Modeling and Visualization of Cardiovascular Systems

Thomas Wischgoll[1]

1 **Computer Science and Engineering Department, Wright State University**
thomas.wischgoll@wright.edu

Abstract

Modeling complex organs, such as the human heart, requires a detailed understanding of the geometric and mechanical properties of that organ. Similarly, the model is only as accurate as the precision of the underlying properties allow. Hence, it is of great importance that accurate measurements of the geometric configuration are available. This paper describes the different steps that are necessary for creating and visualizing such a vascular model, ranging from determining a basic geometric model, gathering statistical data necessary to extend an existing model up to the visualization of the resulting large-scale vascular models.

1998 ACM Subject Classification I.3.8 Applications, J.3 Life and Medical Sciences

Keywords and phrases Volumetric Data, Curve-skeleton, Cardiovascular Structure

Digital Object Identifier 10.4230/DFU.SciViz.2010.210

1 Introduction

In order to precisely model a complex organ, such as the human heart, the organ's mechanical and physiological properties need to be fully understood. This includes the analysis of, for example, the geometric configuration of the myocardium and its supporting vasculature as well as the structural configuration of the vessels themselves. Therefore, this paper describes methodologies that help analyze such properties and provides some answers to the visual analytics challenges this ongoing research is facing.

In order to extract morphometric data from a volumetric data set that resembles a scan of a coronary system with the arteries highlighted using some form of contrast agent, the center lines of the individual vessel segments need to be identified resulting in a curve-skeleton describing the medial axis of the vessels. This then allows the software to compute the vessel radii as the distance between the center lines and the vessel boundary. Similarly, the length of a vessel segment can be computed as the length of the center lines, as well as bifurcation angles as the different angles formed by the center lines at a bifurcation. Obviously, it is quite important that both the boundary and the center lines are determined as accurately as possible. The method used in this paper computes both at a sub-voxel level to achieve a very high precision.

Curve-skeletons represent the very basic features of an object. They describe a thinned version of the object represented as some type of stick model resulting in the center-lines of the object. Therefore, the use of curve-skeletons can prove useful for applications, such as animation [46] or flight planning for virtual colonoscopy [19]. Similarly, accurate curve-skeleton methods can be used for extracting quantitative measurements from computed tomography (CT) scanned vascular structures. Here, the curve-skeleton describes the center lines of the vessels. These can then be used to measure vessel radius, vessel lengths, and angles between vessels within the volumetric data set retrieved by using the CT scanner. In order to derive these measurements from the volumetric data set, an accurate extraction method for curve-skeletons is desirable. For example, thinning-based techniques that work

Scientific Visualization: Advanced Concepts.
Editor: Hans Hagen; pp. 210–226

Dagstuhl Publishing
Schloss Dagstuhl – Leibniz Center for Informatics (Germany)

in the voxel space of the volumetric data set tend to generate jagged lines which are in no way suitable for determining angles between vessels. Similarly, inaccuracies can occur when computing the radii of the vessels. Hence, an approach that only uses the volumetric data set in order to identify the boundary surface of the contained object is more promising.

The algorithm used in this paper is exactly of this type. It is capable of extracting the boundary surface of an object that is defined by a volumetric data set at sub-voxel level. For this, it determines the location of the maximal gradient within the volumetric data set similar to Canny's [8] maxima-suppression technique but extended to three dimensions. Since the algorithm only relies on points extracted from the volumetric data set but not on its underlying structured grid, it can also be applied to objects defined by a point set without any restrictions.

Techniques used for computing the topological graph of a vector field are applied to determine the curve-skeleton. First, for all points on the object's boundary vectors are computed that are orthogonal to the boundary surface. There are different options for computing these vectors. They either can be derived by determining the normal vector of a plane that is defined by a least-square fit of the point and its neighbors. Or – in case of the object being defined by a volumetric data set – the image gradients determined in the previous step can be used. In both cases, the normal vectors can be determined in such a way that they are facing inwards with respect to the object. The entire vector field can then be determined by computing a tetrahedrization of the entire point cloud and then linearly interpolating within the tetrahedra. In order to ensure that only the curve-skeleton inside the object is extracted, all tetrahedra that are located outside the object are removed based on the normal vectors.

A topological analysis of the vector field within the faces of every tetrahedron yields points on the curve-skeleton. By following the topology of the tetrahedrization, points on the curve-skeleton within neighboring tetrahedra can be connected resulting in the entire curve-skeleton. Based on this curve-skeleton, vessel radii are computed at a very high accuracy. Similarly, bifurcation angles can be computed and statistically analyzed. This information, i.e. statistical information about the vessel lengths, vessel radii, and bifurcation angles for vessels of different sizes, can then be used to grow additional vessels onto an existing model resulting in a large-scale model of the vasculature that includes vessels down to the capillary level.

Rendering such a large-scale model is quite challenging for currently available computing hardware since commodity graphics cards are presently not able to display this amount of information interactively. For the complete coronary arterial model, traditional rendering algorithms require the total of 6 GB of geometric information to be transferred from main memory to the graphics card, which presents a limit for interactive rendering. Furthermore, most desktop computers are not capable of handling this amount of data due to insufficient main memory. Hence, the size of such a large-scale anatomical model is prohibitive for rendering on desktop computers without employing out-of-core techniques or more sophisticated rendering algorithms. Occlusion culling techniques are usually capable of achieving better performance. However, when applied to tree-shaped data sets only little occlusion occurs which in turn requires the removal of partly visible areas of the data set in order to achieve an increase in rendering performance [49]. In addition, medical personnel and researchers tend to prefer to see the entire data set without anything being removed in fear of missing essential information. Hence, the visualization methods in this paper refrained from applying any type of geometry reduction methods but improved on the rendering method itself instead.

The objective of this paper is to outline methods that allow for the analysis, modeling, and visualization of cardiovascular structures. This includes methodologies that are capable of extracting quantitative morphometric measurements from scans of cardiovascular data sets. These statistical data can then be used to generate a large-scale vascular model with vessels from the most proximal level down to the capillaries. The enormous amount of vessel segments included in such a model requires novel visualization methods in order to achieve interactive rendering of these data sets. The

proposed approach computes the necessary geometry that is required for visualizing the data set on-the-fly and utilizes all computational resources available on today's desktop computers, i.e. all cores of the CPU and the GPU at the same time, thereby achieving maximal performance of the visualization algorithm. The rendering algorithm is especially optimized for rendering large-scale tree-shaped data sets and the computation of all necessary geometry in parallel on the CPU assisted by the GPU. The techniques described in this paper can easily be applied to data extracted from any type of tree-like structure.

