Hex Meshing*: A volumetric CC subdivision solid is created from the fitted CC subdivision surface. Finally, a hex mesh is generated by refining the CC subdivision solid.

4 Workflow

We now present a more detailed description of the intermediate steps of the proposed workflow in the following sections.

4.1 Centerline Extraction

The centerline provides a natural structural representation of tubular structures. Although many algorithms for computing centerlines exist, few are both automated and robust. We employ a method to extract the centerline based on mesh contraction [2]. In our experiments, this method produced high-quality centerlines, even for complicated geometry. Briefly, this method includes two steps. In the first step, the input mesh is contracted with Laplacian-based smoothing to obtain a new mesh that is geometrically similar to a centerline-like structure but topologically identical to the original mesh. In the next step, the topology is simplified by edge collapse operations until a 1D centerline is obtained (Fig. 3 (a)). By analyzing the constituent branches (blue dots in Fig. 3 (a)), junction and ending (red dots in Fig. 3 (a)) nodes of the centerline, we can identify the topological composition of the tubular structure. In addition, the mapping between a centerline node and its associated vertices of the input mesh is recorded during edge collapse, which facilitates the branch decomposition step described in the next section.

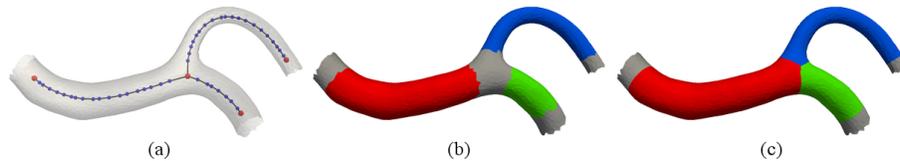


Fig. 3. (a) Centerline extraction; (b, c) branch decomposition and trimming.

4.2 Decomposition and Trimming

Guided by the centerline, an initial decomposition is carried out by dividing the surface into branch (colored areas in Fig. 3 (b)), junction and ending

(gray areas in Fig. 3 (b)) components. It should be noted that the surface has to be further decomposed since the junction component also encloses regions from branches, while our goal is to make every branch topologically equivalent to a cylinder and seamlessly joined at the junction (e.g. in Fig. 3 (c)). Another observation of the initial decomposition is that the boundaries between different components are jagged, especially when the triangle density is low.

We adopt a mesh segmentation method based on random walks [12, 17] to further decompose the junctions. Besides the robustness and efficiency, one advantage of the method is that it guarantees each partition produced is contiguous, i.e. no *island* is created. While the original method was triangle-based, we choose to assign region labels to vertices instead because it leads to smoother boundaries in combination with a triangle splitting procedure. The boundary curves between junction and branch regions from the initial decomposition are selected as the *seeds*. The number of seeds is determined by the number of branches joining at the junction of interest, denoted by \mathcal{B} . We define the probability of random walks, $p_{i,j}$, between two neighboring vertices v_i and v_j as

$$p_{i,j} = \exp \frac{\langle N_i, N_j \rangle}{|e_{i,j}|} \quad (1)$$

, where N is the normal and $e_{i,j}$ is the edge between v_i and v_j . We also define \mathcal{L}_j^b as the likelihood of v_j belonging to branch b , which satisfies a system of linear equations $\mathcal{L}_i^b = \sum_{e_{i,j}} p_{i,j} \mathcal{L}_j^b$, $b = 1 \dots \mathcal{B}$. The final label that assigns to the vertex v_j is $\arg \max_b \mathcal{L}_j^b$.

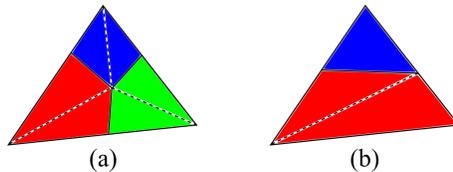


Fig. 4. Triangle splitting cases: (a) three and (b) two different vertex labels.

In order to find the boundary curves, triangles with different vertex labels have to be split. For these triangles, there are only two cases shown in Fig. 4. In both cases, solid segments will be part of boundary curves after splitting, whereas dashed ones are inserted to divide quadrilaterals. The locations of new vertices are determined by linear weighting with corresponding likelihoods as weights. A side effect of triangle splitting is that it is guaranteed that no more than 3 different branch regions can meet at a given vertex on the surface no matter how many branches actually join at the junction. Fig. 5 illustrates some decomposition results.

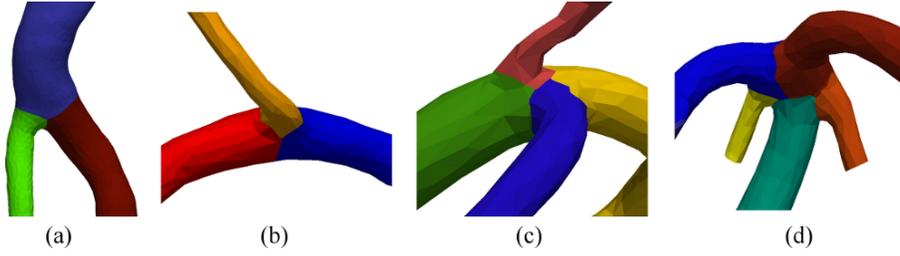


Fig. 5. Decomposition examples at junctions with (a,b) three, (c) four, (d) five branches.

