Topology aware Quad Dominant Meshing for Vascular Structures

Dominik Sibbing, Hans-Christian Ebke, Kai Ingo Esser, and Leif Kobbelt

Computer Graphics and Multimedia
RWTH Aachen University Germany
{sibbing,ebke,esser,kobbelt}@cs.rwth-aachen.de
www.graphics.rwth-aachen.de

Abstract. We present a pipeline to generate high quality quad dominant meshes for vascular structures from a given volumetric image. As common for medical image segmentation we use a Level Set approach to separate the region of interest from the background. However in contrast to the standard method we control the topology of the deformable object – defined by the Level Set function – which allows us to extract a proper skeleton which represents the global topological information of the vascular structure. Instead of solving a complex global optimization problem to compute a quad mesh, we divide the problem and partition the complex model into *junction* and *tube* elements, employing the skeleton of the vascular structure. After computing quad meshes for the *junctions* using the Mixed Integer Quadrangulation approach, we re-mesh the *tubes* using an algorithm inspired by the well known Bresenham Algorithm for drawing lines which distributes irregular elements equally over the entire *tube* element.

Keywords: Level Sets, Quad dominant meshes

1 Introduction

Modern medical imaging techniques like, e.g, Magnetic Resonance Imaging (MRI) or X-ray computed tomography (CT) are able to capture many details of anatomical structures. Often however the relevant information is hidden by additional data, simultaneously collected during the scan. Besides this, a pure image based representation is often insufficient for subsequent processing. Mesh based representations on the other hand open the door to a broad range of applications, like physical simulation (fluid simulation, deformable objects), high quality renderings (global illumination and shadows) or even modeling and animation. Such polygonal meshes often have to fulfill some application specific quality requirements. Compared to commonly used unstructured triangular representations, shape-adaptive quad meshes have proven to be visually more pleasant, require fewer elements, produce much more stable results for Finite Elements Methods and enable the implementation of intuitive modeling metaphors.

In this paper we focus on angiogram data obtained from an MRI scanner. The task is to compute a quad dominant mesh representing the vascular structure embedded in a volume image. Since we are using a variant of the Level Set method which is able to control the topology of the deformable object, the topology of our extracted mesh is a plausible interpretation of the topology of the vascular structure. We employ this topological information to break down the difficult global quad meshing task of the entire vessel tree into a set of simpler sub-tasks namely the individual meshing of *junctions* and *tube* elements. Our *tube* meshing approach is not only faster than a global mesh generation approach but it is also able to effectively adapt the quad's aspect ratio and thereby needs fewer polygons to approximate the surface.

1.1 Related Work

Direct volume rendering is a technique to visualize the relevant information contained in a volumetric image in real-time [1]. A transfer function is used to highlight the region of interest and blend out uninteresting regions. Although this technique produces visually pleasant images, it offers not the same flexibility provided by polygonal meshes.

Image segmentation. Can be performed robustly by Active Contours like Level Sets where the surface of an deformable object evolves over time [2]. For the segmentation task a speed function for the evolution of the deformable model has to be defined, such that the region of interest is conquered before non interesting regions are conquered. In naive implementations this evolution process might produce a high genus object with many holes. Since our re-meshing approach is based on the topology of the vessel structure, we want our segmentation to represent this topology as realistically as possible. This is why we use a variant of the Level Set method which is able to control the topology of the deformable object [3].

Vascular structure surface reconstruction techniques can be differentiated into *model based* and *model free* approaches. Model based approaches usually assume that the vessels have a circular section at every point. While this simplification commonly leads to fast runtime performance it makes the reconstruction less accurate. Examples of model based methods are the ones presented by Felkel et al. [4] as well as Yim et al. [5], both of which produce quad dominant meshes.

In contrast, model free approaches attempt to reconstruct the original vascular structure's profile without simplifying assumptions, yielding more realistic results at the cost of a higher runtime. Many existing methods in this domain either produce triangle meshes (e.g. [6]) or are limited to certain structural configurations. For instance, the method introduced in [7] only works satisfactory on bifurcations (as opposed to general n-furcations).