The following section discusses work related to the topics described in this paper. A description of the algorithms can be found in section 3. Section 5 discusses the performance of the algorithm applied to the large-scale vascular data set, followed by conclusions and future work.

2 Related Work

Several approaches for extracting curve-skeletons or medial axes can be found in the literature. A very good overview of available techniques can be found in the paper by Cornea et al. [9].

Some methods start with all voxels of a volumetric data set and use a thinning technique to shrink down the object to a single line. Directional thinning approaches use a specific order in which voxels are removed. For example, directions, such as up or down, are used to define this order and conditions are used to identify endpoints [2, 7, 17, 22, 24, 31, 32, 37, 45]. Since these methods are sensitive with respect to the order in which the voxels are removed the resulting curve-skeleton may not be perfectly centered. Non-directional methods [5, 42] or fully parallel approaches [12, 25, 27] do not suffer from this disadvantage. Ideally, the topology of the object should be observed. Such an approach was proposed by Lobregt et al. [23] which is the basic technique used in commercial software systems, such as Analyze^{TM} developed by the Mayo Clinic. The disadvantage of this approach is that it tends to produce jagged lines which do not allow accurate measurements of angles between parts of the object, such as individual vessels of a vascular structure. Other approaches [43] classify the voxels in different groups, such as edge, inner, curve, or junction and re-classify after removal of a voxel. A similar algorithm is proposed by Palagyi et al. [31]. The disadvantage of thinning algorithms is that they can only be applied to volumetric data sets due to the nature of these algorithms.

To avoid this disadvantage, other approaches deploy the distance transform [16] or distance field in order to obtain a curve-skeleton. For each point inside the object, the smallest distance to the boundary surface is determined. For this, the Euclidean metric or the <3,4,5> metric [4] can be used. Also, fast marching methods [39, 44] can be deployed to compute the distance field. Voxels representing the center lines of the object are identified by finding ridges in the distance field. The resulting candidates must then be pruned first. The resulting values are then connected using a path connection or minimum span tree algorithm [41, 47, 52]. Methods used to identify points on the ridges include distance thinning [10, 14, 15, 34], divergence computing [6], gradient searching [3], thresholding the bisector angle [26], geodesic front propagation [33], or shrinking the surface along the gradient of the distance field [38]. The distance field can also be combined with a distance-from-source field to compute a skeleton [53]. Based on an anisotropic diffusion applied to the image gradients, Yu et al [51] extract skeletons from 2D images.

Techniques based on Voronoi diagrams [1, 11] 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 a curve-skeleton. Voronoi-based methods can be applied to volumetric data sets as well as point sets. Due to the fact that clustering of the resulting points is required, these approaches can lack some accuracy.

The center lines in combination with the vessel radii computed at the center points allow for a geometric reconstruction of the vasculature. Various techniques for visualizing vascular structures

can be found in the literature. Hahn et al. [18] employ geometrical primitives, such as truncated cones, to visualize vessels inside the human liver. Masutani et al. [28] used cylinders aligned to the vessel skeleton to visualize the vasculature. Different radii at branchings resulted in discontinuities when using this method. Felkel et al. [13] reconstructed liver vessels from center line and radius information to supply an augmented reality tool for surgery. Puig et al. [35] developed a system for exploring cerebral blood vessels using a symbolic model with a focus on geometric continuity and on realistic shading. Oeltze et al. [29, 30] use convolution surfaces to obtain a smoother representation of blood vessels extracted from CT or MR data. Ritter et al. [36] extended on illustrative rendering techniques to accentuate spatial depth by use of GPU-accelerated shadow-like depth indicators. The method was applied to vascular structures to better distinguish vessels in the front from vessels further in the back. Stoll et al. [40] introduced stylized primitives which utilize the GPU to render cylindrical objects using a single quadrilateral. The approach was used for fast rendering of streamlines in vector field data sets.

3 Methodology

As indicated in the Physiome Project [20], modeling the human body requires its analysis at various levels ranging from the genomic over to the tissue level and organ level up to the human body itself. Similarly, modeling the human heart requires a detailed understanding of the vascular structure and the individual vessels of which the structure composed.

A previously developed software package extracts morphometric data from a volumetric image in several steps. Although a brief summary of the algorithm is given here, a detailed description can be found in the original publication [48]. The algorithm first segments the tubular objects within the volumetric image based on the image gradients. In order to get a more accurate representation of the boundary, the points resulting from the segmentation step are moved along the gradient direction in such a way that they are located at the maximal gradient. This provides a more precise and smoother representation of the boundary compared to using the original voxel locations. Then, a vector field is computed in such a way that all vectors are pointing inwards to the center of the tubular object. In the simplest case, the image gradients can be used at the boundary. Using a tri-linear interpolation, the vector field can be computed after a tetrahedrization of all the boundary points is determined. Finally, the points on the center lines are computed using a topological analysis of the vector field within the cross sectional area of the tubular objects and connected based on the topology of the tetrahedrization. This then results in a precise representation of the center lines of all tubular objects within the volumetric image. At the same time, the algorithm computes the radii of the tubular shapes along with the center lines as the distance between center line at boundary.

The algorithm has a proven accuracy with smaller error than most other methods. For the validation of the algorithm, vessel radii were computed for the main trunk of the arterial branches of a series of five porcine heart data sets as the distance between the center line and the vessel boundary and then compared to manual optical measurements. The agreement between the measurements is very good, with an error of $0.06mm$ (scan resolution was $0.6mm \times 0.6mm \times 1.0mm$), which underlines the accuracy of the center lines. For the three major branches (LAD, LCx, and RCA) of the five porcine hearts, the root mean square error between the two measurements is $0.16mm$ and the average deviation is $0.13mm$.

Based on these accurate morphpometric data, the vascular geometry can be reconstructed. The morphometric data extracted from the volumetric data set provides vessel segments identified as line strips defined by the center lines of the vessels with radii measurements at both ends of each line segment. Conic cylinders can then be computed that represent each vessel segment. Since two consecutive line segments may not have the exact same direction, the end caps of the conic cylinders

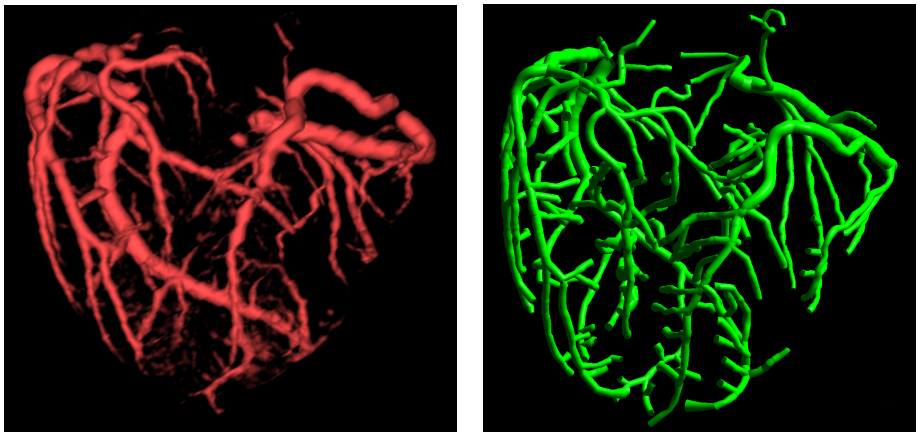

Figure 1 Vasculature of a pig heart: volume rendering (a) and geometric reconstruction (b) of the same heart.

