

# Filament Tracking and Encoding for Complex Biological Networks

David M. Mayerich\*  
Department of Computer Science  
Texas A&M University

John Keyser†  
Department of Computer Science  
Texas A&M University‡



Figure 1: Traced and segmented microvasculature of the mouse spinal cord. The entire network has been traced and displayed (orange) as well as several sub-networks (green and purple) represented as cliques within the network.

## Abstract

We present a framework for segmenting and storing filament networks from scalar volume data. Filament structures are commonly found in data generated using high-throughput microscopy. These data sets can be several gigabytes in size because they are either spatially large or have a high number of scalar channels. Filaments in microscopy data sets are difficult to segment because their diameter is often near the sampling resolution of the microscope, yet single filaments can span large data sets. We describe a novel method to trace filaments through scalar volume data sets that is robust to both noisy and under-sampled data. We use a GPU-based scheme to accelerate the tracing algorithm, making it more useful for large data sets. After the initial structure is traced, we can use this information to create a bounding volume around the network and encode the volumetric data associated with it. Taken together, this framework provides a convenient method for accessing network structure and connectivity while providing compressed access to the original volumetric data associated with the network.

**CR Categories:** E.4 [Image Processing and Computer Vision]: Image Representation—; I.3.5 [Computer Graphics]: Computational Geometry and Object Modeling—object hierarchies; I.4 [Image Processing and Computer Vision]: Reconstruction—;

**Keywords:** microscopy, fiber, volumetric, segmentation

## 1 Introduction

Long and thin filament structures are common in microscope imaging. Structures such as neuronal fibers and capillaries traverse large data sets but have diameters close to the maximum resolvable distance of the imaging medium. With the introduction of high-throughput microscopy techniques such as Array Tomography [Micheva and Smith 2007] and Knife-Edge Scanning Microscopy (KESM) [Mayerich et al. 2008], fast algorithms for extracting filament positions and connectivity have become more important.

We present a framework for extracting topological and structural information from dense filament networks stored in scalar volume data sets. In particular, we describe a novel method for extracting the network structure that is robust in the presence of noise. This method is specifically designed for use in high-throughput microscopy, however we also show that these techniques can be used in standard medical imaging as well.

## 2 Previous Work

Many filament tracking algorithms have been developed for medical imaging methods such as X-Ray and MRI. There are numerous publications in this area [Sarwal and Dhawan 1994; Zhang et al. 2007; Yu and Bajaj ; Tozaki et al. 1996; Sato et al. 1998] and an extensive review on the subject is given by Kirbas, et al. [2004]. These algorithms generally rely on filtering to remove high-frequency noise followed by isosurface extraction. Due to the thin

---

\*e-mail: mayerichd@neo.tamu.edu

†e-mail: keyser@cs.tamu.edu

‡The authors would like to thank Stephen Smith and Charles Taylor for their Array Tomography and Lung data. Funding was provided by NIH (#1R01-NS54252) and NSF (#CCR-0220047).

diameters of filaments in microscopy data sets, filtering tends to destroy important features. The presence of high-frequency noise, however, prevents the computation of a valid isosurface.

Algorithms have also been developed for the segmentation of filament data in serial microscopy volumes. In particular, the algorithm developed by Al-Kofahi et al.[2002] is designed to track neuron filaments in confocal microscopy image stacks. This method follows a filament by taking samples on the filament surface. Because of this, filaments that have ill-defined or "fuzzy" surfaces are difficult to follow. The algorithm that we propose works for both cyclic and acyclic structures and has aspects in common with both the center-line tracking used by Al-Kofahi et al. and with filter matching methods. Also, we do not require pre-processing of the raw image data.

### 3 Filament Tracking

We track individual filaments through a data set using a predictor-corrector algorithm. We estimate the medial axis of a filament by determining the path of a particle, which we call a *tracer*, over time as it moves down a filament. At any given time, the tracer has three properties:

- The tracer position  $\mathbf{p}$  is an estimated point on the filament axis.
- A vector  $\mathbf{v}$  representing the normalized filament trajectory at the current tracer position.
- An estimate of the filament radius  $r$  at the current point.

The axis of the filament is traversed using the predictor-corrector described in Algorithm 1. Each of the functions **PredictPath**, **CorrectPosition**, and **EstimateSize** used to update the properties of the tracer over time are described next.