In [8] Kälberer et al. extended the QuadCover approach to tube-like surfaces, where directional information is computed from the principal curvature of the input mesh, which can be sensitive to surface noise. In this paper we also focus on

the topologically correct extraction of the branching structure. Since the skeleton based segmentation approach is not sensitive to a noisy mesh surface, we can even deal with rather rough surfaces at a low resolution, which are commonplace when computing polygonal meshes from volumetric images.

Quad re-meshing Many approaches have been proposed to compute quad (dominant) meshes for triangular input meshes. To compute a quad mesh for an arbitrary input mesh Ray et al. [9] utilized two given orthogonal vector fields to compute a globally smooth parametrization from which the quad layout can be extracted. Similar to this early approach other techniques have been proposed which formulate the quad meshing problem as a global parametrization problem [10–13]. Once this parametrization is known, the basic idea is to map the canonical integer grid of the parameter domain back onto the surface in order to obtain a quad mesh.

Finding a parametrization that leads to a sensible, geometrically sound singularity structure, that produces quads aligned with the curvature of the surface and that exhibits low distortion is a challenging and time consuming global optimization problem. The solution to this problem for the mesh of the entire vascular structure would come at prohibitive runtime costs. By segmenting the vessel structures, we can efficiently compute a quad covered surface. At the same time the boundaries originating from the segmentation give us directional information for orienting the quads. For a recent survey on quad meshing see [14].

2 Mesh generation pipeline

The input to our algorithm is an angiogram, which is typically stored as a volumetric image. The image intensities are represented by a real valued function

$$f : \mathbb{R}^3 \to \mathbb{R}$$

mapping each point $\mathbf{x}$ in 3D space to an intensity value $f(\mathbf{x})$. We first identify the voxels belonging to the interior of the vascular structure by using a variant of the Level Set method, while an adapted Marching Cubes algorithm is applied to extract a triangular mesh the topology of which is a plausible interpretation of the topology of the vascular structure (Section 2.1).

For this mesh we generate a skeleton consisting of nodes along the vessels' center-lines connected by simple line segments. Since the skeleton naturally represents the topology of the vascular structure we can easily identify branches, endpoints and connecting tube segments and thus partition the triangular mesh according to those classifications (Section 2.2) and thereby split the hard global quad meshing task into easier local sub-tasks: for junctions and endpoints we employ the Mixed Integer Quadrangultion approach by Bommes et al. [12]. Since tube segments are topological cylinders, we can use a simpler meshing technique (Section 2.3).

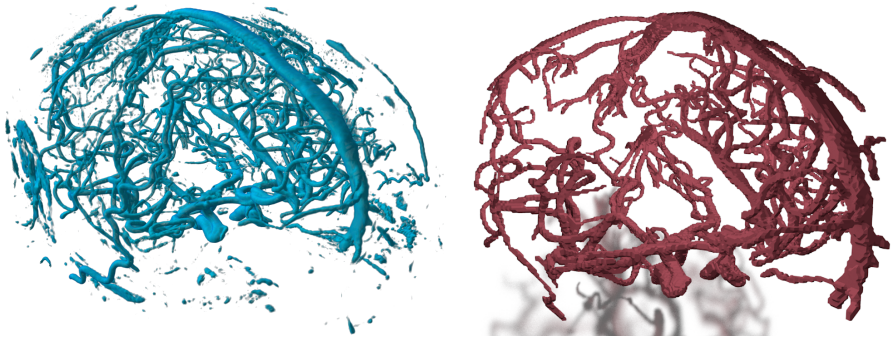


Fig. 1. Left: Direct volume rendering of the angiogram data. Right: Level Set volume at an early time of arrival. An adapted version of the Marching Cubes Algorithm extracts a noisy mesh with genus zero, which is the input to the subsequent stages of the re-meshing pipeline.

2.1 Separating the vascular structure from the background

Deformable models are well established for 2D and 3D image segmentation [15]. Such models are initialized in the interior of the region of interest, in our case the vascular structure. Forces, acting on the surface of the deformable model, change its shape over time. Interior forces preserve surface smoothness while exterior forces are applied such that the surface is dragged towards the border of the region of interest. Level Sets can be considered as the Eulerian point of view of this idea: to each specific point $\mathbf{x} \in \mathbb{R}^3$ a time of arrival $\eta(\mathbf{x})$ is assigned. A point $\mathbf{x}$ is said to lie on the surface if its time of arrival matches a certain value t, i.e. the surface is represented by all points $\mathbf{x}$ with $\eta(\mathbf{x}) = t$.

