IMAGING AND COMPUTATIONAL METHODS FOR EXPLORING SUB-CELLULAR ANATOMY

A Dissertation

by

DAVID MATTHEW MAYERICH

Submitted to the Office of Graduate Studies of
Texas A&M University
in partial fulfillment of the requirements for the degree of

DOCTOR OF PHILOSOPHY

May 2009

Major Subject: Computer Science

IMAGING AND COMPUTATIONAL METHODS FOR EXPLORING SUB-CELLULAR ANATOMY

A Dissertation

by

DAVID MATTHEW MAYERICH

Submitted to the Office of Graduate Studies of
Texas A&M University
in partial fulfillment of the requirements for the degree of

DOCTOR OF PHILOSOPHY

Approved by:

Chair of Committee,	John Keyser
Committee Members,	Yoonsuck Choe
	Louise Abbott
	Donald House
Head of Department,	Valerie Taylor

May 2009

Major Subject: Computer Science

ABSTRACT

Imaging and Computational Methods for Exploring Sub-Cellular Anatomy. (May 2009)

David Matthew Mayerich, B.S., Southwestern Oklahoma State University;

M.S., Texas A&M University

Chair of Advisory Committee: Dr. John Keyser

The ability to create large-scale high-resolution models of biological tissue provides an excellent opportunity for expanding our understanding of tissue structure and function. This is particularly important for brain tissue, where the majority of function occurs at the cellular and sub-cellular level. However, reconstructing tissue at sub-cellular resolution is a complex problem that requires new methods for imaging and data analysis.

In this dissertation, I describe a prototype microscopy technique that can image large volumes of tissue at sub-cellular resolution. This method, known as Knife-Edge Scanning Microscopy (KESM), has an extremely high data rate and can capture large tissue samples in a reasonable time frame. We can therefore image complete systems of cells, such as whole small animal organs, in a matter of days.

I then describe algorithms that I have developed to cope with large and complex data sets. These include methods for improving image quality, tracing filament networks, and constructing high-resolution anatomical models. These methods are highly parallel and designed to allow users to segment and visualize structures that are unique to high-throughput microscopy data. The resulting models of large-scale tissue structure provide much more detail than those created using standard imaging and segmentation techniques.

To my parents and brother, who have always pushed for the best.

ACKNOWLEDGMENTS

I would like to thank John for his support and supervision through our wide range of projects. I particularly want to thank him for introducing me to this project and for being a great friend and colleague for several years.

I would also like to thank my committee: Yoonsuck, who has directed my imaging research as the head of the Brain Networks Laboratory and has been exceedingly patient with technology that doesn't always cooperate; Louise for walking me through everything I've ever learned about biology and neuroanatomy; and Don for being the best lecturer I've had and teaching me everything I know about Physically Based Modeling.

I would also like to thank Jaerock, who was behind the KESM automation and user interface and is the best office mate, colleague, and programmer I have had the pleasure of working with.

Finally, I would like to thank Bruce, whose vision and constant support has been with me throughout my graduate career. None of us could have done this work without his dream. He will be greatly missed.

TABLE OF CONTENTS

CHAPTER Page

CHAPTER Page

LIST OF TABLES

LIST OF FIGURES

FIGURE Page

FIGURE Page

FIGURE Page

FIGURE Page

FIGURE Page

FIGURE Page

CHAPTER I

INTRODUCTION

The ability to create large-scale high-resolution models of biological tissue provides an excellent opportunity for expanding our understanding of tissue structure and function. This is particularly important for brain tissue, where the majority of function occurs at the cellular and sub-cellular level. However, reconstructing tissue at sub-cellular resolution is a complex problem that requires new methods for imaging and data analysis.

Current imaging methods have strong limitations, either in the size of the volume of tissue that can be imaged, or the resulting resolution of the captured data sets. Imaging methods designed for large-scale volumes, such as MRI and CT, provide very coarse resolution detail which is sufficient for large anatomical features but insufficient for understanding cellular anatomy. Microscopy methods are frequently used to study cells in two-dimensions while confocal [70] and multi-photon [15] imaging can be used to construct three-dimensional data sets. Unfortunately microscopes are limited to imaging only tissue at or near the specimen surface, preventing the analysis of large-scale three-dimensional anatomy.

This dissertation follows the style of *IEEE Transactions on Visualization and Computer Graphics*.

In this dissertation, I describe a prototype microscopy technique that can image large volumes of tissue at sub-cellular resolution. This method, known as Knife-Edge Scanning Microscopy (KESM), has an extremely high data rate and can capture large tissue samples in a reasonable time frame. We can therefore image complete systems of cells, such as whole small animal organs, in a matter of days. Because KESM has such a large data rate, I refer to this technology, and others currently in development by colleagues, as *high-throughput microscopy*.

The resulting data sets produced using KESM are often several terabytes in size, making them difficult to analyze using standard image processing, segmentation, and visualization algorithms. This is because lengthy imaging times generally dominate the study of tissue samples using standard techniques. Therefore, high-cost algorithms can be used to study standard data sets without a significant impact on the time required for the result. High-throughput microscopy fundamentally changes this framework because the sheer volume of data produced would take an unreasonable time to analyze using standard methods. This problem is exacerbated by the fact that microscopy data sets are often more complex and feature-rich than other large-scale methods such as MRI and CT.

In the second part of this dissertation, I describe algorithms that I have developed to cope with large and complex data sets. These methods are highly parallel and designed to allow users to segment and visualize structures that are unique to high-throughput microscopy data.

1.1. Motivation

The ability to image and reconstruct large volumes of tissue at the microscopic level has many applications, including:

- Creating databases of tissue features

- Constructing high-resolution three-dimensional models
- Understanding connectivity in the brain

The methods discussed in this dissertation provide the imaging and algorithmic groundwork to begin exploring each of these problems.

Creating morphometric databases would allow researchers to compare disease models with control animals. The usefulness of such a database has been shown by the success of the Brain Atlas constructed by the Allen Brain Institute (www.brainmaps.org). While this database allows researchers to explore gene expression patterns in the brain, the resolution is far too low to allow anatomical reconstruction at the sub-cellular level. High-throughput microscopy techniques such as KESM provide significantly higher-resolution data sets, allowing individual cells and their components to be seen.