---

**Algorithm 1** The predictor-corrector algorithm used to determine the next point along the axis of the filament.

---

**Function TracerStep**( $\Delta t, \mathbf{p}, \mathbf{v}, r$ )  
 $\mathbf{v} = \mathbf{PredictPath}(\mathbf{p}, \mathbf{v}, r)$   
 $\mathbf{p}' = \mathbf{p} + \Delta t \mathbf{v}$   
 $\mathbf{p} = \mathbf{CorrectPosition}(\mathbf{p}', \mathbf{v}, r)$   
 $r = \mathbf{EstimateSize}(\mathbf{p}, \mathbf{v}, r)$

---

#### 3.1 Predicting Trajectories

Given the current tracer properties, we want an accurate method for estimating the trajectory of the tracer  $\mathbf{v}$  such that the next estimate of the tracer position

$$\mathbf{p}' = \mathbf{p} + \Delta t \mathbf{v} \quad (1)$$

lies as close to the filament axis as possible. In order to find the optimal direction vector, we sample a series of vectors  $\mathbf{v}_i$ , where  $i = 0 \dots N$ , that lie within a solid angle  $\alpha$  of the previous vector  $\mathbf{v}$ .

We now discuss the heuristic function used to evaluate each sample vector. For each vector, we take a *planar volume sample* (PVS), or cross-sectional image orthogonal to  $\mathbf{v}$  (Figure 2) at  $\mathbf{p}$ . In the ideal case where  $\mathbf{v}_1$  perfectly matches the filament trajectory, the cross-section of the filament on the PVS image will be minimal and positioned at the center. We specify the cost function  $C_{pred}$ , which provides a minimum response when these conditions are met.

$$C_{pred} = \sum_{x=0}^n \sum_{y=0}^n w(x, y) |p_{pvs}(x, y) - f_{template}(x, y)| \quad (2)$$

Here,  $p_{pvs}(x, y)$  is a point on the PVS image for the computed sample.  $f_{template}(x, y)$  is a Gaussian template function and  $w(x, y)$  is a windowing function.

The function  $f_{template}(x, y)$  is a user specified template that matches the filament contrast. For example, the template function for a dark filament on a light background could be specified as

$$f_{template}(x, y) = 1.0 - G(x, y) \quad (3)$$

where  $G$  is a Gaussian function scaled to the range between 0.0 and 1.0. The windowing function  $w(x, y)$  is used to eliminate the effect of filaments that may be near the filament under consideration. The window function is also used to make the PVS rotationally invariant since the actual sample image is square. For  $w(x, y)$  we used a Gaussian slightly wider than the template function. Even for low-contrast filaments,  $C_{pred}$  provides an adequate measurement of alignment (Figure 3).

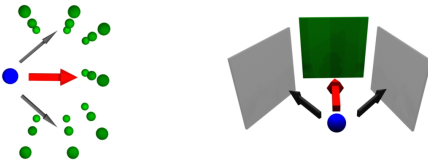


Figure 2: View vector samples are taken by sampling rays that intersect a surrounding cube (left). The sample itself is an image made up of a cross section of the data set taken just ahead of the camera at each of the sample points (right).

An optimal direction estimate  $\mathbf{v}'$  is selected by finding the sample view vector with the lowest cost (i.e. the vector that produces a cross-section image most closely matching the contrast of the template).

The use of a single PVS image along the sample vector works well for very clean data sets, however noise and contrast fluctuations are frequently present in microscopy data sets, particularly when using high-throughput techniques. In these cases, we take two or more cross-sectional sub-samples spaced evenly along the sample direction. The distance between  $\mathbf{p}$  and the furthest sample plane is user-defined but scales linearly with the estimate of the filament cross-section (Section 3.3). By default, this distance is equal to the size of the sample plane, causing our algorithm to sample a cube-shaped