Although there are many fast implementations of this segmentation method [2], most of them are oblivious to topological changes of the evolving surface. If the surface touches itself, it is possible that additional tunnels or handles are produced. This actually changes the topology of the model which may lead to unnatural anatomical structures. In our setting this would mean that the algorithm generates, e.g., unwanted branches or connections in the vascular structure. Bischof et al. [3] suggested to tag edges between voxels as *cut edges* whenever the front touches itself. An adapted version of the Marching Cubes Algorithm [16] inserts additional faces at those *cut edges* to separate the two fronts at sub-voxel accuracy. Thus, topological changes of the model are effectively prevented and the resulting surface is guaranteed to be genus zero.

The evolution of the Level Set function is implemented using the Fast Marching Method [3, 2]. Voxels are classified into *conquered*, *front* and *free* voxels. At the beginning some voxels are marked as *conquered* and their time of arrival is set to $\eta = 0$. For this initialization step and as depicted in Figure 1, we visualize the vascular structure using direct volume rendering and allow the user

to click on a vessel. The picked pixel position together with the position of the eye defines a ray which can be traced through the volume to identify possible intersections with the visible surface. The midpoint between the first two intersections is likely to lie in the interior of a vessel and we use the surrounding voxels for the initialization.

In each iteration of the Fast Marching Method one *front* voxel $\mathbf{x}$ with smallest time of arrival $\eta(\mathbf{x})$ is marked as *conquered*. Then every *free* neighboring voxel $\mathbf{y}$ of $\mathbf{x}$ is marked as *front* and the time of arrival of all *front* neighbors are updated according to

$$\eta(\mathbf{y}) = \min\{\eta(\mathbf{y}), \eta(\mathbf{x}) + cost(\mathbf{x}, \mathbf{y})\}$$

The resulting segmentation is very sensitive to the design of the cost function. In our setting we define the cost function to be

$$cost(\mathbf{x}, \mathbf{y}) = (1 - f(\mathbf{y}))^2 + \nabla f(\mathbf{y}) \cdot \frac{(\mathbf{y} - \mathbf{x})}{\|(\mathbf{y} - \mathbf{x})\|}$$

Intuitively we allow the surface to grow fast into homogenous regions of high intensity, which is characteristic for vessels represented in angiogram data. If the conquering of a voxel (*free* $\rightarrow$ *front*) causes a topology change of the *front* surface, we mark the corresponding edge such that the resulting mesh preserves its genus zero topology (see [3] for details).

After all voxels have been conquered we use the Marching Cubes Algorithm to extract a triangular mesh representing the Level Set at some (user defined) early time of arrival. The resulting mesh is shown in the right image of Figure 1. The topology of this mesh is already close the topology of the vascular structure but it may still contain many geometric noise artifacts as it is common for meshes extracted with the Marching Cubes Algorithm. We use mesh smoothing [17] and an isotropic re-meshing technique [18] to generate a high quality triangle mesh which is passed on to the next steps of our pipeline.

2.2 Skeleton extraction and segmentation

In order to efficiently compute a quad dominant mesh given the previously computed triangular mesh $\mathcal{M}$, we segment this mesh into simpler elements, namely into *junctions* and *tubes*, and separately compute quad meshes for each part (we consider an endpoint as a special *junction* with only one neighbor). While in general the segmentation of a two manifold surface is a rather difficult task, the segmentation of an embedded skeleton naturally emerges from the branchings and neighborhood relations between its vertices.

We employ the skeleton extraction algorithm suggested in [19] to compute a graph structure $T = (V, E)$ encoding the branching behavior of the vascular structure. The set $V = \{1, \ldots, n\}$ are the indices of the skeleton vertices and the (undirected) edges $E = \{(i, j) \mid i \text{ is connected to } j\}$ encode the neighbor relations. Since the mesh $\mathcal{M}$ resulting from the previous step is genus zero, T is a tree (no cycles). The idea of the skeleton extraction is to iteratively perform