In order to remove irregularities at non-junction ends, we trim the surface with planar cuts through centerline ending points as shown in Fig. 3 (c).

4.3 Mesh Parameterization

An advantage of the surface decomposition is that mesh parameterization is simplified to a problem of parameterizing each single branch. As in Fig. 6, we naturally assign two parameters u and v to be in the longitudinal and circumferential directions, respectively. To simplify the construction of quad mesh structures, we choose the integer-valued intervals of the parameters as $u = 1 \dots \mathcal{U}_b$ and $v = 1 \dots \mathcal{V}_b$, at which iso-parameter lines will be traced in the next step. In the longitudinal direction, \mathcal{U}_b can be simply determined as $\mathcal{U}_b = \alpha L_b^u$, where L_b^u be the length of the tube and α controls the intervals per unit length in the longitudinal direction. Circumferential direction is more complicated because global constraints must be satisfied to ensure the iso-parameter lines properly joined at the junctions. First, we mark the boundary contours for each branch. As a cylinder shape, there should be exactly two contours on either end. Based on adjacency relationship with its neighboring branches, each contour is then divided into \mathcal{S}_b curve segments, which is denoted as $s_{b,e} = 1 \dots \mathcal{S}_{b,e}$, where $b = 1 \dots \mathcal{B}$ and $e = 1, 2$ indicates which end. Note that there may be more segments than the number of neighboring branches because the boundary between two branches can be disconnected. Next, we find the interval of v for each curve segment, which is denoted as $\mathcal{V}_{s_{b,e}}$. The following constraints are considered: (1) Intervals corresponding to the same curve segment as seen from two different branches must be equal; (2) The sums of intervals on both ends of a branch must be equal; (3) The interval of each curve segment $\mathcal{V}_{s_{b,e}}$ is roughly proportional to its length $L_{s_{b,e}}^v$. The former two are hard equality constraints, whereas the latter one is a soft inequality constraint. Due to the dependency of intervals among all branches, we solve this problem using a constrained optimization involving integer variables. The objective is to minimize the sum of all intervals. Mathematically,

$$\begin{aligned}
& \text{Minimize : } \sum_{b=1}^{\mathcal{B}} \sum_{e=1}^2 \sum_{s_{b,e}=1}^{S_{b,e}} \mathcal{V}_{s_{b,e}} \\
& \text{Subject to : } \mathcal{V}_{s_{b,e}} = \mathcal{V}_{s'_{b',e'}}, \text{ if } s_{b,e} = s'_{b',e'} \\
& \sum_{s_{b,1}=1}^{S_{b,1}} \mathcal{V}_{s_{b,1}} = \sum_{s_{b,2}=1}^{S_{b,2}} \mathcal{V}_{s_{b,2}} \\
& \beta L_{s_{b,e}}^v \leq \mathcal{V}_{s_{b,e}} \leq \gamma L_{s_{b,e}}^v
\end{aligned} \tag{2}$$

To map the tubular surface to a rectangular parameter domain, a splitting line is determined as shown in Fig. 6, which also establishes the correspondence between the references $v = 1$ on both ends. Care must be taken to suppress the twisting of this splitting line for long and tortuous tubes. We first choose a vertex corresponding to $v_{b,1} = 1$ on one end. Next, we search for a shortest path using Dijkstra's algorithm that connects this vertex to another vertex $v_{b,2} = 1$ on the other end. The following edge cost function is adopted:

$$C(e_{i,j}) = |1 - \mathbf{t}_j \cdot \mathbf{e}_{i,j}| + |1 + \mathbf{r}_j \cdot \mathbf{n}_j| \tag{3}$$

, where $\mathbf{t}_j, \mathbf{r}_j$ are the tangent and radial directions of the centerline at vertex v_j , $\mathbf{e}_{i,j}$ is the direction from v_i to v_j , and \mathbf{n}_j is the normal direction at v_j . With the u and v parameter values defined on two boundary contours and the splitting line, we finally parameterize the tube using Laplacian-based harmonic functions [11]. Note the vertices on the boundary contours, which are shared by two or more branches, have different parameter values (for both u and v) as seen from different branches. However, the number of intervals (for v) of each contour segment is always identical. An example of mesh parameterization can be seen in Fig. 2(4).

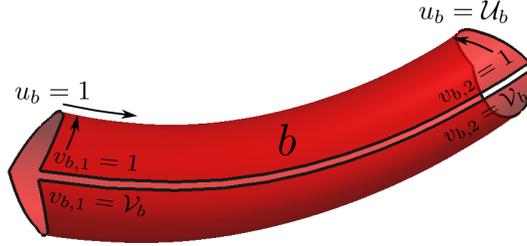


Fig. 6. Mesh parameterization for a single tube. u and v are in longitudinal and circumferential directions, respectively.