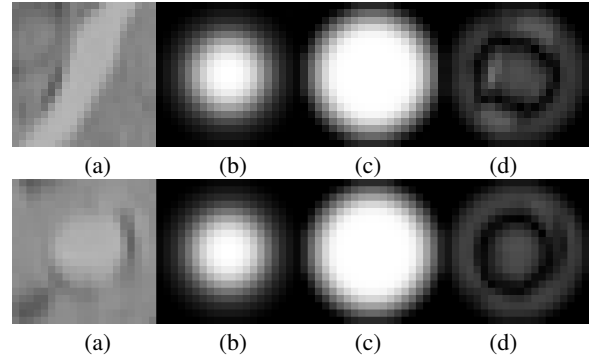


Figure 3: Predictor evaluation of a low-contrast filament (unaligned [top] and aligned [bottom]). (a)  $p_{pvs}$ , (b)  $f_{template}$ , (c)  $w$ , (d)  $C_{pred}$ . The average image intensity  $\frac{C_{pred}}{n^2}$  is 31.2 (aligned) versus 37.8 (unaligned).

volume aligned along each direction estimate. We have found that this gives significant improvements in Array Tomography when the stain used to label the cell is unevenly distributed throughout the filament. This is also useful for filaments in KESM images that drop below the minimum resolvable diameter, causing small gaps in the filament.

### 3.2 Axis Correction

Once a new point is created along the filament path, there is generally some error in the point position due to the finite number of samples taken in the predictor stage. Correction is performed by taking a second series of samples that are used to adjust the tracer position  $\mathbf{p}'$ . Instead of sampling a series of tracer directions, we sample a set of tracer positions. Each sample lies in the plane defined by the new vector  $\mathbf{v}'$  within a certain region  $\epsilon$  from  $\mathbf{p}'$ . The cost function used to evaluate each sample point is given by

$$C_{corr} = \sum_{x=0}^n \sum_{y=0}^n [p_{pvs}(x, y)G(x, y)] \quad (4)$$

where the values for  $p_{image}$  and  $G(x, y)$  are as described above. The sample with the lowest value  $C_{corr}$  is selected as the new tracer position. This is similar to a simple correlation algorithm [Gonzalez and Woods 2002]. This equation assumes that the filament is dark (in a generally lighter data set). We can handle the opposite case by inverting the PVS image. We could also use the cost function specified for  $C_{pred}$ , but found  $C_{corr}$  to be less sensitive to filament diameter (which has not yet been re-calculated for this time step) and requires fewer operations.

### 3.3 Estimation of Filament Radius

Finally, we have to update the value of  $r$  for the current time step. Estimation of the radius of the filament allows us to dynamically adjust the size of the template and window functions used in  $C_{pred}$  and  $C_{corr}$ . As the size of these functions approach the size of the filament cross-section, the accuracy of the algorithm improves.

The radius of the filament is acquired by taking a series of scale-space samples at the tracer position  $\mathbf{p}$  oriented along  $\mathbf{v}$ . We do this by calculating several discrete sizes for  $f_{template}$  and  $w$  and testing which one provides an optimal response to the predictor function  $C_{pred}$ .

Depending on the types of data being analyzed, degenerate cases in tracing have to be dealt with. These include branching and filament termination. The termination of filaments is handled by implementing constraints on filament size and detecting self-intersections and intersections with other filaments. Intersections are detected by searching through all filaments and computing whether or not the filament intersects the volume being sampled by the tracer.

### 3.4 Seed Point Selection

In order for our algorithm to properly execute, an initial state for the tracer has to be determined. We create initial values for  $\mathbf{p}$ ,  $\mathbf{v}$ , and  $r$  using *seed points* to select filaments. We define a seed point as a location in 3D space inside of (or close to) a filament to be traced. In this section, we discuss how to initialize a tracer using a seed point as well as some techniques for automating seed point selection.

After a filament is labeled by a seed point, two tracers are required to completely trace it. Each tracer travels in opposite directions down the filament. Initial seed points were specified by setting a

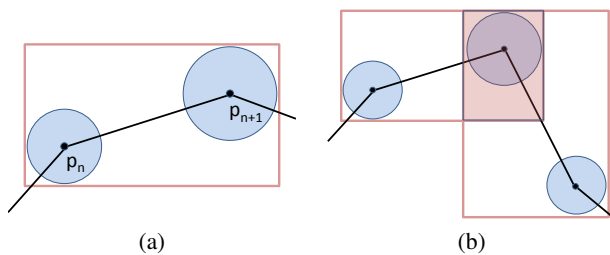


Figure 4: Bounding two spheres of radii  $r_n$  and  $r_{n+1}$  (a) guarantees that the filament segment will be bound however there will be significant overlap (b).