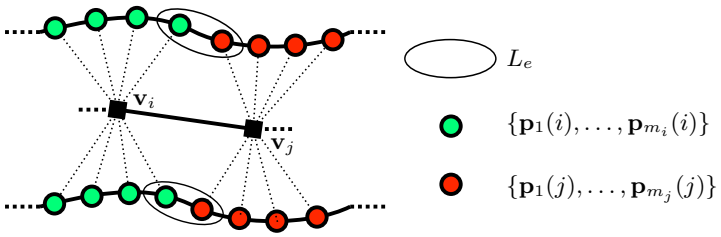


Fig. 2. Vertices approximately lying around the center of an edge are used to estimate the thickness of the structure.

a number of smoothing steps on the input mesh to shrink it nearly to a one dimensional structure. A sequence of edge collapses finally generates the tree T, where nodes are connected by line segments. As described in [19] the procedure stores a set of associated parent vertices $S_i = S(\mathbf{v}_i) = \{\mathbf{p}_1(i), \ldots, \mathbf{p}_{m_i}(i)\} \in \mathcal{M}$ for each skeleton node $\mathbf{v}_i$, which we use to estimate the radius $r(e)$ for an edge $e = (i, j)$ as the average orthogonal distance to the surrounding parent vertices:

$$r(e) = r(i,j) = \frac{1}{|L_e|} \sum_{\mathbf{p} \in L_e} \|(Id - \mathbf{d}\mathbf{d}^T)(\mathbf{p} - \mathbf{v}_i)\|$$

Here $\mathbf{d}$ is the normalized vector pointing from $\mathbf{v}_i$ to $\mathbf{v}_j$ and $L_e = \{\mathbf{p} \in S(\mathbf{v}_i) | \ \exists \mathbf{q} \in S(\mathbf{v}_j) : (\mathbf{p}, \mathbf{q}) \in \mathcal{M}\} \cup \{\mathbf{p} \in S(\mathbf{v}_j) | \ \exists \mathbf{q} \in S(\mathbf{v}_i) : (\mathbf{p}, \mathbf{q}) \in \mathcal{M}\}$ is the set of all parent vertices of $\mathbf{v}_i$ (or $\mathbf{v}_j$) who have neighbors in $\mathcal{M}$, that are parents of the node $\mathbf{v}_j$ (or $\mathbf{v}_i$), see Figure 2.

Given the tree T and the radii $r(e)$ the idea for the mesh segmentation is rather simple. We start a growing procedure from all junctions and walk over the tree nodes until we reach a point whose distance to the junction is larger than a multiple of the vessel radius (in our settings we use 1.5 times the maximal radius of the edges adjacent to the junction). With the edge direction at this point as normal vector, a plane is defined which cuts the mesh into two components.

Algorithm 1 shows the details of this procedure. Observe that this procedure merges simple *junctions* to more complex *junctions* by removing cuts whenever neighboring *junctions* lie close together. Each resulting cut plane intersects a set of edges of the original triangular mesh. By splitting those edges we get a 2D polygon lying in the respective plane. Cloning those new vertices finally splits the triangular mesh into two parts. We call the two versions of the cloned 2D polygon an interface between adjacent mesh segments. The final result of the segmentation is a set of meshes $\mathcal{M}_1, \ldots, \mathcal{M}_k$ where two adjacent mesh segments share a common interface. Thereby segments labeled as *junction* exhibit a number $\neq 2$ of interfaces to neighboring *tube* segments which have exactly two interfaces (Figure 5(b)).

Input: $T = (V, E)$
Output: a set of cuts $\{(e, \alpha)\} \subset E \times [0, 1)$
$S := \{\}$
for *each junction* $\mathbf{v}_j \in V$ **do**
  let $r = \max_{e=(j,k)\in E}\{r(e)\}$
  grow from $\mathbf{v}_j$ the connected subgraph $(V_J, E_J) \subseteq T$ with
    $\|\mathbf{v} - \mathbf{v}_j\| \leq 1.5r \quad \forall \mathbf{v} \in V_J$
  find the cut edges $C_J = ((i, k), \alpha) \subset E \times [0, 1)$ with
    $\mathbf{v}_i \in V_j$, $\mathbf{v}_k \notin V_j$, $\|(\mathbf{v}_i - \mathbf{v}_j) + \alpha\,(\mathbf{v}_k - \mathbf{v}_i)\| = 1.5r$
  add (V_J, E_J, C_J) to S // "collect junction segments"
end
while $\exists (V_1, E_1, C_1), (V_2, E_2, C_2) \in S$ *with* // *"merge overlapping"*
    $\exists (e, \alpha) \in C_1$ *so that* $e \in E_2$ *or*
    $\exists (e, \alpha) \in C_2$ *so that* $e \in E_1$ *or*
    $\exists ((i, j), \alpha) \in C_1, ((j, i), \beta) \in C_2$ *so that* $\alpha + \beta > 1$ **do**
  remove (V_1, E_1, C_1) and (V_2, E_2, C_2) from S
  merge (V_1, E_1, C_1) and (V_2, E_2, C_2) and add the result to S
end
return $\{C \mid \exists V, E : (V, E, C) \in S\}$