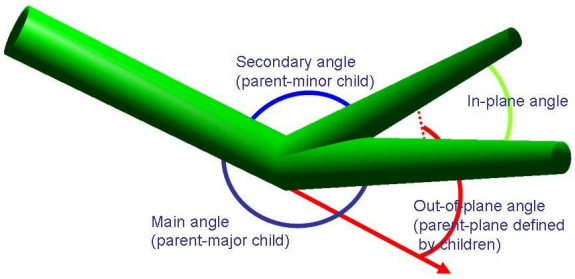

Figure 2 Definition of bifurcation angles.

are rotated in such a way that they describe half the angle between the line segments. This way, a smooth transition between segments can be created that avoids gaps at the transition. By computing such a conic cylinder for every vessel segment that was extracted from the volumetric data set, a geometric reconstruction of the vasculature is achieved. Figure 1 depicts such a reconstruction and compares it to a volume rendering of the exact same volumetric data set.

Using the above algorithm, the arterial center lines were determined for seven porcine hearts and for all vessels detected by the software. Since the accuracy for vessels smaller than the scan resolution is questionable as indicated by our previous study, vessels with a diameter of less than $1mm$ were not included in the analysis. Based on these center lines of the vessel segments, bifurcation angles were measured as defined in Figure 2. The two child segments define a plane. The angle between the two children within that plane describes the in-plane angle. The out-of-plane angle is formed between the parent segment and the plane defined by the two child segments. The main angle refers to the angle between the parent and the larger child segment while the secondary angle is determined by the angle between the parent and the smaller child segment.

Due to the fact that the first segment of a center line merely represents the connection of the vessel

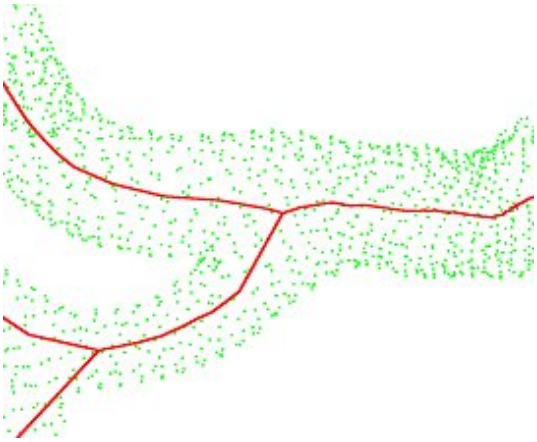

Figure 3 Sample bifurcation extracted from the volumetric data.

branches but not necessarily the direction of the daughter vessel due to possible curvature, more than just the first segment were included in the calculation of the bifurcation angle (Figure 3). The first center line segment usually only leads out of the main vessel formed by the parent and the larger child. Using this segment alone would result in erroneous bifurcation angles. Hence, the center line segments starting with the first segment after the bifurcation until the length of the daughter segment reaches three times the radius of the parent vessel were considered. The vectors representing the center line segments identified in such a way were then averaged and weighed by the length of each vector to determine a representative for the orientation of the daughter vessels. These were used for the computation of the bifurcation angles as described before.

For the statistical analysis, the bifurcations were grouped with respect to the order of the respective vessel segments. These groups were defined by the diameter of the vessels thereby associating a range of diameter values to a specific order. The relation between vessel diameters and orders are shown in Figure 4 for LAD, LCx, and RCA. The bifurcations were then classified based on the order of the parent and daughter vessels. Bifurcations between vessel orders ranging from 9 to 11 for LAD and RCA, and 9 and 10 for LCx were considered since vessels of that specific size can be extracted reliably from CT scanned imagery with a resolution of $1mm$. Average angles and standard deviation within each of these groups were computed for each of the major coronary arterial branches.

Figures 5 through 8 list the statistical summary of the secondary angles and out-of-plane angles in matrix form according to the order number of the parent vessel segment (vertical axis) and the daughter vessel segment (horizontal axis). Each entry lists the average angle within that group and the standard deviation. The tables confirm assumptions typically made for vascular structures. The smaller daughter vessel usually forks off at a relatively larger angle whereas the larger vessel continues the vessel segment in a rather straight fashion. The low out-of-plane angles confirm that bifurcations typically are relatively planar. Despite the important fact of confirming observations and assumptions often found in the literature, it is often difficult to analyze the values for specific bifurcations. A focus+context-oriented visualization can help in this case. By annotating a visualization of the vascular geometry with textual information representing the angular values, it is much easier to correlate the values with the geometry of the vascular structure making an analysis significantly less cumbersome. Figure 9 shows an example for such a visualization. The user can zoom in on specific bifurcations and check the attached angular values interactively.

The size of the vessel segments that can be extracted from the volumetric data depends on the resolution of the scanning device used. Typically, capillary vessels have a diameter of $6-7\mu m$. This is significantly smaller than the resolution of typical scanners. In order to derive a detailed model

		range	
RCA	Order	low	high
	9	552	955
	10	955.1	2,319
	11	2,319.1	3,606

		range	
LAD	Order	low	high
	9	554	986
	10	986.1	2,189
	11	2,189.1	3,830

		range	
LCx	Order	low	high
	9	649	1,782
	10	1782.1	2,940

Figure 4 Relation between order numbers and vessel diameter.

RCA	order 9	order 10	order 11
order 9	29.66±13.93	19.79±11.17	12.10±0.64
order 10		24.72±20.69	15.73±7.79
order 11			17.31±1.06

LAD	order 9	order 10	order 11
order 9	25.36±15.88	23.26±13.33	15.41±9.48
order 10		26.94±20.66	23.28±17.93
order 11			35.00±20.65

LCx	order 8	order 9	order 10
order 9		20.05±15.00	17.35±16.95
rder 10			24.74±30.33

Figure 5 Main bifurcation angles.