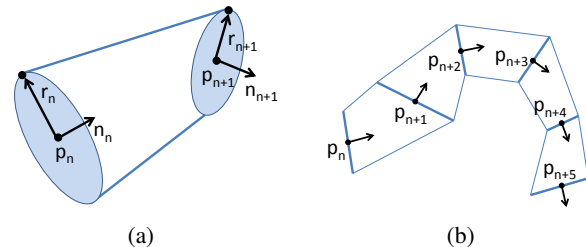


Figure 5: Truncated generalized cones (a) fit end-to-end to construct the bounding volume for a filament (b).

conservative threshold and seeding points known to be within the filament network.

### 3.5 Acceleration on Graphics Hardware

One reason we believe that PVS images are not often used is the time required for evaluation and interpolation of the sample as well as resizing the templates used in the evaluation. Using commodity graphics hardware, we have actually found that PVS images can be created and evaluated efficiently. For extremely large data sets, we load the scalar volume data in blocks that will fit into the memory available on modern graphics cards. The PVS images can then be specified using texture coordinates on a quadrilateral and rendered to a texture. This allows the GPU to perform accelerated texture lookup and interpolation, performing many of the operations in parallel.

## 4 Data Storage

Storing volumetric data on a uniform grid has several advantages. In particular, random access can be done in constant-time making fast ray-tracing possible. GPU-based algorithms used in volume visualization also require data to be stored as a uniform grid in graphics memory. For the large size of 3D grids required to store comprehensive filament networks, however, this advantage is quickly lost due to the constant need to transfer 3D texture data across the PCI-E/AGP bus. As noted previously, even dense filament networks tend to be described by only a sparse number of scalar values in a 3D data set. Most of the information stored in a uniform 3D grid describes the spacing between filaments. In this section we will discuss methods for encoding the network data for efficient storage and analysis.

### 4.1 Network Bounding Volumes

In order to encode the data associated with a filament network, we must first determine if any given voxel is part of the network.

Some previous work has been done on encoding sparse data. In particular, L-Block structures [McCormick et al. 2004; Doddapaneni 2004] have been used to represent connected structures in volumetric data sets. These algorithms use a connected series of axis-aligned bounding boxes to define an optimal bounding volume. This method requires many iterations to converge to a solution and is usually very time consuming to compute. In addition, L-Blocks use an adaptive thresholding scheme to determine if a voxel is part of the network. As we have described previously (Section 2), these segmentation techniques do not work well for high-throughput microscopy data. L-Blocks are also designed for archival storage, requiring that the structure be uncompressed for direct access to the data.

Our method uses an implicit representation of the network from the information acquired during tracking (Section 3). Recall that the data produced by the tracking algorithm consists of:

- a list of vertices, or *nodes*, that contain a position and single radius measurement, and
- an edge list describing how each of these nodes are connected.

Given the traced network, we would like to create a bounding volume around the network that has several key features:

- The bounding volume should store a minimum amount of voxel data required to reconstruct the filament.
- The mapping from our structure to the original data set should not require that the data be re-sampled. That is, we want to retain the original volumetric data.
- We require that the structure store all of the necessary information to reconstruct the filament independently of all other filaments.

In order to maintain simple one-to-one mapping between the original data and the bounding structure, we could use a connected series of Axis-Aligned Bounding Boxes (AABBs) placed along each filament. The voxel data stored inside the AABB can be inserted or extracted from the original image without re-sampling. The overhead for an AABB is also minimal, requiring a single position and size along each axis.

Given a single filament segment as two points with radii, we can guarantee that the filament segment will be bounded by creating a bounding box around two spheres positioned at the points  $\mathbf{p}_n$  and  $\mathbf{p}_{n+1}$  with radii  $r_n$  and  $r_{n+1}$  (Figure 4a). As neighboring segments are considered, however, the bounding volumes contain a significant amount of overlap (Figure 4b). In addition to requiring additional storage space, any analysis or visualization algorithms have to take special care when dealing with this redundant data.