Algorithm 1: Skeleton Segmentation

2.3 Quad Dominant Re-Meshing

Junctions. The first step is to compute quad meshes for all *junctions*. For this we use the Mixed Integer Quadrangulation proposed by Bommes et al. [12]. We avoid the difficult search for good feature directions to align the quads and simply require the quads to align with the boundary defined by the interfaces. This automatically produces quads which are radially arranged around the interfaces of *tube* segments (cf. Figure 5(c)).

Tubes. Before we start re-meshing the *tubes* we resample the interfaces of a *tube* such that number and positions of the vertices lying on an interface match the number and positions of the vertices after quad meshing of the neighboring *junction* segments. Assume that after this step the left interface has n_l and the right interface has n_r vertices, where without loss of generality $n_l \leq n_r$.

Since the tube segments are topological cylinders there is a natural and simple way to generate quad meshes for them by intersecting a sequence of rings around the tube with a set of trajectories along the tubes. The only complication arises from the fact that the number of samples (= number of trajectories) at the left and right interface may be different since they are implied by the quad meshes that have been generated independently for the two neighboring junction segments.

Our procedure works in three steps. First, the rings are defined by equidistant iso-contours of a harmonic scalar field computed on the tube. Second, the number of samples is determined for each ring such that they coincide with the prescribed

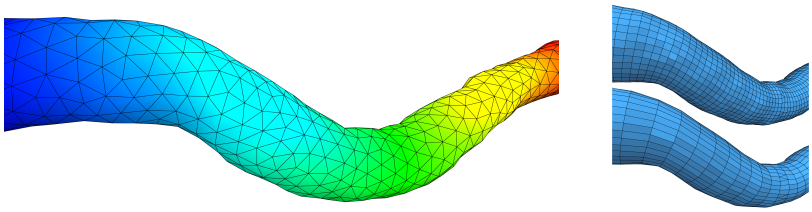


Fig. 3. Left: Harmonic field with boundary conditions zero (color coded as blue) at the left interface and one (color coded as red) at the right interface. Right: Since we can change the number of equally distributed rings, it is very easy to control the anisotropy of the quad elements. In the upper part we set the scaling factor to $s = 1$ while the lower part is an example for $s = 3$. This significantly reduces the total number of elements in the mesh.

interfaces on both sides and in between we place a number of transition rings where the number of samples per ring changes. In the third step we connect the samples of neighboring rings to form quad elements whenever possible. While this step is trivial for rings with the same number of samples, we have to introduce triangle elements between rings with differing numbers of samples. Ideally these triangles should be distributed uniformly around the circumference of the ring.

Ring generation. We compute a harmonic scalar function f_{harm} defined at the vertices of the *tube* segment by requiring

$$\Delta^2 f_{harm} = 0$$

where Δ is the cotangent weighted Laplace operator for triangle meshes [17]. As seen in the left part of Figure 3 we define proper boundary conditions by requiring the harmonic field to be zero at the left and one at the right interface. This allows us to extract non intersecting polylines (rings) by tracing the iso-contours of the harmonic function f_{harm}, on which we equally distribute a number of vertices which later become the new vertices of our quad dominant mesh.

Including both interfaces, the remesher generates

$$n_{iso} = \frac{l_{tube}}{s \cdot e_{target}} + 1$$

isolines, where l_{tube} is the length of the *tube*, e_{target} the target edge length and s a scale factor which controls the anisotropy of the quads. In general the minimal and maximal curvature on tubular structures differ severely, meaning that quads which are stretched in the direction of the tube axis will approximate the surface very well. Notice that one has no influence on the anisotropy of the elements when using one of the above mentioned global quad meshing approaches like the Mixed Integer Quadrangulation. Our re-meshing algorithm handles such anisotropy in a very intuitive way as illustrated in the right part of Figure 3.