4.4 Initial Quad Mesh Generation

Once the parameterization is done, a quad mesh structure can be obtained by tracing the iso-parameter lines at integer-valued parameters on the surface. In

the longitudinal direction, branch decomposition contours or trimming contours serve as the iso- u lines on either end of a branch. The intermediate lines are determined by first inserting new vertices corresponding to integer-valued u 's on the splitting line and then tracing circumferentially, for each vertex, until it reaches the same vertex. Similarly in the circumferential direction, the iso- v lines are found by first inserting new vertices corresponding to integer-valued v 's on boundary segments on both ends and then tracing from the inserted vertex from on one end to the other. An initial quad mesh for each branch is constructed by linking the crossing nodes of the iso-parameter lines. Finally, the quad mesh for the whole surface is obtained by merging shared vertices at the decomposition contours. An example of iso-parameter lines tracing and quad meshing is shown in Fig. 2(5).

4.5 Catmull-Clark Surface Fitting and Quad Mesh Generation

CC subdivision surfaces, introduced by Catmull and Clark [6], are widely used in modeling high quality smooth shapes in computer graphics [10]. They are a generalization of tensor product bicubic B-spline surfaces that address extraordinary (irregular) vertex topology. A CC surface is specified by a *base* mesh of arbitrary topology that defines a smooth surface as a limit of recursively defined refinement operations. The *limit* surface is $C^{(2)}$ at all locations except at extraordinary vertices of the base mesh, where the limit surface is $C^{(1)}$. Fig. 7 shows an example of the CC surface subdivision scheme. A CC base mesh is specified by six quads forming a cube (Fig. 7(a)). Resulting quadrilateral mesh surfaces obtained by applying the subdivision iterations are shown in Figs. 7(b) and 7(c). The limit surface is shown in Fig. 7(d). The meshes obtained by CC refinement are guaranteed to contain only quads.

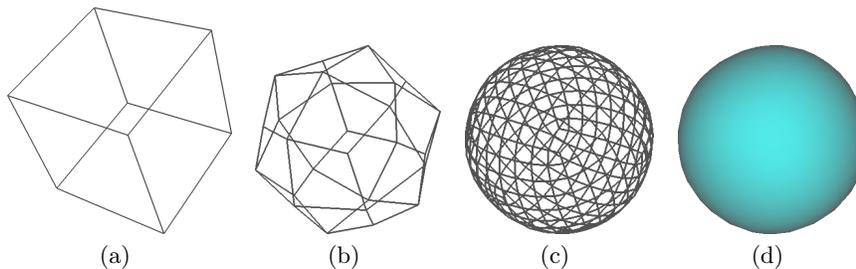


Fig. 7. CC surface subdivision example. (a) A CC surface with six quad faces. (b) Quad mesh after one iteration of CC surface subdivision. (c) Quad mesh after three iterations of CC surface subdivision. (d) The limit surface.

In this step of the proposed workflow, the parameterized quad mesh is approximated by a CC subdivision surface. The parameterized quad mesh is used as an initial CC surface base mesh. The fitting algorithm optimizes the

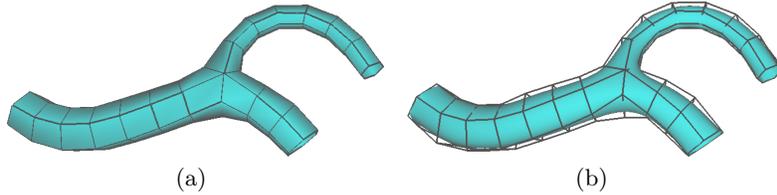


Fig. 8. (a) The initial quad mesh computed from parameterized surface. (b) CC surface that approximates the input surface. The CC base mesh is shown in wireframe and the limit surface is shown in cyan.

locations of the vertices of the base mesh such that the limit surface of the fitted CC subdivision mesh is within a user specified error tolerance of the input surface. Several techniques for fitting CC surfaces have been previously proposed [13, 20, 18, 7]. A progressive-iterative approximation scheme [7] is chosen in our work for fitting the CC surface since it is computationally fast and scales well with the number of vertices in the mesh being optimized.

We present a brief summary of the progressive-iterative approximation scheme here. Let \mathcal{M}^0 as the initial CC surface base mesh and \mathcal{M}^k be the CC surface mesh after k iterations. In the $k + 1^{th}$ iteration, for each vertex of the mesh, $v_{\mathcal{M}^k}$, compute its limit point on the CC surface $v_{\mathcal{M}^k}^\infty$ as presented in [7]. Let $d(v_{\mathcal{M}^k}) = v_{\mathcal{M}^0} - v_{\mathcal{M}^k}^\infty$. If $\|d(v_{\mathcal{M}^k})\| > \epsilon$, where ϵ is the maximum allowed error, then $v_{\mathcal{M}^{k+1}} = v_{\mathcal{M}^k} + d(v_{\mathcal{M}^k})$. The algorithm stops when there are no vertex updates. Fig. 8 (also see Fig. 2(6)) shows an initial quad mesh obtained from the parameterized input surface and the corresponding fitted CC surface that approximates it. The limit surface is represented as an all-quad mesh.