RCA	order 9	order 10	order 11
order 9	73.13±28.13	56.99±26.18	60.71±13.14
order 10		63.87±34.63	57.55±22.56
order 11			36.31±21.93

LAD	order 9	order 10	order 11
order 9	66.19±31.75	70.87±28.62	64.89±19.56
order 10		66.85±32.95	65.02±26.20
order 11			51.36±6.31

LCx	order 8	order 9	order 11
order 9		65.75±31.95	69.46±19.11
order 10			86.75±60.92

Figure 6 Secondary bifurcation angles.

RCA	order 9	order 10	order 11
order 9	89.46±19.12	65.18±27.89	70.35±11.46
order 10		68.80±35.15	62.30±20.31
order 11			38.79±43.47
LAD	order 9	order 10	order 11
order 9	66.37±31.68	76.55±32.03	69.25±16.86
order 10		68.57±33.86	75.29±31.87
order 11			25.49±26.03
LCx	order 8	order 9	order 10
order 9		71.99±32.03	64.39±26.57
order 10			74.08±48.50

Figure 7 In-plane bifurcation angles.

RCA	order 9	order 10	order 11
order 9	12.81±11.26	10.97±10.01	5.82±3.95
order 10		11.47±10.62	7.02±4.05
order 11			9.32±8.65
LAD	order 9	order 10	order 11
order 9	16.55±13.52	12.26±8.21	10.78±9.74
order 10		12.92±13.46	12.09±10.11
order 11			21.27±1.24
LCx	order 8	order 9	order 10
order 9		9.58±10.13	6.74±7.01
order 10			6.61±4.04

Figure 8 Out-of-plane bifurcation angles.

of the vasculature that includes vessels on the capillary level, the vasculature extracted from the volumetric data can be extended based on statistical properties of the vasculature. Recently, Kaimovitz et al. [21] developed a three-dimensional (3-D) geometric model of the entire coronary arterial tree (right coronary artery, RCA; left anterior descending artery, LAD; and left circumflex, LCx arterial tree). The model is purely based on morphometric data as extracted using the methodology as described previously, i.e. statistical information of vessel length, vessel diameter, and bifurcation angles. The model spans the entire coronary arterial tree down to the capillary vessels in a prolate spheroid model of the heart and encompasses about 11 million segments. The 3-D tree structure was reconstructed initially in rectangular slab geometry by means of global geometrical optimization using a parallel Simulated Annealing (SA) algorithm. The SA optimization was subject to a global boundary avoidance constraint and local constraints at bifurcations prescribed by previously measured data on branching asymmetry in the coronary arterial tree. Subsequently, the reconstructed tree was mapped onto the prolate spheroidal geometry of the heart. The transformation was made through least squares minimization of the deformation in segment lengths as well as their angular characteristics.

Due to the high number of vessel segments, the geometric data that needs to be generated for creating a high-quality visualization of this data set is quite substantial. When discretizing the end caps of the conic cylinders to represent each vessel segments with 8 vertices, the overall geometric data

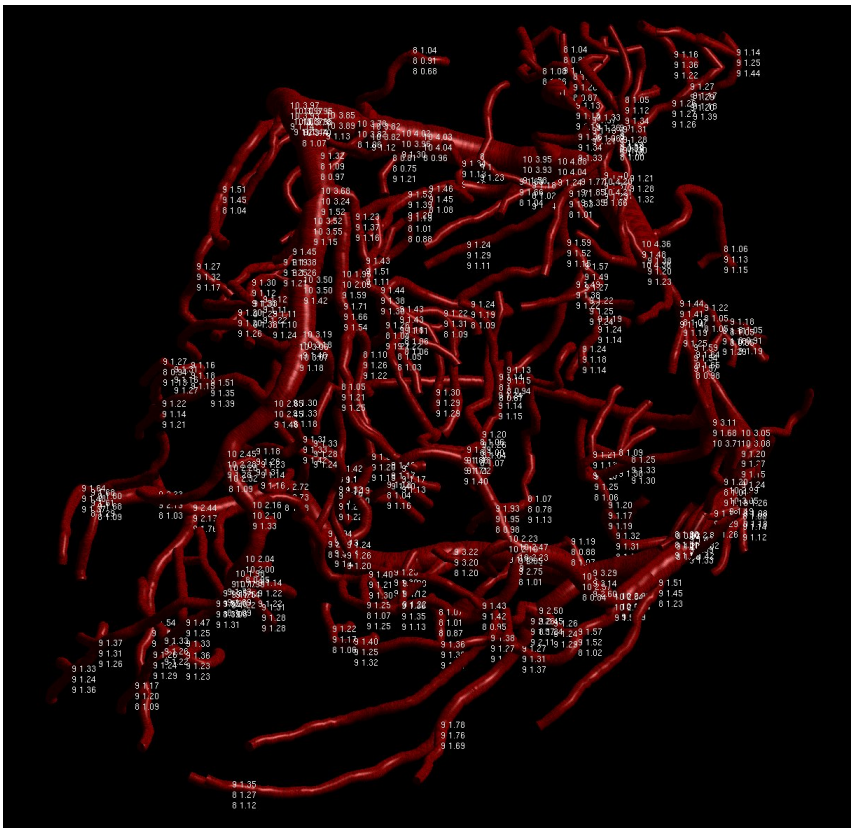

Figure 9 Focus+context visualization of vascular geometry and bifurcation angles.

amounts to 6 GBs of data. Since this amount of data exceeds the amount of memory typically available in desktop computers, out-of-core rendering approaches are required to visualize the data [50]. The entire geometric information is stored in a file and then accessed using memory-mapping so that the operating system automatically swaps chunks of the data set in and out as necessary. In addition to the geometry, a pressure simulation can be performed based on the geometric configuration of the vasculature. The resulting pressure data can then be used to color-code the vessels. Figure 10 shows such an image where the vessel segments are colored from red (low pressure) to green (high pressure).

Even though the rendering is of very high-quality, the fact that out-of-core methods are required results in rather slow rendering performance of about one minute per frame. This obviously is not interactive. However, the center line information combined with the vessel radii as described by the vascular model requires comparably less memory and fits into most desktop computer's main memory. Hence, a visualization technique that is solely based on these center lines and radii only would not require out-of-core techniques and therefore could be able to achieve faster rendering times.

The rendering technique described in this paper follows a similar approach than Stoll et al. [40] where the GPU is used to make lines appear cylindrical for rendering streamlines. In this paper, this idea is extended to allow for more general conic cylinders with different radii at each end since this is a requirement for rendering the vessel segments. In addition, the rendering technique is optimized for the purpose of rendering large-scale tree-shaped data sets and parallelized to run on the multiple cores of a CPU at the same time.