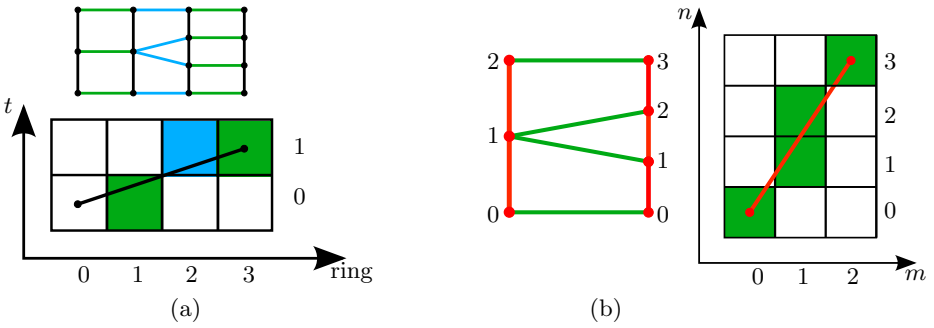


Fig. 4. (a) Identification of transition rings: The first step of the Bresenham Algorithm does not change y, so there will be no transition (green lines). In the second step we go north east, so a transition from $m = 3$ to $n = 4$ vertices occurs (blue lines). (b) Connection pattern between two rings with a different number of samples from $m = 3$ to $n = 4$ vertices.

In order to restrict the transitions where the number of samples per ring changes to as few rings as possible and to make these transitions as (rotationally) symmetric as possible we double the number of vertices with each transition. Hence, the total number of required transitions is equal to the smallest t for which

$$n_l \cdot 2^t \geq n_r$$

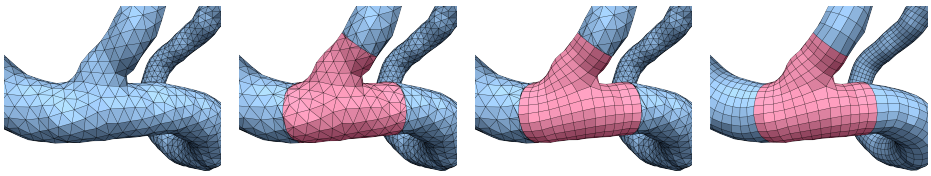
holds. At the last transition the number of vertices is not doubled but clamped to n_r to ensure compatibility with the right interface.

Connecting rings. For the generation of tube meshes we have to solve two tasks. One is the task to select the t transition rings (among the n_{iso} rings along the tube) and the other task is to connect the samples of neighboring rings (which is only non-trivial for transition rings). In both cases, we have to distribute a discrete number of events over a (larger) discrete number of elements. We apply the Bresenham line rasterization algorithm [20] for both tasks. Notice that the Bresenham algorithm essentially distributes a discrete number of y-steps over a (larger) discrete number of x-steps.

If we want to equally distribute the t transition rings among the n_{iso} rings along a tube, we can rasterize a line on a pixel grid from $(0, 0)$ to (n_{iso}, t) and pick those rings for which the y-coordinate changes (cf. Fig. 4(a)).

Similarly, if we want to connect two rings with n and m samples ($m < n$) respectively, we can rasterize a line from $(0, 0)$ to (m, n) and connect the ith sample on the first ring with the jth sample on the second ring if the pixel (i, j) was set by the Bresenham algorithm (cf. Fig. 4(b)).

Geometric Optimization. After placing and connecting all vertices we allow the vertices lying at the n_{iso} rings to move around the rings while keeping their relative geodesic distance fixed. For each ring we find that rotation (parametrized



(a) (b) (c) (d)

Fig. 5. (a) The input to our remeshing stage is a smoothed and re-sampled version of the mesh extracted using the Marching Cubes Algorithm. (b) The mesh is partitioned into *junction* and *tube* components. The cuts insert additional vertices at the respective interfaces. (c) We re-mesh the *junctions* yielding a fair quad layout. (d) Tubes are re-meshed, such that the final mesh is a quad dominant mesh.

by one scalar degree of freedom) which yields a configuration where the angles within all quads are as close as possible to 90°. The left and right interface rings are not rotated to preserve compatibility (Figure 5(d)).