Constructing high-resolution models of sub-cellular structure and activity can provide insight into nutrient transport through tissue. The structure and function of microvessels is of great interest, particularly in neurodegenerative disease and cancer research. While current microscopy methods can image the small local structure of microvessels, KESM allows the entire network to be reconstructed.

Reverse-engineering the brain has been revealed as one of the National Academy of Engineering's Grand Challenge problems [79]. Confronting this and other such challenges requires the ability to obtain information about the three-dimensional structure and connectivity of neurons across large volumes. Imaging techniques such as KESM can play an important role in understanding complex connections that span the entire brain.

1.2. Overview

The techniques presented in this dissertation describe methods for imaging and analysis of tissue features at the sub-micron scale (0.3μm - 1.0μm) across large volumes of tissue (on the order of several cm^3). In this dissertation, I describe new methods for imaging, modeling, and visualizing complex microscopic structures in large tissue samples. The goal of my research is three fold:

- **Advance microscope technology** in order to support the imaging of large volumes of tissue. These new imaging methods provide data sets with a sampling resolution comparable to or exceeding current standard microscopy methods, such as confocal or multi-photon microscopy, while allowing large three-dimensional volumes to be imaged in a reasonable amount of time.
- **Develop modeling methods** that support large data sets. These include methods for segmenting structures commonly found in microscopy data sets. These segmentations are then used to construct models that can be used to perform morphometry and statistical analysis of biological structures embedded in the tissue.
- **Develop visualization techniques** that can cope with large data sets. More importantly, these visualization methods should help researchers understand the structure of complex microscopic features. In particular, I focus on methods for visualizing filament structures, such as microvessels, and their relationships to the surrounding tissue.

In this chapter, I describe the motivation for this work as well as provided an overview of my methods. In Chapter II I discuss advances that I have made in imaging large data sets in high-resolution. In particular, I will talk about the development of Knife-Edge Scanning Microscopy (KESM) and its usefulness in imaging large-scale tissue specimens at sub-cellular resolution. I will place a particular focus on brain tissue because of the abundance of highly complex features visible in these data sets as well as the relatively limited knowledge we have of the sub-cellular structure of the brain in three-dimensions.

Chapter III describes high-speed image processing algorithms that are used to eliminate noise and other artifacts from KESM images. In particular, I focus on algorithms that are fast, perform minimal processing, and are highly parallel. These features are important when working with large data sets in order to maintain high-throughput capture and storage of the images.

In Chapter IV I focus on fast segmentation algorithms for filament networks, which are fundamental features in microscopy data sets. These structures consists of vast interconnected networks of thin fibers that span large volumes of tissue. Some examples of filament networks include microvascular networks as well as the neuronal networks formed by interconnected axons and dendrites in the brain. These structures are fundamentally important to tissue function and become prominent features in high-throughput microscopy data sets. In fact, high-throughput microscopy methods such as KESM are the only techniques that allow us to reconstruct the three-dimensional structure of these networks. This is because they are composed of very thin components that span large volumes, requiring large-scale high-resolution imaging.

In Chapter V, I discuss segmentation methods for creating biologically accurate models of microvascular networks. These models become particularly important because of our limited understanding of the three-dimensional structure of the microvascular system and provide a means of measuring and comparing these structures.

Finally, in Chapter VI I discuss new methods for visualizing microscopy data. Again, I focus primarily on filament networks which are particularly difficult to visualize because of their complexity and the fact that connected components span large portions of a data set. I describe methods for storing the volumetric data captured using high-throughput microscopy and using the stored data to selectively visualize microvascular components based on selected features from a graph query. I then show how these network components can be related to the surrounding tissue by allowing a user to visualize the relationships between microvessels and the surrounding cells.

The main goal of my work in this dissertation is to provide the basic tools required to analyze and model tissue at the sub-cellular level. Understanding the three-dimensional structure of tissue at this scale is important in understanding tissue function. This is particularly true in biological tissues, such as the brain, where function is closely tied to sub-cellular structure. This work provides advances in both imaging and computational methods that can be used to extend our knowledge of three-dimensional tissue structure at the microscopic scale.

CHAPTER II

KNIFE-EDGE SCANNING MICROSCOPY

The study of spatial distribution, morphology, and interconnection of cells in the nervous system has been a primary focus since the beginning of neuroscience. The study of neural connections started with the development of staining techniques by Camillo Golgi, who used them initially to study interrelationships of neurons and glia in the olfactory bulb [24] and cerebellar cortex [22, 23]. The staining methods developed by Golgi were also used by Santiago Ramon y Cajal, who surmised that the visible ”mesh-like” arrangement of fibers were actually complex networks of independent but interconnected neurons [91].

Reconstructing and modeling the three-dimensional anatomical structure of individual cells in situ is vital to understanding their function in larger units. To create detailed, three-dimensional cellular maps of organisms, we require methods for creating volumetric data sets from blocks of tissue or whole organs. These datasets should also be of sufficient resolution to allow cells and their processes to be identified and traced. Reconstructing thin cellular processes also requires that individual sections be accurately aligned and in proper registration.

There are several well-understood methods for achieving volumetric data sets at the cellular level. Ultramicrotomy and immunohistochemical labeling have given rise to the reconstruction of gene expression in the mouse brain [77]. Although methods have been developed to account for issues that arise in section alignment and warping [33], relatively thick sections (10 to 20 micrometers) are required to sample the dataset in order to complete the process within reasonable time constraints. Confocal light microscopy is a well-known method for extracting high-resolution details from tissues stained with fluorescent dyes [31, 39, 70, 84]. These methods can be used to construct a volumetric dataset [2], however the presence of backscattered light limits these datasets to sampling that takes place on or very near the specimen surface. In addition, the point spread function along the optical axis limits the resolution of optical sectioning. Multi-photon microscopy [15] allows deeper sectioning but still requires sampling relatively near the tissue surface. Extension of multi-photon microscopy to depths exceeding 1mm requires photo-ablation such as All-Optical Histology (AOH) [87]. This technique still uses standard multi-photon microscopy in consecutive tissue regions, therefore the resolution along the imaging axis is still limited by the three-dimensional structure of the point-spread function. In addition to resolution limits, three-dimensional optical methods require the tissue surface to be rasterized using a laser, making these methods far too time consuming for large tissue volumes.