4.6 Hex Meshing

The CC subdivision scheme has been extended to volumetric meshes [15, 3] where the base mesh consists of volume elements (hexahedra, prisms, tetrahedra, etc) and the refined meshes are guaranteed to consist of hexahedral elements. Fig. 9 shows an example of the CC volume subdivision scheme. A CC base solid is specified by two hexahedral elements (Fig. 9(a)). Resulting hexahedral meshes obtained by applying the subdivision iterations are shown in Figs. 9(b) and 9(c).

From the fitted CC surface, a volumetric mesh is generated by computing centroids of all iso-parameter lines along the circumferential direction and connecting them to the surface vertices as well as the adjacent centroids. At the branch junctions, all shared vertices on the branch segments are used to compute a single centroid node as shown in Fig. 10. This results in a consistent volume mesh of the entire geometry that consists of prism elements. Then, CC solid subdivision rules are applied to generate the hexahedral elements. A single step of the subdivision will create a mesh consisting of only hexahedral

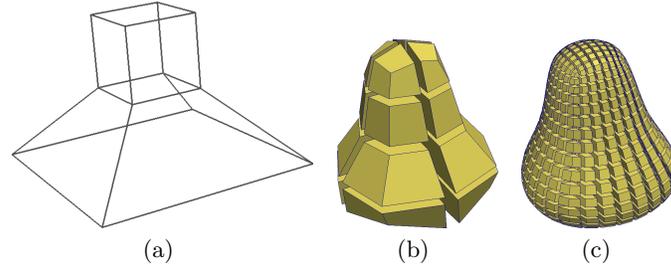


Fig. 9. CC volume subdivision example. (a) A CC solid with two hexahedral elements. (b) Hexahedral mesh after one iteration of CC volume subdivision. (c) Hexahedral mesh after three iterations of CC volume subdivision.

elements. The subdivision iterations are performed until the boundary surface of the hexahedral mesh is within an error tolerance of the input surface mesh. As a result of the subdivision, the generated geometry is smooth with high quality elements. Fig. 11 shows an initial volumetric mesh computed from the fitted CC base mesh and the corresponding hexahedral mesh generated by subdivision (also see Fig. 2(7)).

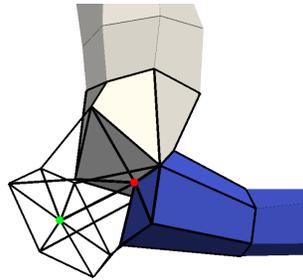


Fig. 10. Placement of centroid nodes at a junction. The edges of cells incident on a junction are shown in black. The centroid node at the junction is shown in red. Another centroid from an incident branch on the junction centroid is shown in green.

5 Results

This section presents the results of applying the proposed workflow to real-world geometries constructed from medical imaging data. For all the examples presented here, the CC surface fitting error tolerance is specified as 0.001 of the diagonal size of the surface bounding box. Fig. 1 shows a complicated data set consisting of human cerebral arteries. The data set has many branches that are long tortuous tubes of significantly varying diameters meeting at

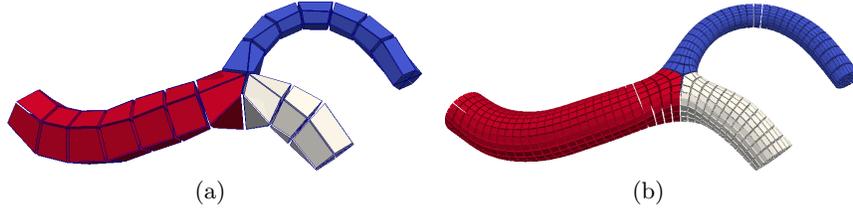


Fig. 11. (a) The initial prism hex mesh computed from fitted CC surface. (b) Hex mesh generated by volume subdivision. The elements of each branch are shown in a different color.

several branch junctions. It includes a tri-furcation and a quad-furcation, as well as a loop of branch structures. Fig. 12 shows the results of intermediate steps of the proposed method for this dataset. Fig. 12(a) shows the centerline computed from the surface mesh. The branches of the centerline are shown as blue curves and branch junction points are indicated by red spheres. Fig. 12(b) shows the surface decomposition. Each branch is indicated by a different color. Fig. 12(c) shows an initial quad mesh computed by surface parameterization and iso-parameter line tracing. Finally, Fig. 12(d) shows the hex mesh for this data set. The maximum distance of the outer surface from the input triangulated mesh surface for this example is 0.018 of the model size after two steps of CC refinement. Figs. 12(e) and 12(f) show magnified views of the Circle of Willis and aneurysm regions, respectively. Hex elements are shrunk to better illustrate element shapes. Fig. 12(g) shows a histogram of the scaled Jacobian values of the hex elements. This figure indicates that a majority of the elements are close to 1.0. More than 90% of the elements have a scaled Jacobian value greater than 0.68.

Fig. 13 shows the result on an abdominal aortic artery data set, also with an aneurysm. Fig. 13(a) shows the centerline, Fig. 13(b) shows the surface branch decomposition, Fig. 13(c) shows the initial quad mesh and Fig. 13(d) shows the hex mesh. Figs. 13(e) and 13(f) show magnified views of the renal and iliac bifurcation regions. A histogram of the scaled Jacobian values of the hex elements shown in Fig. 13(g) indicates the quality. Figs. 14 and 15 show results of applying the proposed method to healthy coronary artery data sets. The data sets contain branches of varying diameters that exhibit several bifurcations. These results indicate that the proposed method generates high quality hex meshes for tubular structures.