3 Results

We ran our experiments on an Intel(R) Core(TM) i7 CPU with 2.67GHz. The angiogram data set has a resolution of 288 x 384 x 185 voxels and it took less than 2 minutes to extract the mesh representing the vascular structure using the Level Set method. The smoothed and remeshed version of this mesh, which serves as input for the quad re-meshing has a total number of 110k vertices.

The skeleton of this mesh was extracted in less than four minutes, while segmenting the resulting tree into 215 *junction* and 214 *tube* elements was nearly instantaneous. Computing the quad layouts for each of the *junctions* was the most time consuming task and took approximately 16 minutes. In comparison to this the proposed algorithm to re-mesh the tubes is highly efficient and took less than 4 seconds. The total time to perform the re-meshing is with approximately 20 minutes much more efficient than solving the global optimization problem, which took more than 12 hours to compute due to the complex layout structure. When we place anisotropic quads with a stretch factor of $s = 3$ onto the *tubes*, the resulting mesh has 120k faces, while the isotropic version counts 200k faces (67% overhead) with the same approximation quality. Exemplary close-ups are shown in Figure 6.

4 Conclusion

We presented a pipeline to generate quad dominant meshes from volumetric images. By controlling the topology during the Level Set segmentation such that the topology of the final polygonal mesh comes close to the topology of the scanned vascular structure, we are able to compute the skeleton of the vascular

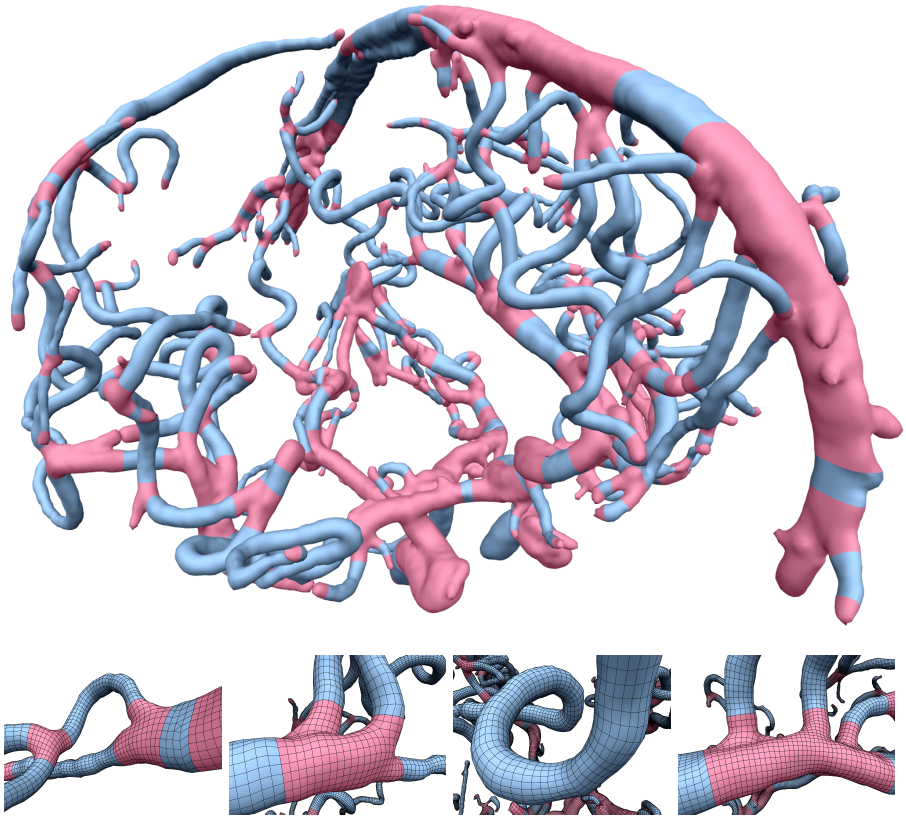


Fig. 6. Results produced with our re-meshing pipeline.

structure. Segmenting the skeleton into *junctions* and *tubes* effectively divides the difficult global quad meshing task into a set of more simpler sub-tasks where we treat *tubes* like cylindrical components. Irregular elements lying on *tubes* are distributed over the structure using a procedure inspired by the Bresenham-Algorithm for lines. The uniform distribution of rings using a harmonic field effectively allows us to control the anisotropy of quad elements, which is not easily possible for global quad meshing approaches.