The use of serial block face scanning electron microscopy (SBF-SEM) [14] for gathering three-dimensional volumetric data at sub-cellular resolution provides a method for reconstructing small blocks of tissue. Using an in-chamber ultramicrotome, serial sections (less than 50nm thick) are removed, allowing the block face to be scanned repeatedly by the SEM. Although this sequential sectioning and scanning produces an aligned dataset at sufficient resolution to follow even the smallest diameter cellular processes and resolve cellular organelles, the time required to resolve a single image prohibit it from being used for large volumes of tissue, such as a whole mouse brain.

In this chapter we describe Knife-Edge Scanning Microscopy (KESM), a novel technique using light microscopy for dataset acquisition that allows us to section and image blocks of tissue at the cellular level of detail quickly and automatically. Our imaging techniques allow us to cut and image embedded specimens at a rate of approximately one section every two seconds. These sections are sampled at sub-micron intervals in all three spatial dimensions, maximizing the spatial resolution to allow three-dimensional reconstruction of the entire tissue sampled. The use of serial sectioning allows us to overcome limits in optical resolution along the imaging axis as well as removes the need for complex image processing, such as deconvolution [70].

This technology provides a means of extracting full-scale cellular microstructure from large tissue samples. We have tested KESM using several en bloc staining techniques that can be used to answer many open questions in neuroscience. These include creating full-scale soma maps of the brain, exploring the connectivity of brain microvasculature, and gathering large statistical samples of neuron morphology.

2.1. Methods

Our primary goal is to develop methods for acquiring high-resolution volumetric datasets from biological tissue and organs. To this end, we developed and are using the prototype Knife-Edge Scanning Microscope (KESM). The KESM is designed to act simultaneously as both a microtome and a microscope, to allow sections as thin as 0.5μm to be cut from tissue embedded in plastic. We constructed the instrument from a stacked series of mechanical stages, a knife/collimator assembly, a microscope, and a high-sensitivity line-scan camera (Figure 1). Embedded tissue is sectioned using a diamond knife. Concurrent with cutting the tissue, each section is scanned through the microscope and the subsequent images are compressed and stored for additional processing to yield three-dimensional reconstructions.

2.1.1. KESM Design

The most distinctive feature of our design is the knife and collimator assembly. This assembly provides illumination as well as the means to cut individual sections of tissue (Figure 2). The knife/collimator assembly consists of a removable diamond knife module rigidly mounted to a granite bridge such that the top knife facet is 45° to the vertical and perpendicular to the optical axis of the microscope. When perfectly aligned, the top facet of the knife and the specimen plane of the microscope coincide. The collimator assembly is

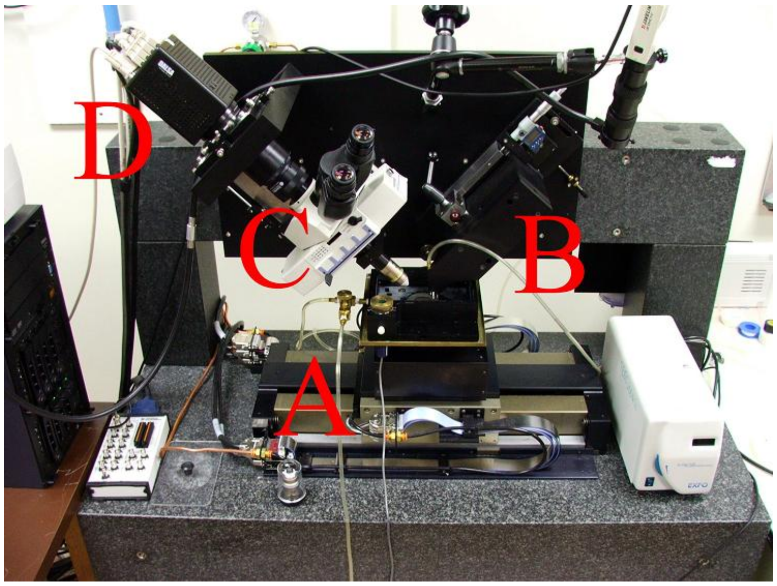


Fig. 1. Knife-Edge Scanning Microscope. The KESM is constructed with three Aerotech high-precision stages (A), a diamond knife and illuminator (B), modified microscope (C), and high-sensitivity line-scan camera (D).

slightly adjustable, allowing rotation around two axes for precision orientation of the knife.

The motion of the tissue block under the stationary diamond knife is controlled by a series of mechanical and air-bearing stages, allowing movement in all three dimensions. The objective is positioned such that the edge of the knife bisects the objective field-of-view. In order to improve the resolution of our images, all cutting is performed under water and uses water-immersion objectives.

In order to maintain the highest possible bandwidth for imaging, specific hardware is devoted to image capture. The high-sensitivity camera sends data directly to a personal computer that acts as a camera server. The digital camera works in conjunction with two frame-capture boards. The twin boards work concurrently to send data from the camera into a 3-gigabyte frame buffer stored in the server's main memory.

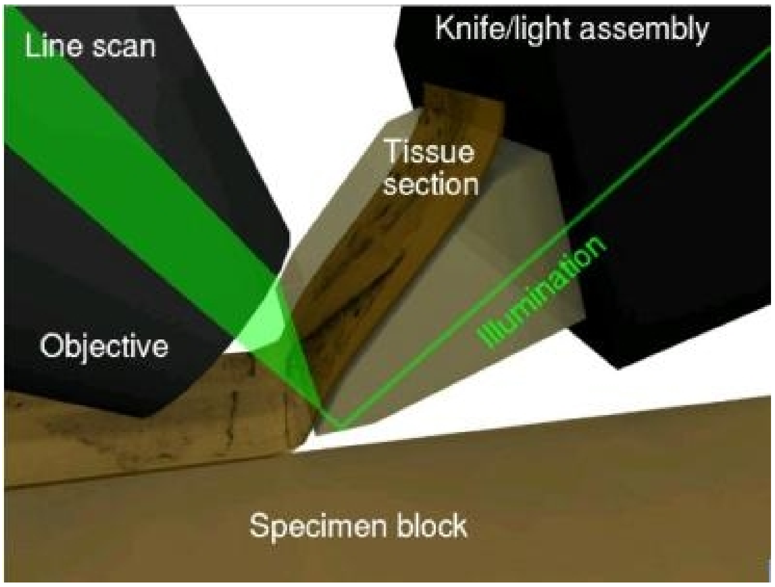


Fig. 2. Objective, knife, and specimen photograph (top) and cartoon (bottom) illustrating light transport through the diamond knife and into the objective for imaging.