A tighter bound could be constructed by using a series of truncated generalized cones (TGCs), commonly used in vascular visualization [Hahn et al. 2001]. Each TGC is defined by two adjacent points along a filament and has a medial axis defined by the line segment connecting the points. The TGCs are connected end-to-end with the end caps oriented using normals defined as

$$\mathbf{n}_b = \frac{1}{2} \left( \frac{\mathbf{b} - \mathbf{a}}{\|\mathbf{b} - \mathbf{a}\|} + \frac{\mathbf{c} - \mathbf{b}}{\|\mathbf{c} - \mathbf{b}\|} \right) \quad (5)$$

where  $\mathbf{a}$ ,  $\mathbf{b}$ , and  $\mathbf{c}$  are three consecutive points on the estimated medial axis of a filament. The radius of each end cap is equal to the filament radius defined at the associated point (Figure 5).

In order to determine if a given point exists within the bounding volume, it is useful to implicitly define the region within a TGC. Assuming that there is a point  $\mathbf{p}_x$  under consideration. We want to

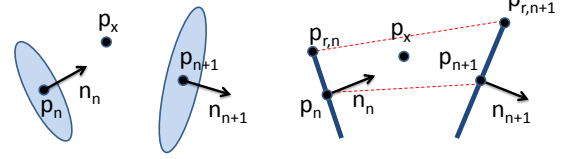


Figure 6: (a) End caps of a TGC and (b) the intersection of the TGC with the plane defined by  $\mathbf{p}_x$ ,  $\mathbf{p}_n$ , and  $\mathbf{p}_{n+1}$ .

determine if the point lies within the TGC specified by the end caps  $(\mathbf{p}_n, \mathbf{n}_n, r_n)$  and  $(\mathbf{p}_{n+1}, \mathbf{n}_{n+1}, r_{n+1})$  where  $\mathbf{p}_n$  is the point at the center of the end cap,  $\mathbf{n}_n$  is the end cap normal calculated from Equation 5 and  $r_n$  is the end cap radius. We first find the plane that passes through points  $\mathbf{p}_x$ ,  $\mathbf{p}_n$ , and  $\mathbf{p}_{n+1}$  with normal (Figure 6a)

$$\mathbf{n}_{\text{plane}} = \frac{\mathbf{p}_{n+1} - \mathbf{p}_n}{\|\mathbf{p}_{n+1} - \mathbf{p}_n\|} \times \frac{\mathbf{p}_x - \mathbf{p}_n}{\|\mathbf{p}_x - \mathbf{p}_n\|} \quad (6)$$

For each end cap, we then find the end point of the line segment that represents the intersection of the end cap with the plane:

$$\mathbf{p}_{r,n} = \mathbf{p}_n + r_n(\mathbf{n}_{\text{plane}} \times \mathbf{n}_n) \quad (7)$$

we know that the point  $\mathbf{p}_x$  is within the TGC if it lies within the polygon formed by  $(\mathbf{p}_n, \mathbf{p}_{r,n}, \mathbf{p}_{r,n+1}, \mathbf{p}_{n+1})$  (Figure 6b).

Using the implicit representation of the filament volume described above (Section 4.1), we can determine if any given voxel lies within the bounding volume. This test can be accelerated by computing the AABB around each TGC. Computing the AABB for a TGC can be done efficiently knowing the position  $\mathbf{p}_n$ , normal  $\mathbf{n}_n$ , and radius  $r_n$  of each end cap. Since the end cap is circular, we can compute the minimum and maximum extent of the circle by scaling the radius by the angle between the normal  $\mathbf{n}_n$  and each axis. For example, the minimum and maximum size of the bounding volume for end cap  $n$  along the X-axis is computed as

$$AABB_{X,n} = \mathbf{p}_n \pm r(1 - |\mathbf{n}_n \cdot \mathbf{X}|) \quad (8)$$

where  $\mathbf{X} = [1 \ 0 \ 0]$ . The position and size of the AABB along the X-axis are

$$AABB_{X,pos} = \min(AABB_{X,n}, AABB_{X,n+1}) \quad (9)$$

$$AABB_{X,size} = \max(AABB_{X,n}, AABB_{X,n+1}) - AABB_{X,pos} \quad (10)$$

Standard collision detection algorithms can be used to sort the AABBs, allowing each query to be done in time logarithmic to the number of TGC segments in the model. If an initial intersection is found, the point is then tested against the implicit function for the associated TGC.