The characteristics of the meshes is detailed in Table 1 and running time characteristics are presented in Table 2. For the examples presented in this paper, the method takes a few seconds to a few minutes on a recent desktop personal computer. While the most time consuming step is the CC solid subdivision, much improvement can be made by replacing our current serial implementation with a parallel one.

Table 1. Input and output mesh characteristics

Dataset	Input tris	Output quads	Output hexes	Max relative distance	Scaled Jacobian		
					Min	Median	Max
Fig. 11	5k	2k	5k	0.02	0.4412	0.9632	0.9936
Fig. 14	14k	16k	46k	0.01	0.4425	0.9591	0.9942
Fig. 15	13k	12k	36k	0.01	0.0339	0.9539	0.9945
Fig. 13	41k	22k	65k	0.009	0.1129	0.9220	0.9996
Fig. 12	20k	160k	474k	0.018	0.0017	0.9381	0.9968

Table 2. Running time characteristics (seconds)

Dataset	Centerline	Decomp.	Param.	Quad.	CC surface	Hex.	Total
Fig. 11	1.63	2.58	0.09	0.08	0.16	1.61	6.15
Fig. 14	3.21	3.92	0.25	0.20	1.42	16.88	25.88
Fig. 15	2.17	1.84	0.28	0.24	1.00	12.43	17.96
Fig. 13	9.11	6.86	0.92	0.61	2.58	25.47	45.55
Fig. 12	3.70	4.18	0.44	0.51	61.33	213.77	283.93

6 Conclusions

In summary, we have proposed a new workflow to generate all-quad surface meshes and all-hex volumetric meshes of tubular structures from triangulated surface meshes. The method consists of a sequence of steps including 1) centerline extraction; 2) surface decomposition and trimming; 3) mesh parameterization, 4) quad meshing by iso-parameter line tracing, 5) CC surface fitting; 6) hex meshing by CC solid subdivision and refinement. In contrast to previous methods, our approach is fully automated and is capable of handling not only bifurcations but also higher-order branching geometry. The method generates hex elements that are naturally aligned along the length of the tubular structures and are of high quality as indicated from the all-positive scaled Jacobian values. In addition, the boundary surface of the hex mesh is approximated to the input surface within a user specified accuracy. Furthermore, the classification of hex elements into individual branches facilitates branch-wise specification and analysis.

We present several results on data sets arising from the medical field. These examples are complex tubular structures that are highly tortuous and contain junctions where branches of significantly differing diameters meet. Our automated approach has the potential to stimulate computational simulation of fluid flow in tubular structures using hex meshes, where tet meshes currently dominate. In addition, although isotropic meshing is our focus in this paper, our approach can be extended to generate locally-refined or boundary-layer meshes as well. While refinement in the circumferential direction can be achieved by simply inserting iso- u lines within a branch, refinement in the longitudinal direction does require propagating the iso- v to other branches to avoid T-junctions.

There are some limitations in our current work that calls for future work. First, the branch decomposition procedure may fail if the branch length or the distance between two junctions is too short. This failure can be sometimes avoided by considering the two junctions as a single one. Next, having a single centroid node during prism element construction may lead to a few internal elements of poor quality at the junction. Different ways to place internal nodes may be explored [23].

Ongoing work includes applying computational analysis simulations using the generated hex meshes. Although our goal in this paper is to compute hex meshes, recent methods have proposed performing simulations directly on CC volumes [5]. Thus, we believe that the fitted CC solids generated by our method can be directly applicable for CC based higher order analysis as well.

References

1. ANTIGA, L., ENE-IORDACHE, B., CAVERNI, L., PAOLO CORNALBA, G., AND REMUZZI, A. Geometric reconstruction for computational mesh generation of arterial bifurcations from ct angiography. *Computerized Medical Imaging and Graphics* 26, 4 (2002), 227–235.
2. AU, O., TAI, C., CHU, H., COHEN-OR, D., AND LEE, T. Skeleton extraction by mesh contraction. In *ACM Transactions on Graphics (TOG)* (2008), vol. 27, ACM, p. 44.
3. BAJAJ, C., SCHAEFER, S., WARREN, J., AND XU, G. A subdivision scheme for hexahedral meshes. *The Visual Computer* 18, 5 (2002), 343–356.
4. BLACKER, T. The cooper tool. In *Proceedings of the 5th International Meshing Roundtable* (1996), Springer, pp. 13–29.
5. BURKHART, D., HAMANN, B., AND UMLAUF, G. Iso-geometric finite element analysis based on catmull-clark subdivision solids. In *Computer Graphics Forum* (2010), vol. 29, Wiley, pp. 1575–1584.
6. CATMULL, E., AND CLARK, J. Recursively generated b-spline surfaces on arbitrary topological meshes. *Computer-Aided Design* 10, 6 (1978), 350–355.
7. CHEN, Z., LUO, X., TAN, L., YE, B., AND CHEN, J. Progressive interpolation based on catmull-clark subdivision surfaces. In *Computer Graphics Forum* (2008), vol. 27, Wiley, pp. 1823–1827.
8. DE SANTIS, G., DE BEULE, M., SEGERS, P., VERDONCK, P., AND VERHEGGHE, B. Patient-specific computational haemodynamics: generation of structured and conformal hexahedral meshes from triangulated surfaces of vascular bifurcations. *Computer Methods in Biomechanics and Biomedical Engineering* 14, 9 (2011), 797–802.
9. DE SANTIS, G., DE BEULE, M., VAN CANNEYT, K., SEGERS, P., VERDONCK, P., AND VERHEGGHE, B. Full-hexahedral structured meshing for image-based computational vascular modeling. *Medical Engineering & Physics* 33, 10 (2011), 1318–1325.
10. DEROSE, T., KASS, M., AND TRUONG, T. Subdivision surfaces in character animation. In *Proceedings of the 25th Annual Conference on Computer Graphics and Interactive Techniques* (1998), ACM, pp. 85–94.