2.1.2. KESM Construction

The specimen stage is constructed by stacking three different stages to provide movement along all three axes. The stages are constructed and assembled by Aerotech Inc. according to our specifications. Movement along the cutting axis (x-axis) is performed by an ABL20030 air-bearing stage. This stage provides smooth motion but will oscillate slightly around a fixed position. Because of this, all imaging is performed when the stage is moving at a constant velocity. Movement along the edge of the knife (y-axis) is performed by an ALS130 linear drive stage and movement up and down (z-axis or depth) is provided by an AVL112 vertical lift stage. Both of these latter stages are mechanical, not air-bearing, stages and remain very rigid at a fixed position. All imaging is performed while these axes are stationary so the rigidity is required. Both the x- and y-axes have a feedback resolution of 20nm and the z-axis has a feedback resolution of 25nm. All feedback error is well below the maximum resolution of the objective and therefore any misalignment will be too small to be visible in the images.

Image capture is performed using a DALSA CT-F3 high-sensitivity line-scan camera. The maximum speed of the camera (and thus, the limit of our data acquisition rate) is 44KHz. Since the camera provides a line of 4096 8-bit pixels, our maximum data rate is 180MB/s. Camera triggering is controlled by either an internal clock or an external Transistor-Transistor-Logic (TTL) trigger signal. In our case, the TTL signal is sent from the X-axis stage controller. This allows us to key camera imaging to the stage position and provides two advantages over using an internal trigger. First, the sampling rate could easily be set based on stage position. Secondly, basing camera firing on stage position makes image capture independent of stage velocity. Therefore, any small variance in stage velocity does not adversely affect image quality.

Image capture from the camera is supported by two PIXCI D3X frame capture cards (EPIX Inc.). Both cards are required in order to support the maximum data rate for the camera. To capture and store images, we use the Application Programming Interface (API) provided with the PIXCI cards. This software also allows some preliminary image processing (e.g., formatting) and storage. All stage motion and camera control applications are programmed in C++, allowing automatic control and image capture. The PIXCI API is used to format the images as TIFF files and store them on disk after every section is cut and imaged.

Magnification of the tissue is done with a modified Nikon E600FN brightfield microscope. Two objectives were used to acquire the datasets described in this dissertation: a Nikon Fluor 10X objective with a 0.3 numerical aperture (NA) and a Nikon Fluor 40X objective with a 0.8 NA. Both objectives are designed for water immersion and have a working distance of 2.0mm. The numerical aperture (NA) of each of these objectives allows us to sample at 600nm and 300nm intervals, respectively [10]. A 2.5X coupler is used to match the intermediate field of view of the microscope to the CCD sensor of the line-scan camera.

2.1.3. En Bloc Staining

For Nissl staining, mice were deeply anesthetized using ketamine and xylazine, and then perfused transcardially using 50 mls of room temperature phosphate-buffered saline (pH 7.4), followed by 250 mls of cold 4% phosphate-buffered paraformaldehyde (pH 7.4). The mice were then perfused with 100 mls of a solution of 0.1% thionin dye in deionized water, and the bodies were placed in the refrigerator (4°C) for 24 hours. This is a novel way to introduce stain into the whole mouse brain, since most staining occurs on sectioned tissue. Perfusion of the thionin stain speeds up the staining process, which was previously accomplished in our laboratory entirely through diffusion, taking 2 or more months to complete. In this case, after 24 hours the brains were then removed from the calvaria and placed in a fresh solution of 0.1% thionin and left at 4°C for 7 days. The brains were destained and dehydrated through a graded series of ethanols starting with 50% ethanol and water and increasing to 100% ethanol over a time period of 6 weeks. After three changes of acetone (2-4 days in each solution), the brains were then embedded in araldite plastic following a standard protocol [1], with the exception that each step needed to infiltrate the brains with araldite took 24 hours. KESM sectioning requires that the whole brains be completely dehydrated and infiltrated with araldite plastic. Normal plastic embedding is typically carried out on much smaller pieces of tissue, so we have modified the processing steps to allow us to embed whole mouse brains that we can cut using the KESM.

For Golgi-Cox staining of individual neurons, mice were deeply anesthetized using isoflurane inhalant anesthesia and then decapitated. The brains were quickly removed and placed into a Golgi-Cox fixation solution containing, 1% potassium chromate, 1% potassium dichromate and 1% mercuric chloride in deionized water. The brains were left in this solution, in the dark, at room temperature for 10-16 weeks. The brains were then rinsed in deionized water overnight in the dark, and placed in a 5% ammonium hydroxide solution in deionized water for 7 to 10 days, in the dark and at room temperature. The brains were then rinsed again in deionized water at room temperature for 4 hours and then dehydrated through a graded series of alcohols and embedded in araldite using the same protocol and times described previously for Nissl-stained mouse brains.

2.2. Results

In this chapter we demonstrate the use of KESM to section and image three datasets. Since each dataset can consist of thousands of sections, we provide examples of individual sections as well as images showing cross sections of each 3D dataset. The first dataset is mouse brainstem and spinal cord that was stained en bloc with the Nissl stain, thionin. This dataset was cut into 1μm thick sections and the tissue sections were imaged at a 0.6μm resolution (Figure 3 and Figure 4). The sampling resolution used by the KESM is large enough in all three dimensions to allow full reconstruction of cell soma in the brain. In addition, the structure of the cell soma can provide some information about the type and orientation of the cell. With the development of segmentation algorithms designed to parse these complex data sets, this opens the door to constructing complete maps of cells containing cell type, position, and orientation. This would provide a high-resolution atlas to which other data sets can be mapped and related. The speed at which KESM data sets can be imaged also allows biologists to counterstain structures of interest with Nissl. These counter-stained structures can then be put in context by mapping the volume (using the Nissl-stained cells as fiducials) to the whole-brain map.