In this method, each vessel segment is represented only by a single quadrilateral. This approach provides two advantages. First, the quadrilateral can be solely computed based on the center line

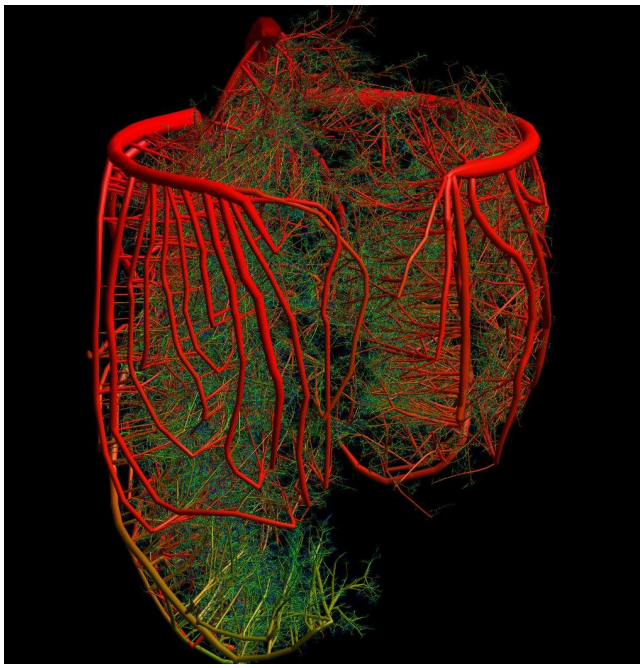

Figure 10 High-quality rendering of the arterial tree of a pig heart including vessels from the most-proximal level down to the capillaries.

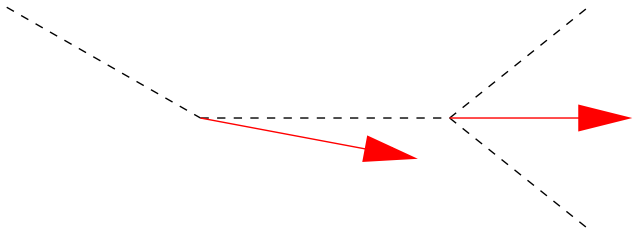

Figure 11 Computation of vessel directions (red arrows attached to its center point) based on the center lines of the vessels (dashed).

and radii information describing the vessel segments. Second, the rendering algorithm draws only four vertices per vessel segment compared to at least 18 vertices for the traditional approach. A fragment program can be used to make the quadrilateral appear like a cylindrical object. Of course, this rendering technique requires that the quadrilateral is always parallel to the projection plane in order to be completely visible. Only then a cylindrical appearance can be achieved. Consequently, the quadrilateral needs to be recomputed for every frame if the view has changed.

In a pre-computational step, the vessel directions at each center point are computed as the average of the direction of vessel segment ending at that center point and the direction of the vessel segment starting at that center point. In case of bifurcations, there may be more than one vessel segment starting at that center point. Consequently there are more than one direction available as well which then are included in the computation of the average direction. Figure 11 illustrates two examples. The resulting vessel direction is necessary to ensure a smooth transition between the vessel segments.

In order to compute the quadrilateral to represent a vessel segment, an offset vector is computed for every center point in such a way that this offset vector is parallel to the projection plane and orthogonal to the vessel direction at that center point. Figure 12 illustrates this procedure. Based on

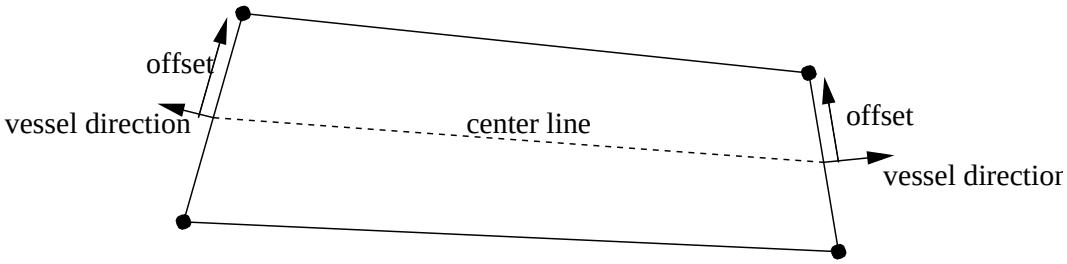

Figure 12 Computation of the quadrilateral representing a single vessel segment.

the offset vectors of two consecutive center points, four vertices can be computed by offsetting the center point in positive and negative direction of the offset vector as described in figure 12. This then results in the quadrilateral representing that vessel segment. Since neighboring quadrilaterals are based on the same vessel direction and therefore on the same offset vector, a smooth transition from one vessel segment to the next is guaranteed. Due to the fact that the offset vectors are parallel to the projection plane, it is guaranteed that the quadrilateral itself is parallel to the projection plane as well. This quadrilateral is then used as an approximation for the vessel segment.

Obviously, representing the vessel segment using a single quadrilateral works best when looking at the segment from the side. When looking at the vessel segment's cross-sectional area, it is approximated as a single vertical line since only a single quadrilateral is used as a representation. However, this is more obvious only for terminal vessels, i.e. vessel segments that represent leaves in the vascular tree. For vessels with adjacent segments most of the resulting artifact will be covered by that adjacent vessel segment. In addition, the percentage of vessels that are orthogonal to the projection plane and where the cross-sectional area would be visible is very small among all 11 million vessel segments. It would be easy to test for those cases and draw a circular area instead of a quadrilateral. However, this would require additional computation for the test and drawing more vertices for approximating a circle compared to drawing a quadrilateral. The test would have to be performed for every vessel segment. Considering the minimal artifact resulting from neglecting these cases and the loss of performance from the additional testing, this was not implemented to achieve the maximal performance possible.

In order to find the most efficient way of computing the quadrilaterals representing the vessel segments, three different approaches were implemented and compared. The first one computes all offset vectors for two consecutive center points individually. Hence, the offset vector for two consecutive quadrilaterals are computed twice. To avoid this double computation, two other approaches were tested. The first one stored all offset vectors in memory. This resulted in a huge array of offset vectors so that eventually the amount of available main memory was exceeded and the system started swapping during the rendering. Consequently, the performance of the rendering algorithm dropped. The second approach recursively traversed the vascular tree structure so that consecutive quadrilaterals were computed right after each other. This allowed the system to reuse the offset vector computed for the previous quadrilateral. However, the additional burden introduced by the recursion slowed down the rendering compared to the direct approach. Even transforming the recursion into a sequential approach using queues resulted in a slower performance. As a result it was found that computing both offset vectors for each quadrilateral results in the best performance possible despite the fact that some offset vectors are computed twice.