## 4.2 Volumetric Encoding

For each voxel that lies within the bounding volume of the filament network, we insert it into a dynamic tubular grid (DT-Grid) data structure [Nielsen and Museth 2006]. This is an efficient encoding scheme that is generally used to store a narrow band around an implicit surface in evolving front applications [Sethian 1999; Osher and Fedkiw 2002]. Because of the thin nature of filament networks, this structure is ideally suited to store volumetric information representing each filament. Implementation details can be found in the article by Nielsen and Museth.

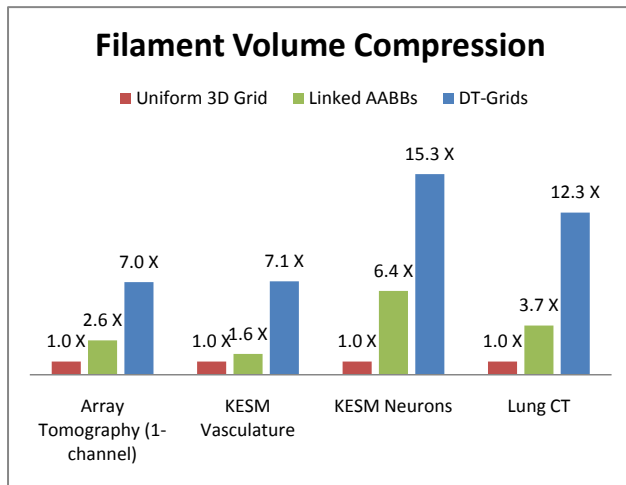


Figure 7: Comparison of the compression achieved by storing the volume data on a uniform grid, as a series of AABBs, and using DT-Grids.

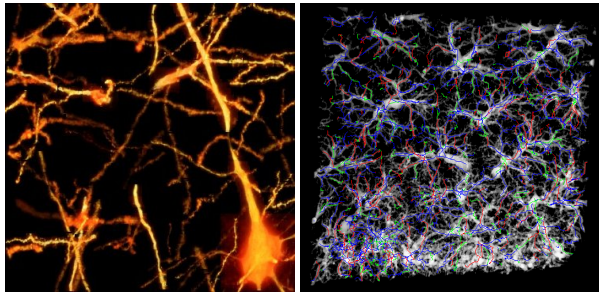


Figure 8: (left) Visualization of segmented neurons in AT data and (right) traced lung filaments overlaid over a volume visualization of the original data set.

## 5 Results

We have tested our algorithm for tracking filaments in several high-throughput data sets, including:

- Neuronal filaments from Array Tomography data (Figure 8).
- A network of capillaries from a KESM data set.
- A network of neurons from a KESM data set.
- Macro-scale lung vasculature from a CT data set (Figure 8).

After filament tracking, we store the data associated with the network. We have found that we can achieve significant compression results by culling volumetric data that is not associated with the filament network. This can be done by storing data in a series of bounding boxes placed along each filament or using DT-Grids [Nielsen and Museth 2006] (Figure 7).

Since the culled data often contains significant amounts of noise, visualization of the network after culling shows significant improvements in quality (Figure 10 and 11). Our tracking method also extracts the network connectivity, allowing us to traverse the network (Figure 9) as well as extract independent components (Figure 11c and 1).

## 6 Conclusions and Future Work

In this paper we have proposed a framework for segmenting filament networks from scalar volume data sets. These techniques are the only methods we have found that can successfully segment the dense networks of fine filaments found in high-throughput microscopy. Because of the size of high-throughput data sets and the rate at which they can be produced, our main goal was fast and automated segmentation. We have also tested our algorithms on more standard forms of biomedical imaging.

These algorithms provide a basis modeling filament networks. Our algorithms determine network connectivity based on proximity, which is not always an accurate method, particularly when dealing with filament diameters near the sampling resolution of the microscope. For example, an intersection in a network made up of several neurons could represent a branch in a single neuron, a synaptic connection between two neurons, or simply two filaments passing very close to each other. Methods for evaluation and refinement are therefore necessary for creating accurate network models.