11. DESBRUN, M., MEYER, M., AND ALLIEZ, P. Intrinsic parameterizations of surface meshes. *Computer Graphics Forum* 21, 3 (2002), 209–218.
12. GRADY, L. Random walks for image segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 28, 11 (2006), 1768–1783.
13. HALSTEAD, M., KASS, M., AND DEROSE, T. Efficient, fair interpolation using catmull-clark surfaces. In *Proceedings of the 20th Annual Conference on Computer Graphics and Interactive Techniques* (1993), ACM, pp. 35–44.
14. HUGHES, T., COTTRELL, J., AND BAZILEVS, Y. Isogeometric analysis: Cad, finite elements, nurbs, exact geometry and mesh refinement. *Computer Methods in Applied Mechanics and Engineering* 194, 39 (2005), 4135–4195.
15. JOY, K., AND MACCRACKEN, R. The refinement rules for catmull-clark solids. Tech. Rep. CSE-91-1, Computer Science Department, University of California, Davis, 1996.
16. KNUPP, P. Next-generation sweep tool: a method for generating all-hex meshes on two-and-one-half dimensional geometries. In *Proceedings of the 7th International Meshing Roundtable* (1998), Springer, pp. 505–513.
17. LAI, Y., HU, S., MARTIN, R., AND ROSIN, P. Rapid and effective segmentation of 3d models using random walks. *Computer Aided Geometric Design* 26, 6 (2009), 665–679.
18. LITKE, N., LEVIN, A., AND SCHRÖDER, P. Fitting subdivision surfaces. In *Proceedings of the Conference on Visualization '01* (2001), vol. 2001, IEEE, pp. 319–324.
19. LONGEST, P., KLEINSTREUER, C., AND DEANDA, A. Numerical simulation of wall shear stress and particle-based hemodynamic parameters in pre-cuffed and streamlined end-to-side anastomoses. *Annals of Biomedical Engineering* 33, 12 (2005), 1752–1766.
20. MA, W., AND ZHAO, N. Catmull-clark surface fitting for reverse engineering applications. In *Geometric Modeling and Processing 2000. Theory and Applications. Proceedings* (2000), IEEE, pp. 274–283.
21. MAKRIS, E., NEOFYTOU, P., TSANGARIS, S., AND HOUSIADAS, C. A novel method for the generation of multi-block computational structured grids from medical imaging of arterial bifurcations. *Medical Engineering & Physics* (2011).
22. SHEPHERD, J. *Topologic and geometric constraint-based hexahedral mesh generation*. PhD thesis, The University of Utah, 2007.
23. STATEN, M., CANANN, S., AND OWEN, S. Bmsweep: locating interior nodes during sweeping. In *Proceedings of the 7th International Meshing Roundtable* (1998), Springer, pp. 7–18.
24. VERMA, C., FISCHER, P., LEE, S., AND LOTH, F. An all-hex meshing strategy for bifurcation geometries in vascular flow simulation. In *Proceedings of the 14th International Meshing Roundtable* (2005), Springer, pp. 363–375.
25. ZHANG, Y., BAZILEVS, Y., GOSWAMI, S., BAJAJ, C., AND HUGHES, T. Patient-specific vascular nurbs modeling for isogeometric analysis of blood flow. *Computer Methods in Applied Mechanics and Engineering* 196, 29-30 (2007), 2943–2959.