Rendering the vessel segments as quadrilaterals results in a rather flat appearance. In order to achieve a more cylindrical effect, a fragment program can be used that uses darker colors at the edge of the vessel, i.e. the sides of the quadrilateral that are not connected to other quadrilaterals. To achieve this effect, texture coordinates can be assigned to the vertices of the quadrilateral. Based on

```
void main (in float4 color : COLOR0,
           in float2 texcoord : TEXCOORD0,
           out float4 result : COLOR) {
  result = color * (-(texcoord.y - 0.5) *
           (texcoord.y - 0.5) + 1.0);
}
```

Figure 13 Fragment program for making a quadrilateral appear as a cylindrical object.

these texture coordinates, the fragment program can determine the local position of every fragment with respect to the vessel segment and change the color for the fragment if the edge of the vessel is approached. Figure 13 shows the Cg code of the fragment program used in this implementation. The fragment program simply fades the color when approaching the edge of the vessel. A similar effect could be achieved by using a simple texture. However, a fragment program has two advantages over a texture. First, the fragment program is resolution independent and computes the color for each fragment whereas a texture may produce aliasing artifacts. Second, the fragment program is color independent. Hence, color coding could be used for the vessel segments which would be preserved by the fragment program but overwritten by a texture.

Since CPU manufacturers nowadays often times provide more than just a single core, this additional computational resource can be used by the current implementation as well by simply adding a thread that computes quadrilaterals representing vessel segments on the CPU for very core. To synchronize the threads, a queue was introduced which contains all vessel segments that are not rendered yet. Then a separate thread is started which acquires the next vessel segments from the queue and computes the quadrilaterals to represent these vessel segments. The resulting geometry data is stored in another queue. This other queue is then processed by the main thread which renders the quadrilaterals contained in that queue. On a dual-core processor, this then results in two parallel processes which compute the necessary geometric data required for rendering the vascular data set. As can be seen in the next section, the current implementation of the algorithm scales very well and utilizes all computational resources available resulting in an increased rendering performance.

4 Performance

Figure 14 shows the resulting image of the performance rendering method based on a single quadrilaterals for every vessel segment. As can be seen in the image, the rendering quality is still very good. Every of the 11 million vessel segments is represented in this image.

The performance of the algorithms is very good as well. All measurements were performed on a computer equipped with an Intel Core2 Duo E6850 running at 3.0 GHz with 2 GB of memory and an NVidia Quadro FX3700 graphics card running Linux. A Western Digital Raptor WD740ADFD spinning at 10,000 RPM was utilized as the permanent storage device. The original implementation based on an out-of-core methodology requires over a minute (65 seconds on average) to render a single frame due to the fact that the geometry needs to be loaded from disk all the time. Figure 15 shows a plot of the rendering frame rate over a period of time.

When using the improved rendering algorithm, the performance increases significantly. Since the methodology no longer has to rely on the hard drive as the main storage medium, but instead can store all the necessary information in main memory, a rendering frame rate of about 0.68 is achieved which amounts to 1.5 seconds per frame on average. A plot of the frame rates over time is included in Figure 16.

The algorithm scales very well as can be seen when adding a second thread to assist in the

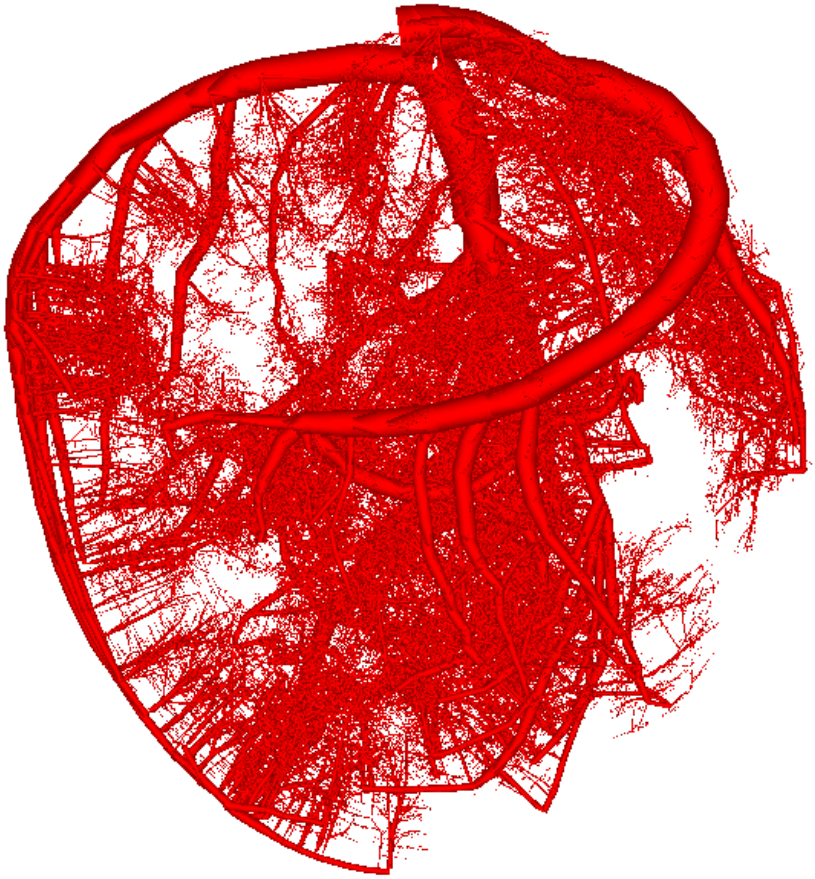

Figure 14 Performance rendering of the arterial tree of a pig heart.

computations necessary to compute all the quadrilaterals to represent the vessel segments. The rendering performance doubles to 1.32 frames per second which is equivalent to 0.75 seconds per frame which allows for an interactive investigation of the data set.

5 Conclusion and Future Work

This paper presented methodologies for extracting quantitative measurements from volumetric data sets at a very high accuracy. These morphometric measurements can be used to reconstruct the geometry of the vasculature resulting in an accurate visualization of the data. The measurements extracted include vessel length, vessel radius, and bifurcation angles which can be used as a statistical basis for growing additional vessels onto an existing vasculature. This way, large-scale vascular models can be generated that include vessels from the most proximal level down to the capillaries. Due to the vast amount of vessels included in such a data set, fast rendering methodologies are required to handle this amount of information. As discussed in this paper, using a more suitable rendering algorithm can result in an improvement of a factor of 43 in rendering performance over traditional out-of-core approaches.