We provide an example of this in the second dataset, where we followed a similar Nissl staining process and introduced India ink in order to enhance the contrast of the vasculature. We then sectioned the entire mouse olfactory bulb (Figure 5 and Figure 6). Again, the

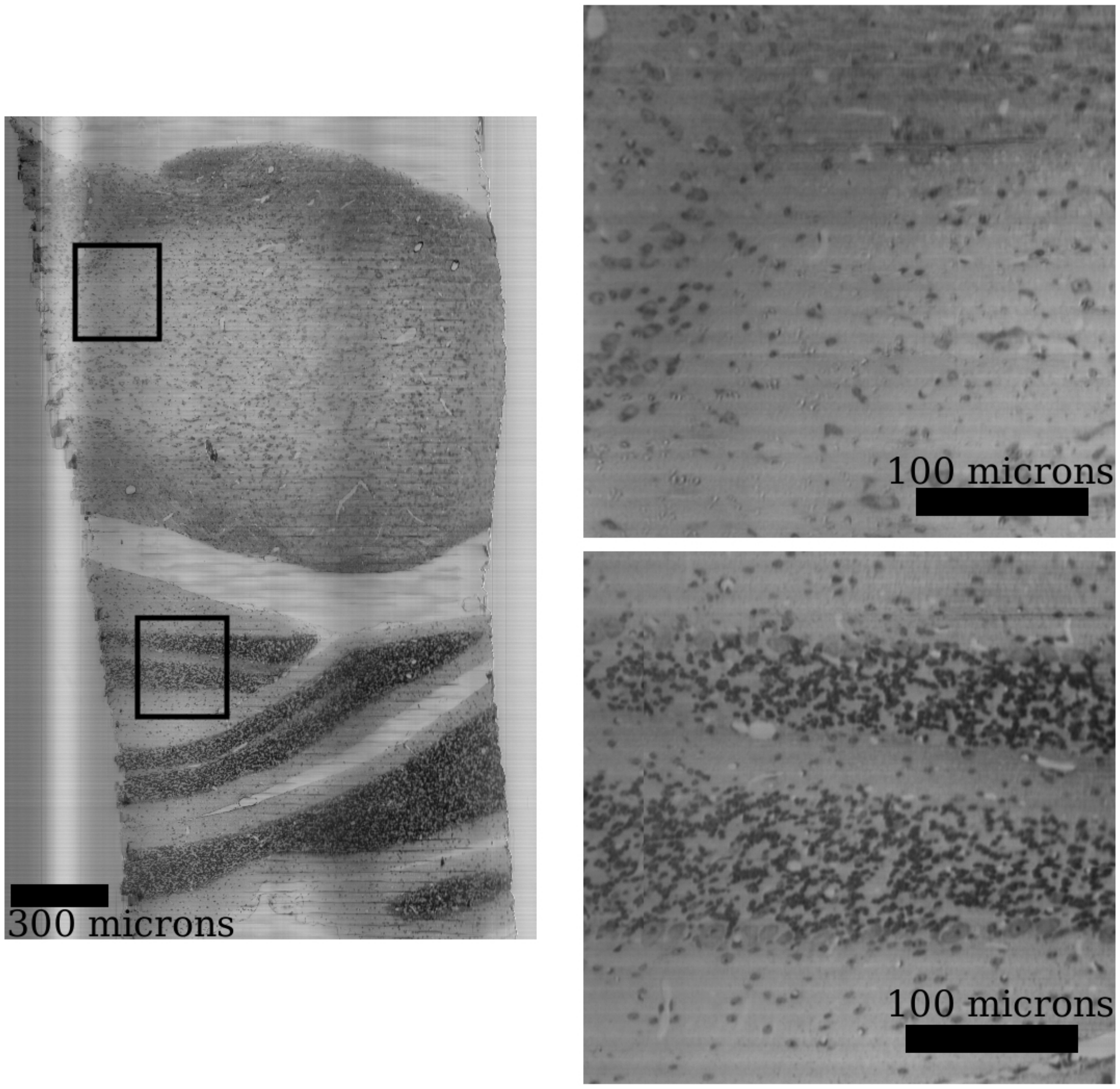


Fig. 3. Coronal section of mouse brain stem. The complete section is shown (left) with two close-up inserts (right). The overall resolution of the image is 0.6μm/pixel and the section thickness is 1μm.

sampling resolution is high enough to image even the smallest capillaries that make up the brain microvasculature. Analysis of these images allows us to study the relationship between brain microvasculature and the cells in the olfactory bulb.

Finally, we stained a mouse brain en bloc using a slight modification of the Golgi-Cox method (see Section 2.1) and scanned several thousand sections of olfactory bulb and cortex to demonstrate fine alignment of sections and the ability to segment fine fibers (Figure 7 and Figure 8). For the Golgi stained brain we used 0.5m thick sections and imaged with a 40X objective, which provided a sample resolution of 0.3μm. Even when using a high-resolution objective, Golgi-stained tissue can be difficult to resolve using standard techniques because of the low resolution along the imaging axis. When thin sections are taken, each section provides less information on its own, however we are able to extract more detailed spatial information for the fiber in three-dimensions. The use of Golgi-Cox allows a more comprehensive study of the morphology of cells in the nervous system. Although the percentage of cells stained is relatively small, the sample size is quite high when compared to the small regions that can be imaged using optical sectioning. This provides a much larger statistical sample from which to characterize cell morphology. In addition, the cells are imaged ”in context”, making their relationships to other cells apparent. This is important in neural simulations and provides a broader base for the stochastic generation of cells and connectivity in computational models.

2.2.1. Reconstruction

Given a three-dimensional dataset, standard techniques exist for surface extraction and visualization. The alignment of consecutive sections is such that contours can be created using software packages such as Amira for visualization of vasculature and neurons (Figure 9). Generally, some processing is required in order to remove or reduce lighting artifacts, however these result in little to no data loss and can be applied as the images are stored.