## References

- AL-KOFAHI, K., LASEK, S., SZAROWSKI, D., PACE, C., NAGY, G., TURNER, J., AND ROYSAM, B. 2002. Rapid automated three-dimensional tracing of neurons from confocal image stacks. *IEEE Transactions on Information Technology in Biomedicine* 6, 171–186.
- DODDAPANENI, P. 2004. *Segmentation Strategies for Polymerized Volume Data Sets*. PhD thesis, Department of Computer Science, Texas A&M University.
- GONZALEZ, R. C., AND WOODS, R. E. 2002. *Digital Image Processing*, 2nd ed. Prentice Hall.
- HAHN, H. K., PREIM, B., SELLE, D., AND PEITGEN, H. O. 2001. Visualization and interaction techniques for the exploration of vascular structures. *Proceedings of the conference on Visualization'01*, 395–402.
- KIRBAS, C., AND QUEK, F. 2004. A review of vessel extraction techniques and algorithms. *ACM Computing Surveys* 36, 81–121.
- MAYERICH, D., ABBOTT, L. C., AND MCCORMICK, B. H. 2008. Knife-edge scanning microscopy for imaging and reconstruction of three-dimensional anatomical structures of the mouse brain. *Journal of Microscopy in press*.
- MCCORMICK, B., BUSSE, B., DODDAPANENI, P., MELEK, Z., AND KEYSER, J. 2004. Compression, segmentation, and modeling of filamentary volumetric data. 333–338.
- MICHEVA, K. D., AND SMITH, S. J. 2007. Array tomography: A new tool for imaging the molecular architecture and ultrastructure of neural circuits. *Neuron* 55, 25–36.
- NIELSEN, AND MUSETH. 2006. Dynamic tubular grid: An efficient data structure and algorithms for high resolution level sets. *Journal of Scientific Computing* 26, 261–299.
- OSHER, S. J., AND FEDKIW, R. P. 2002. *Level Set Methods and Dynamic Implicit Surfaces*. Springer.
- SARWAL, A., AND DHAWAN, A. 1994. 3-d reconstruction of coronary arteries. *IEEE Conference on Engineering in Medicine and Biology 1*, 504–505.

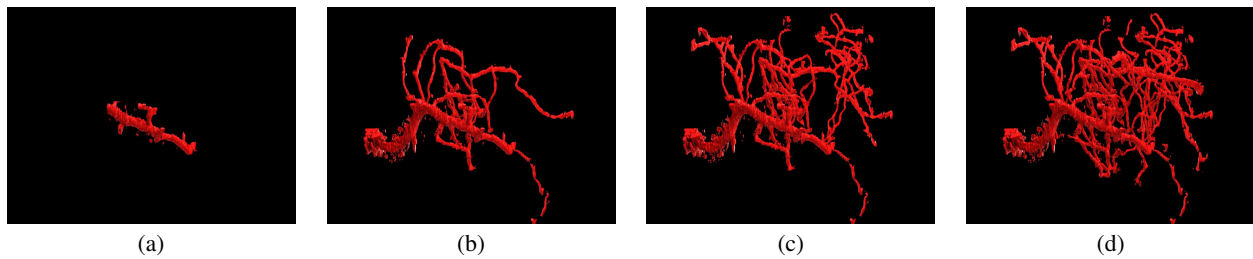


Figure 9: Contouring of filaments selected using a breadth-first search of the network starting from a single user-defined filament(a) - (d).