In the future, the geometric reconstruction will be used to perform detailed flow simulation within the vessels to accurately analyze the flow properties within the vascular structure. Due to the high accuracy of the extracted measurements, the information can be used for disease detection, such as identification of areas of reduced flow. In addition, the extraction algorithm will be extended to

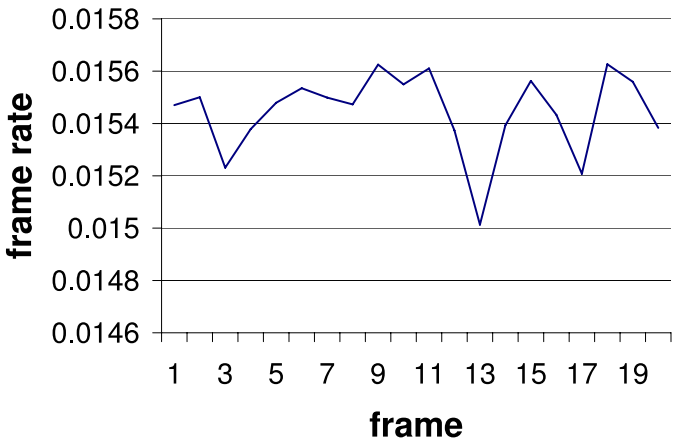

Figure 15 Rendering performance of out-of-core method for the large-scale arterial tree of a pig heart.

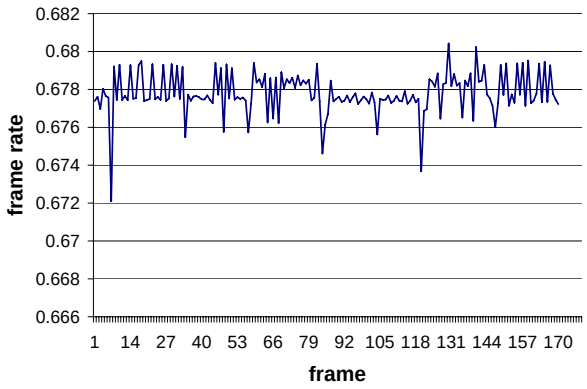

Figure 16 Rendering performance of performance method for the large-scale arterial tree of a pig heart.

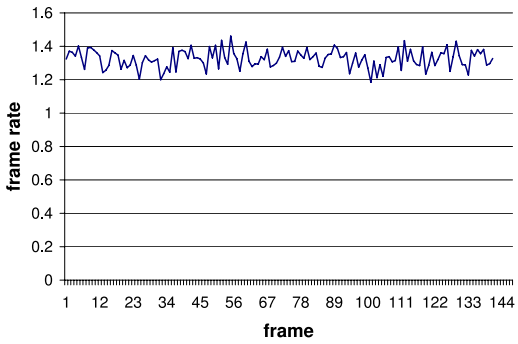

Figure 17 Rendering performance of performance method using two threads for the large-scale arterial tree of a pig heart.

identify individual fibers within volumetric data sets captured using optical coherence tomography of vascular tissue. An analysis of the fiber structure will result in the determination of mechanical properties of individual fibers to model the expansion of a fiber due to increased pressure induced by the blood flow.

6 Acknowledgements

The author wishes to thank Dr. Ghassan Kassab for a fruitful collaboration and providing the data sets. Also, the support of the Ohio Board of Regents and Research and Sponsored Programs, Wright State University, and its department of Computer Science and Engineering as well as the College of Engineering and Computer Science is greatly appreciated.

References

1 N. Amenta and R.-K. Kolluri S. Choi. The power crust. In *Proc. of 6th ACM Symp. on Solid Modeling*, pages 249–260, 2001.

2 G. Bertrand and Z. Aktouf. A three-dimensional thinning algorithm using subfields. *Vision Geometry III*, 2356:113–124, 1994.

3 I. Bitter, A. E. Kaufman, and M. Sato. Penalized-distance volumetric skeleton algorithm. *IEEE Trans. Visualization and Comp. Graphics*, 7(3):195–206, 2001.

4 G. Borgefors. On digital distance transforms in three dimensions. *Computer Vision and Image Understanding*, 64(3):368–376, 1996.

5 G. Borgefors, I. Nyström, and G. S. Di Baja. Computing skeletons in three dimensions. *Pattern Recognition*, 32(7):1225–1236, 1999.

6 S. Bouix and K. Siddiqi. Divergence-based medial surfaces. *ECCV*, 1842:603–618, 2000.

7 D. Brunner and G. Brunnett. Mesh segmentation using the object skeleton graph. In *Proc. IASTED International Conf. on Computer Graphics and Imaging*, pages 48–55. ACTA Press, 2004.

8 J. F. Canny. A computational approach to edge detection. *IEEE Trans. Pattern Analysis and Machine Intelligence*, PAMI-8(6):679–698, 1986.

9 N. D. Cornea, D. Silver, and P. Min. Curve-skeleton applications. In *IEEE Visualization 2005*, pages 95–102, 2005.

10 M. Couprie and R. Zrour. Discrete bisector function and euclidean skeleton. *Lecture Notes in Computer Science*, 3429:216–227, 2005.

11 T. K. Dey and S. Goswami. Tight cocone: A water-tight surface reconstructor. In *Proc. 8th ACM Sympos. Solid Modeling Applications*, pages 127–134, 2003.

12 U. Eckhardt and G. Maderlechner. Invariant thinning. *Pattern Recognition and Artificial Intelligence*, 7:1115–1144, 1993.

13 P. Felkel, A. L. Fuhrmann, A. Kanitasar, and R. Wegenkittel. Surface reconstruction of the branching vessels for augmented reality aided surgery. *BIOSIGNAL*, 1:252–254, 2002.

14 N. Gagvani and D. Silver. Parameter controlled volume thinning. *Graphical Models and Image Processing*, 61(3):149–164, 1999.

15 N. Gagvani and D. Silver. Animating volumetric models. *Academic Press Professional*, 63(6):443–458, 2001.

16 P. Golland and W.E.L. Grimson. Fixed topology skeletons. In *IEEE CVPR*, volume 1, pages 10–17, 2000.

17 W. Gong and G. Bertrand. A simple parallel 3d thinning algorithm. In *IEEE Pattern Recognition*, pages 188–190, 1990.

18 H. K. Hahn, B. Preim, D. Selle, and H. O. Peitgen. Visualization and interaction techniques for the exploration of vascular structures. In *IEEE Visualization 2001*, pages 395–402, 2001.

19 T. He, L. Hong, D. Chen, and Z. Liang. Reliable path for virtual endoscopy: Ensuring complete examination of human organs. *IEEE Trans. Visualization and Comp. Graphics*, 7(4):333–342, 2001.