2.3. Discussion

Imaging of biological tissues and organs at a cellular level is an emergent technology, which promises to elucidate mammalian brain anatomy in unprecedented detail and also has the potential to elucidate cellular organization in other organs both normal and those with pathology related to a wide range of disorders. We have shown how our techniques can help to reveal the density, morphology, and interconnectedness of neurons in the rodent brain. Our KESM instrument makes possible the three-dimensional microscopy of large biological specimens the size of an entire adult mouse brain. The instrument has been designed to volume digitize a specimen (e.g., a plastic-embedded tissue block) at a maximum sampling resolution of 160nm in the image plane at rates up to 200Mpixels/s. Depth resolution in this case is typically 500nm using a 40X objective, and 1000nm using a 10X objective. Our initial results on brain tissue show the feasibility of the KESM method to gather large, high-quality light-microscope datasets of whole tissue / organ samples. The speed of the KESM is such that an entire embedded mouse brain (1 cubic cm) can be scanned in one month of one-shift operation, yielding an uncompressed volume dataset of 15 terabytes (15,000 gigabytes). Although our primary research interests are focused on the central nervous system, colleagues have expressed interest in other organ samples including liver, kidney, heart, and lung.

One current limitation of KESM is the lack of significant research on en bloc staining techniques. We have developed novel methods for staining brain tissue with both Golgi and Nissl stains; however these stains can supply only limited information. Clearly other absorption light microscopy stains are of interest. Transgenic animals should provide a good solution to the problem of en bloc staining, but would require some technological modifications to our current techniques. In particular, since many transgenic animals are engineered to express fluorescent proteins, we need to modify our current system to allow for fluorescence imaging, and this modification is currently underway.

The KESM can be classified as a small-scale planing machine, which uses a single-point cutting tool. An important obstacle to obtaining high-quality data is self-excited oscillations (chatter) generated during the cutting process, when sequential sections are being removed from the sample (McCormick, 2005). As is visible in some of our sample images, chatter continues to be a problem both in the time it takes to filter it out and the overall quality of the final dataset. Altering the cutting velocity for each pass has significantly suppressed the chatter, as described above. Modifications to the instrument to produce a chatter-free operation by increasing the rigidity of the cutting tool also have helped significantly to reduce chatter. In addition, development of en bloc staining and embedding methods yielding specimens that are embedded in harder plastic resins also have helped to reduce the problem of chatter. The exact explanation of the chatter phenomenon encountered with the KESM and its suppression has yet to be completely elucidated. Extensive modeling of the cutting process, including the study of chatter and chatter suppression has been conducted in support of the (metal) machining industry, but the precision sectioning of plastic at sub-micro thickness has hitherto attracted virtually no attention from the mechanical engineering community. Traditionally this group has focused on the work piece geometry and surface quality, and has ignored the tissue ribbon (”chip”) of interest in physical sectioning/imaging of tissue.

The image acquisition speed (approximately 7mm^2/sec using a 10X objective) provided by the KESM as well as the image resolution (600nm/pixel using a 10X objective) can allow us to create full-scale maps of entire organs in a reasonable time frame. Although we are currently limited to tissue volumes the width of the field of view of the objective (2.5mm at 10X), we are developing cutting and alignment techniques that can be used to combine adjacent volumes and thereby eliminate constraints on specimen size.

Alzheimer’s disease (AD), the most common cause of dementia in the elderly, is a complex central nervous system (CNS) disorder characterized by progressive loss of cognitive abilities. Pathological hallmarks of AD are: extracellular amyloid plaques, intracellular neurofibrillary tangles and neuronal cell death [**?**]. Current research in the pathogenesis of AD has implicated various components of the neurovascular unit, which contribute to the dynamic regulation of microvascular permeability. Specifically, atherosclerosis, degeneration of endothelial cells and decreased microvascular density are vascular factors that play a role in the pathogenesis of AD [28,51]. Cerebral hypoperfusion is a major clinical finding in AD and likely plays a critical role in the pathogenesis of AD11. Vascular endothelial growth factor (VEGF) is protein that promotes the proliferation and survival of vascular endothelial cells and enhanced VEGF immuno-reactivity has been observed in the neocortex of subjects with AD as compared to elderly controls. Ultimately, the critical pathological feature in AD is significant loss of cortical cholinergic neurons, which leads to memory loss and dementia in affected individuals. Numerous reports link AD pathology to the presence of extracellular amyloid deposition, and to the presence of pathologic or altogether missing microvessels, and to the presence of increased VEGF activity [92]. It has been suggested that all of these changes may result in perivascular cholinergic neuron loss that is causal in the cognitive decline of AD [34]. However, to date, no one has documented specific co-localization of pathologic or missing microvasculature, VEGF, amyloid, and missing cholinergic neurons in the AD neocortex at the cellular level. We propose that the KESM is the ideal instrument to carry out the necessary three dimensional studies needed to elucidate the complex localization of multiple components involved in AD pathology.