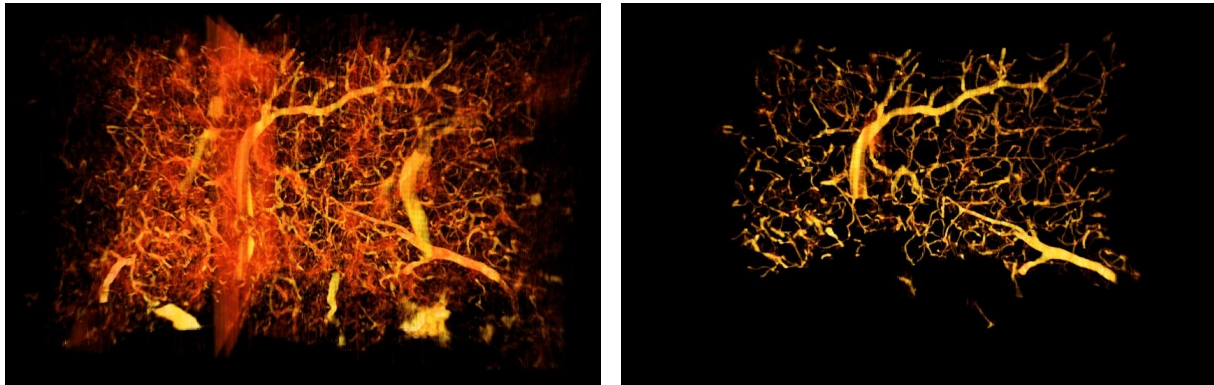


Figure 10: The original volume visualization of the microvasculature in a mouse brain (left) contains background noise and parts of unwanted cellular structures. Tracing and segmentation extract the desired filament and surface information which can be visualized more clearly (right).

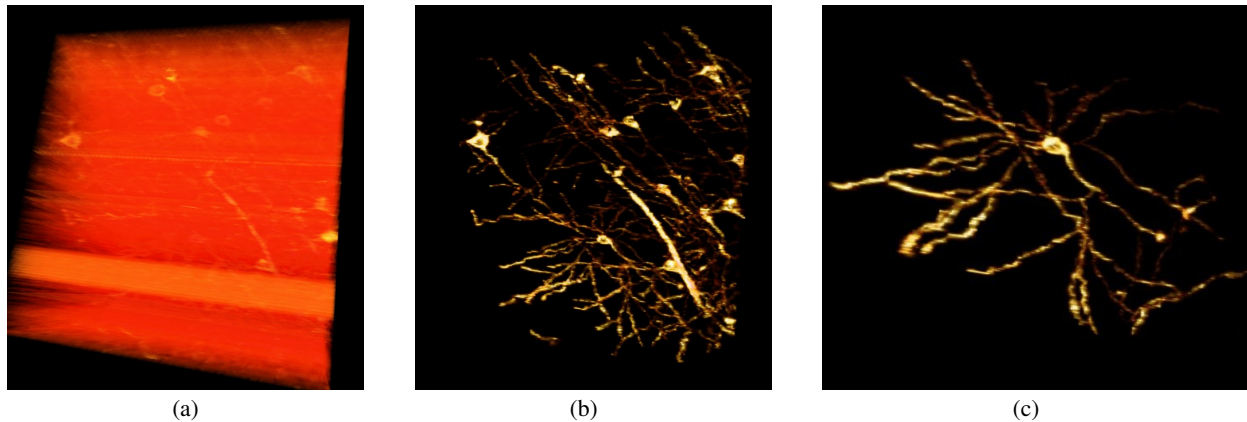


Figure 11: Microscopic data set containing several neurons. The original volume visualization is completely inadequate and it is impossible to extract a contour surface using standard techniques (a). After tracing, blocking, and contrast enhancement of individual filaments, the structure becomes visible (b). Visualization of a filament network representing a single cell (c).

SATO, Y., NAKAJIMA, S., SHIRAGA, N., ATSUMI, H., YOSHIDA, S., KOLLER, T., GERIG, G., AND KIKINIS, R. 1998. Three-dimensional multi-scale line filter for segmentation and visualization of curvilinear structures in medical images. *Medical Image Analysis* 2, 143–168.

SETHIAN, J. A. 1999. *Level Set Methods and Fast Marching Methods: Evolving Interfaces in Computational Geometry, Fluid Mechanics, Computer Vision, and Materials Science*. Cambridge University Press.

TOZAKI, T., KAWATA, Y., NIKI, N., OHMATSU, H., EGUCHI, K., AND MORIYAMA, N. 1996. Three-dimensional analysis of

lung areas using thin slice ct images. *Proc. SPIE* 2709, 1–11.

YU, Z., AND BAJAJ, C. A segmentation-free approach for skeletonization of gray-scale images via anisotropic vector diffusion. *Computer Vision and Pattern Recognition, 2004. CVPR 2004. Proceedings of the 2004 IEEE Computer Society Conference on* 1.

ZHANG, Y., BAZILEVS, Y., GOSWAMI, S., BAJAJ, C. L., AND HUGHES, T. J. R. 2007. Patient-specific vascular nurbs modeling for isogeometric analysis of blood flow. *Computer Methods in Applied Mechanics and Engineering* 196, 2943–2959.