20 Peter J. Hunter and Thomas K. Borg. Integration from proteins to organs: the iups physiome project. *Nature Reviews | Molecular Cell Biology*, 4:237–243, 2003.

21 B. Kaimovitz, Y. Lanir, and G. S. Kassab. Large-scale 3-d geometric reconstruction of the porcine coronary arterial vasculature based on detailed anatomical data. *Ann. Biomed. Eng.*, 33(11):1517–1535, 2005.

22 T. Lee and R.L. Kashyap. Building skeleton models via 3-d medial surface/axis thinning algorithms. *CVGIP: Graphical Models and Image Processing*, 56(6):462–478, 1994.

23 S. Lobregt, P. W. Verbeek, and F. C. A. Groen. Three-dimensional skeletonization: Principle and algorithm. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2(1):75–77, 1980.

24 C. Lohou and G. Bertrand. A 3d 12-subiteration thinning algorithm based on p-simple points. *Discrete Applied Mathematics*, 139:171–195, 2004.

25 C. M. Ma and M. Sonka. A fully parallel 3d thinning algorithm and its applications. *Computer Vision and Image Understanding*, 64(3):420–433, 1996.

26 G. Malandain and S. Fernandez-Vidal. Euclidean skeletons. *Image and Vision Computing*, 16:317–327, 1998.

27 A. Manzanera, T. Bernard, F. Preteux, and B. Longuet. A unified mathematical framework for a compact and fully parallel n-d skeletonization procedure. *Vision Geometry VIII*, 3811:57–68, 1999.

28 Y. Masutani, K. Masamune, and T. Dohi. Region-growing-based feature extraction algorithm for tree-like objects. *Visualization in Biomedical Computing*, pages 161–171, 1996.

29 S. Oeltze and B. Preim. Visualization of Anatomic Tree Structures with Convolution Surfaces. In *IEEE/Eurographics Symposium on Visualization*, pages 311–320, 2004.

30 S. Oeltze and B. Preim. Visualization of Vasculature with Convolution Surfaces: Method, Validation and Evaluation. *IEEE Trans. Medical Imaging*, 24(4):540–548, 2005.

31 K. Palagyi and A. Kuba. Directional 3d thinning using 8 subiterations. *Discrete Geometry for Computer Imagery: Lecture Notes in Computer Science*, 1568, 1999.

32 K. Palagyi and A. Kuba. A parallel 3d 12-subiteration thinning algorithm. *Graphical Models and Image Proc.*, 61(4):199–221, 1999.

33 D. Perchet, C. I. Fetita, and F. Preteux. Advanced navigation tools for virtual bronchoscopy. In *SPIE Conf. on Image Processing: Algorithms and Systems III*, volume 5298, pages 147–158, 2004.

34 C. Pudney. Distance-ordered homotopic thinning: A skeletonization algorithm for 3d digital images. *Computer Vision and Image Understanding*, 72(3):404–413, 1998.

35 A. Puig, D. Tost, and I. Navazo. An interactive celebral blood vessel exploration system. In *IEEE Visualization 97*, pages 443–446, 1997.

36 Felix Ritter, Christian Hansen, Volker Dicken, Olaf Konrad, Bernhard Preim, and Heinz-Otto Peitgen. Real-Time Illustration of Vascular Structures. In *IEEE Visualization 2006*, pages 877–884, 2006.

37 P.K. Saha, B.B. Chaudhuri, and D. Dutta Majumder. A new shape preserving parallel thinning algorithm for 3d digital images. *Pattern Recognition*, 30(12):1939–1955, 1997.

38 H. Schirmacher, M. Zöckler, D. Stalling, and H. Hege. Boundary surface shrinking - a continuous approach to 3d center line extraction. In *Proc. of IMDSP*, pages 25–28, 1998.

39 J.A. Sethian. Fast marching methods. *SIAM Review*, 41(2):199–235, 1999.

40 Carsten Stoll, Stefan Gumhold, and Hans-Peter Seidel. Visualization with stylized primitives. In Claudio Silver, Eduard Gröller, and Holly Rushmeier, editors, *IEEE Visualization 2005*, pages 695–702, 2005.

41 H. Sundar, D. Silver, N. Gagvani, and S. Dickinson. Skeleton based shape matching and retrieval. In *Shape Modeling Int'l*, pages 130–139, 2003.

42 K. Suresh. Automating the cad/cae dimensional reduction process. In *ACM Symp. On Solid Modeling and Applications*, pages 76–85, 2003.

43 S. Svensson, I. Nystrom, and G. Sanniti di Baja. Curve skeletonization of surface-like objects in 3d images guided by voxel classification. *Pattern Recognition Letters*, 23(12):1419–1426, 2002.

44 A. Telea and A. Vilanova. A robust level-set algorithm for centerline extraction. In *Eurographics/IEEE Symp. On Data Visualization*, pages 185–194, 2003.

45 Y. F. Tsao and K. S. Fu. A parallel thinning algorithm for 3d pictures. *Computer Vision, Graphics and Image Proc.*, 17:315–331, 1981.

46 L. Wade and R. E. Parent. Automated generation of control skeletons for use in animation. *The Visual Computer*, 18(2):97–110, 2002.

47 M. Wan, F. Dachille, and A. Kaufman. Distance-field based skeletons for virtual navigation. In *IEEE Visualization 2001*, pages 239–246, 2001.

48 Thomas Wischgoll. *to appear in Visualization and Mathematics*, chapter Computing Center-Lines: An Application of Vector Field Topology. Springer-Verlag, Heidelberg, Germany, 2008.

49 Thomas Wischgoll, Joerg Meyer, Benjamin Kaimovitz, Yoram Lanir, and Ghassan S. Kassab. A Novel Method for Visualization of Entire Coronary Arterial Tree. *Annals of Biomedical Engineering*, 35(5):694–710, 2007.

50 Thomas Wischgoll, Joerg Meyer, Benjamin Kaimovitz, Yoram Lanir, and Ghassan S. Kassab. A novel method for visualization of entire coronary arterial tree. *Annals of Biomedical Engineering*, 35(5):694–710, 2007.

51 Z. Yu and C. Bajaj. A segmentation-free approach for skeletonization of gray-scale images via anisotropic vector diffusion. In *CVPR 2004*, pages 415–420, 2004.

52 Y. Zhou, A. Kaufman, and A. W. Toga. Three-dimensional skeleton and centerline generation based on an approximate minimum distance field. *The Visual Computer*, 14:303–314, 1998.

53 Y. Zhou and A. W. Toga. Efficient skeletonization of volumetric objects. *IEEE Trans. Visualization and Comp. Graphics*, 5(3):196–209, 1999.