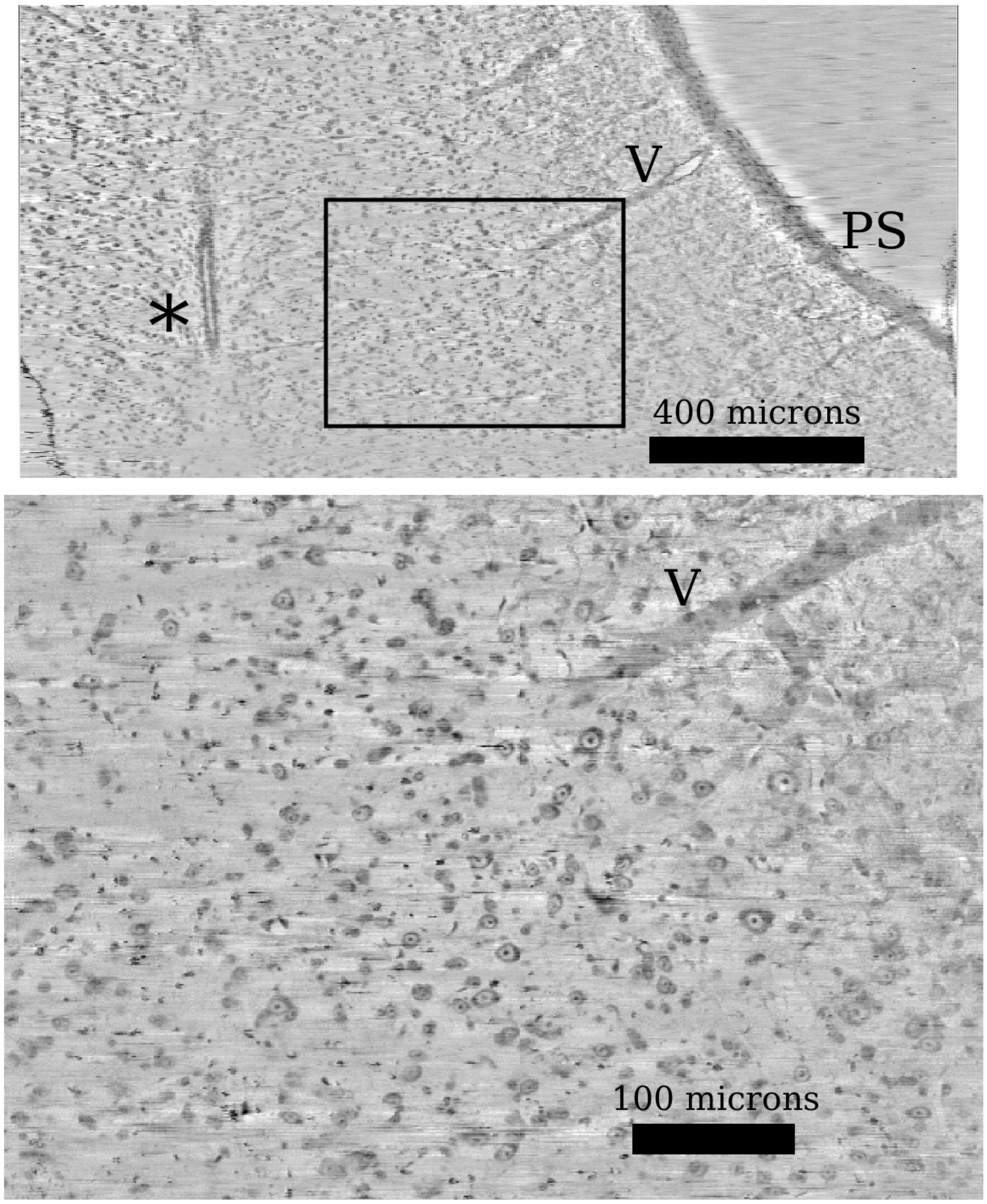


Fig. 4. Re-sectioning of the data set constructed using sections like those in Figure 3. A large cross-section of the data set (mouse brain stem and spinal cord) is shown (top) as well as a close-up (bottom). The section thickness (resolution along the vertical axis) is 1μm. The labels indicate the pial surface (PS), central canal (*) and a blood vessel (V).

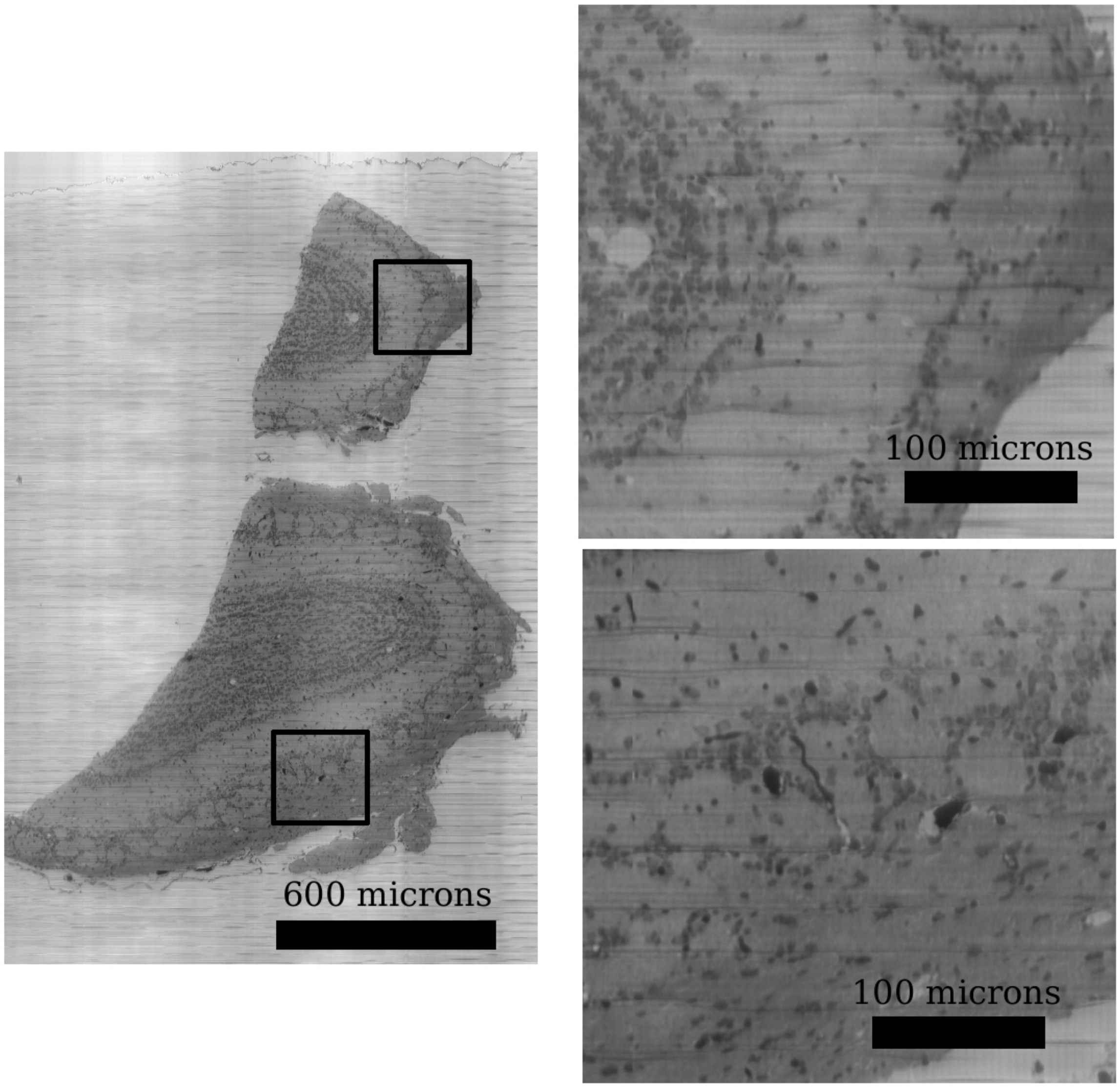


Fig. 5. Coronal section of the mouse olfactory bulb. The complete section is shown (left) along with two close-up inserts (right). The tissue is stained with Nissl along with perfusion of India ink through the vascular system. The optical resolution of the image is 0.6μm/pixel and the section is 1μm thick.