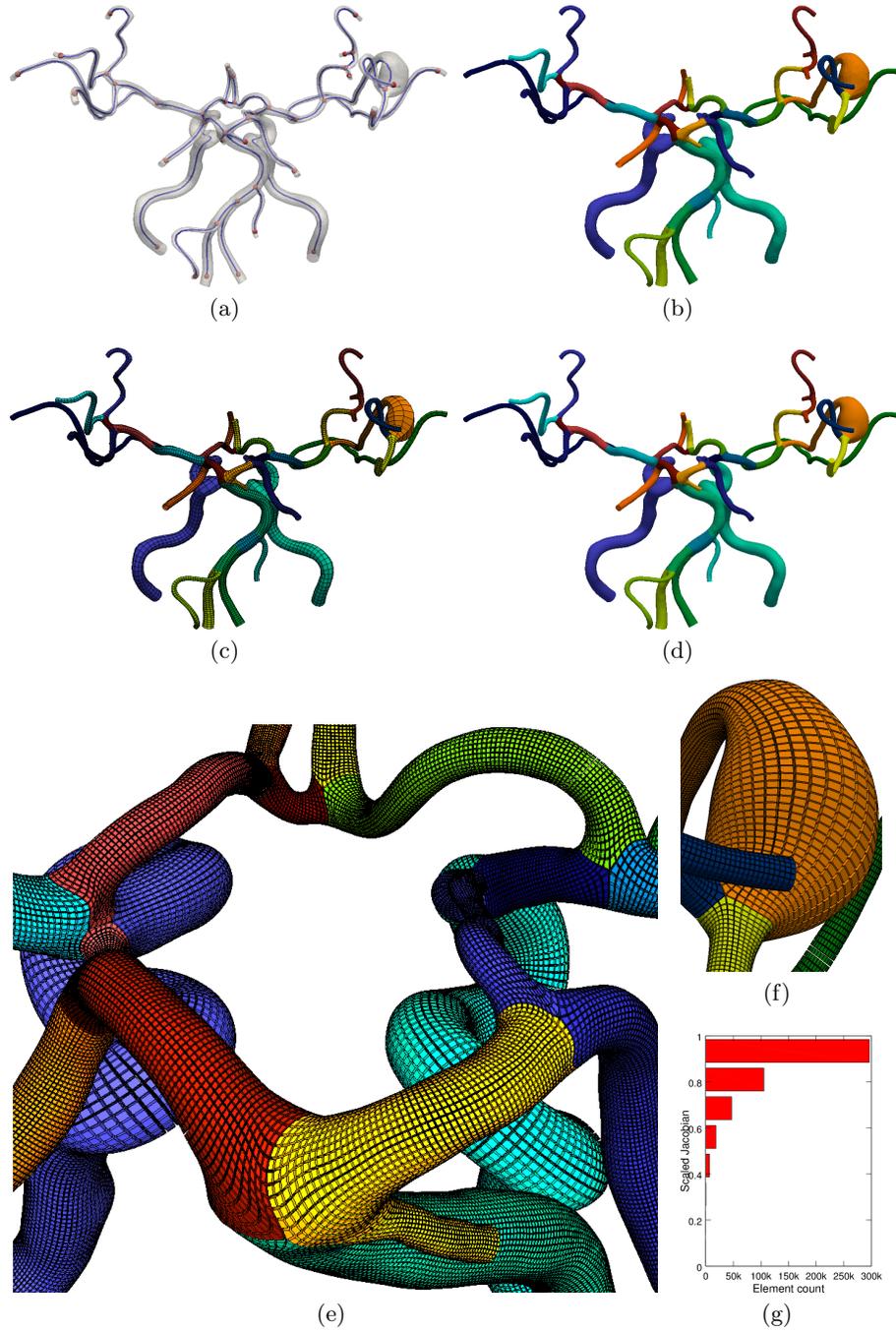


Fig. 12. Results for the geometry shown in Fig. 1. (a) Centerline extraction. (b) Branch decomposition and trimming. (c) Initial quad meshing. (d) Hex meshing. (e), (f) Magnified views of hex elements (shrunk) in the regions of the Circle of Willis and aneurysm. (g) Histogram of the scaled Jacobian values of (d).