In future work we would like to extend this idea to produce pure quad-meshes. Since this can only be achieved if the number of vertices at the left and the right interface differ by an even number of samples, one would need to include additional global constraints for the number of vertices at the interfaces of the *junction* elements.

References

1. Hadwiger, M., Ljung, P., Salama, C.R., Ropinski, T.: Advanced illumination techniques for gpu-based volume raycasting. In: ACM SIGGRAPH 2009 Courses. SIGGRAPH '09, New York, NY, USA, ACM (2009) 2:1–2:166
2. SETHIAN, J.A.: Level Set Methods and Fast Marching Methods/ J. A. Sethian. Cambridge University Press, Cambridge (1999)
3. Bischoff, S., Kobbelt, L.: Sub-Voxel topology control for Level-Set surfaces. Computer Graphics Forum **22**(3) (September 2003) 273–280
4. Felkel, P., Wegenkittl, R., Buhler, K.: Surface models of tube trees. In: Computer Graphics International. (2004) 70 –77
5. Yim, P., Cebral, J., Mullick, R., Marcos, H., Choyke, P.: Vessel surface reconstruction with a tubular deformable model. Medical Imaging **20**(12) (2001)
6. Schumann, C., Neugebauer, M., Bade, R., Preim, B., Peitgen, H.O.: Implicit vessel surface reconstruction for visualization and cfd simulation. International Journal of Computer Assisted Radiology and Surgery **2** (2008) 275–286
7. Antiga, L., Ene-Iordache, B., Caverni, L., Cornalba, G.P., Remuzzi, A.: Geometric reconstruction for computational mesh generation of arterial bifurcations from ct angiography. Computerized Medical Imaging and Graphics **26**(4) (2002) 227 – 235
8. Kälberer, F., Nieser, M., Polthier, K.: Stripe parameterization of tubular surfaces. In: Topological Methods in Data Analysis and Visualization. Theory, Algorithms, and Applications. Mathematics and Visualization. Springer Verlag (2010)
9. Ray, N., Li, W.C., Lévy, B., Sheffer, A., Alliez, P.: Periodic global parameterization. ACM Trans. Graph. **25**(4) (2006) 1460–1485
10. Kälberer, F., Nieser, M., Polthier, K.: Quadcover - surface parameterization using branched coverings. Computer Graphics Forum **26**(3) (2007) 375–384
11. Dong, S., Bremer, P.T., Garland, M., Pascucci, V., Hart, J.C.: Spectral surface quadrangulation. SIGGRAPH '06, New York, NY, USA, ACM (2006) 1057–1066
12. Bommes, D., Zimmer, H., Kobbelt, L.: Mixed-integer quadrangulation. ACM Trans. Graph. **28**(3) (2009) 1–10
13. Zhang, M., Huang, J., Liu, X., Bao, H.: A wave-based anisotropic quadrangulation method. SIGGRAPH '10, New York, NY, USA, ACM (2010) 118:1–118:8
14. Bommes, D., Lévy, B., Pietroni, N., Puppo, E., a, C.S., Tarini, M., Zorin, D.: State of the art in quad meshing. In: Eurographics STARS. (2012)
15. Mcinerney, T., Terzopoulos, D.: Deformable models in medical image analysis: A survey. Medical Image Analysis **1** (1996) 91–108
16. Lorensen, W.E., Cline, H.E.: Marching cubes: A high resolution 3d surface construction algorithm. COMPUTER GRAPHICS **21**(4) (1987) 163–169
17. Pinkall, U., Polthier, K.: Computing discrete minimal surfaces and their conjugates. Experimental Mathematics **2** (1993) 15–36
18. Botsch, M., Kobbelt, L.: A remeshing approach to multiresolution modeling. In: Proc. SGP. (2004) 185–192
19. Au, O.K.C., Tai, C.L., Chu, H.K., Cohen-Or, D., Lee, T.Y.: Skeleton extraction by mesh contraction. ACM Trans. Graph. **27**(3) (2008)