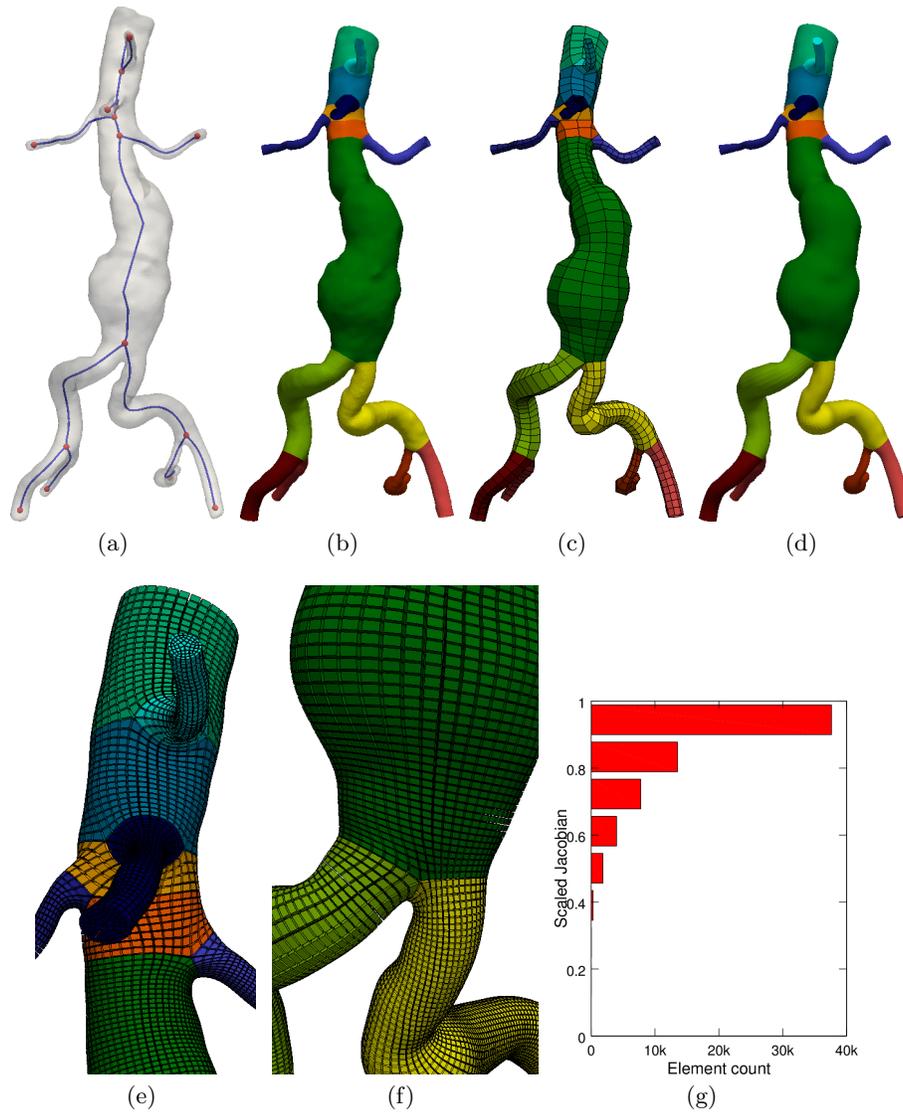


Fig. 13. Results for the abdominal aortic artery geometry. (a) Centerline extraction. (b) Branch decomposition and trimming. (c) Initial quad meshing. (d) Hex meshing. (e), (f) Magnified views of hex elements (shrunk) in the regions of renal and iliac bifurcations. (g) Histogram of the scaled Jacobian values of (d).

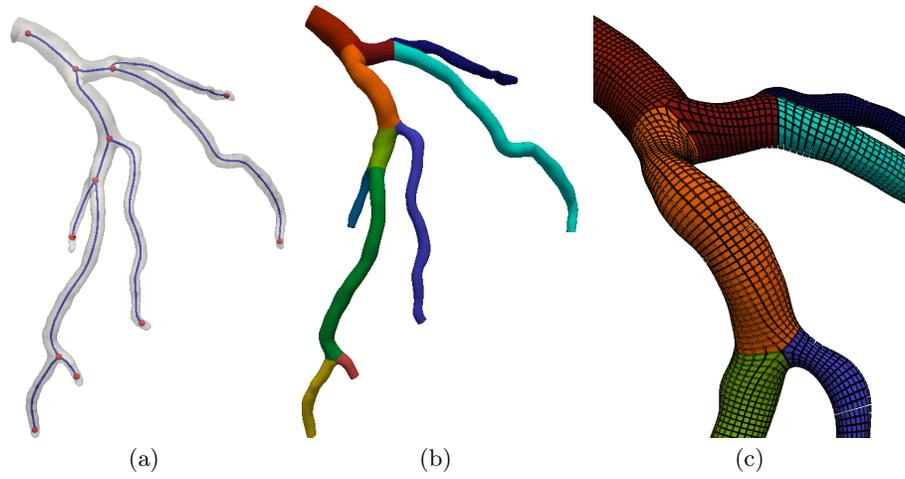


Fig. 14. Results for the left coronary artery geometry. (a) Centerline extraction. (b) Hex meshing. (c) Magnified view of hex elements (shrunken) in the region of bifurcations.

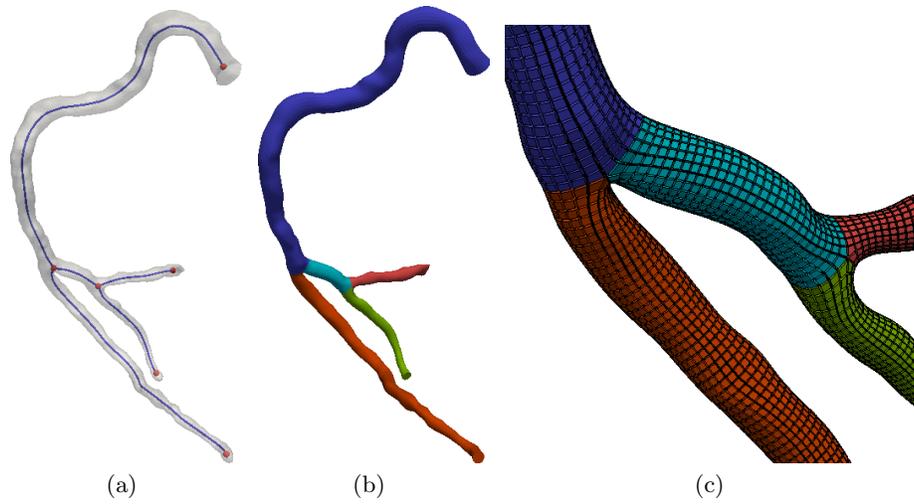


Fig. 15. Results for the right coronary artery geometry. (a) Centerline extraction. (b) Hex meshing. (c) Magnified view of hex elements (shrunken) in the region of bifurcations.