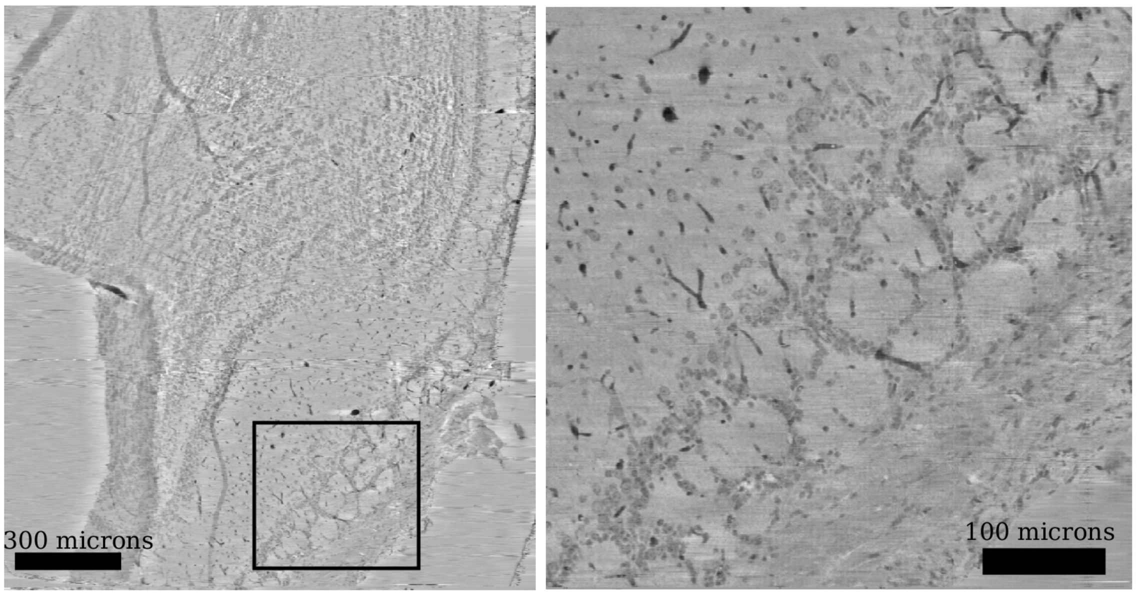


Fig. 6. Sagittal re-sectioning of the India-ink and Nissl stained olfactory bulb shown in Figure 5. The sections are 1μm thick (giving 1μm resolution along the vertical axis).

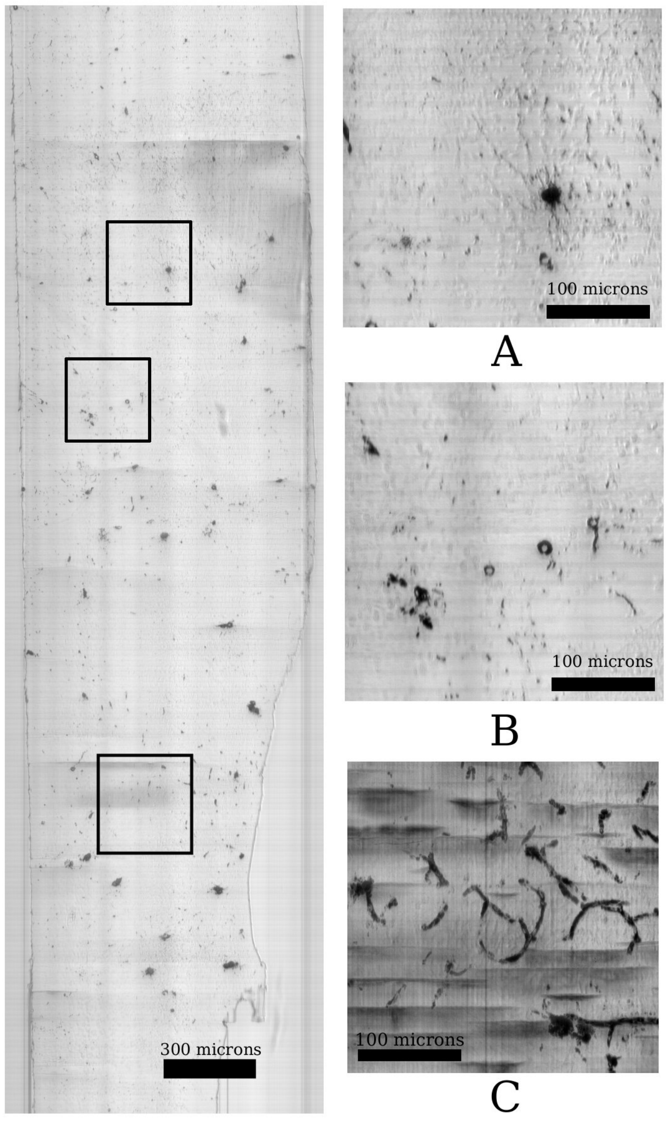


Fig. 7. Coronal sections of Golgi-stained mouse cortex. The optical resolution along the x and y axes is 0.3μm/pixel and the section thickness is 0.5μm. The complete section is shown (left) along with two close-up inserts (A and B). Because Golgi-stained tissue is sparse, small cross-sections of the fibers are generally only visible in a single section. We have shown a close-up (C) created by combining 20 consecutive sections from the associated region into a single image to show the fiber structure of the cell processes.

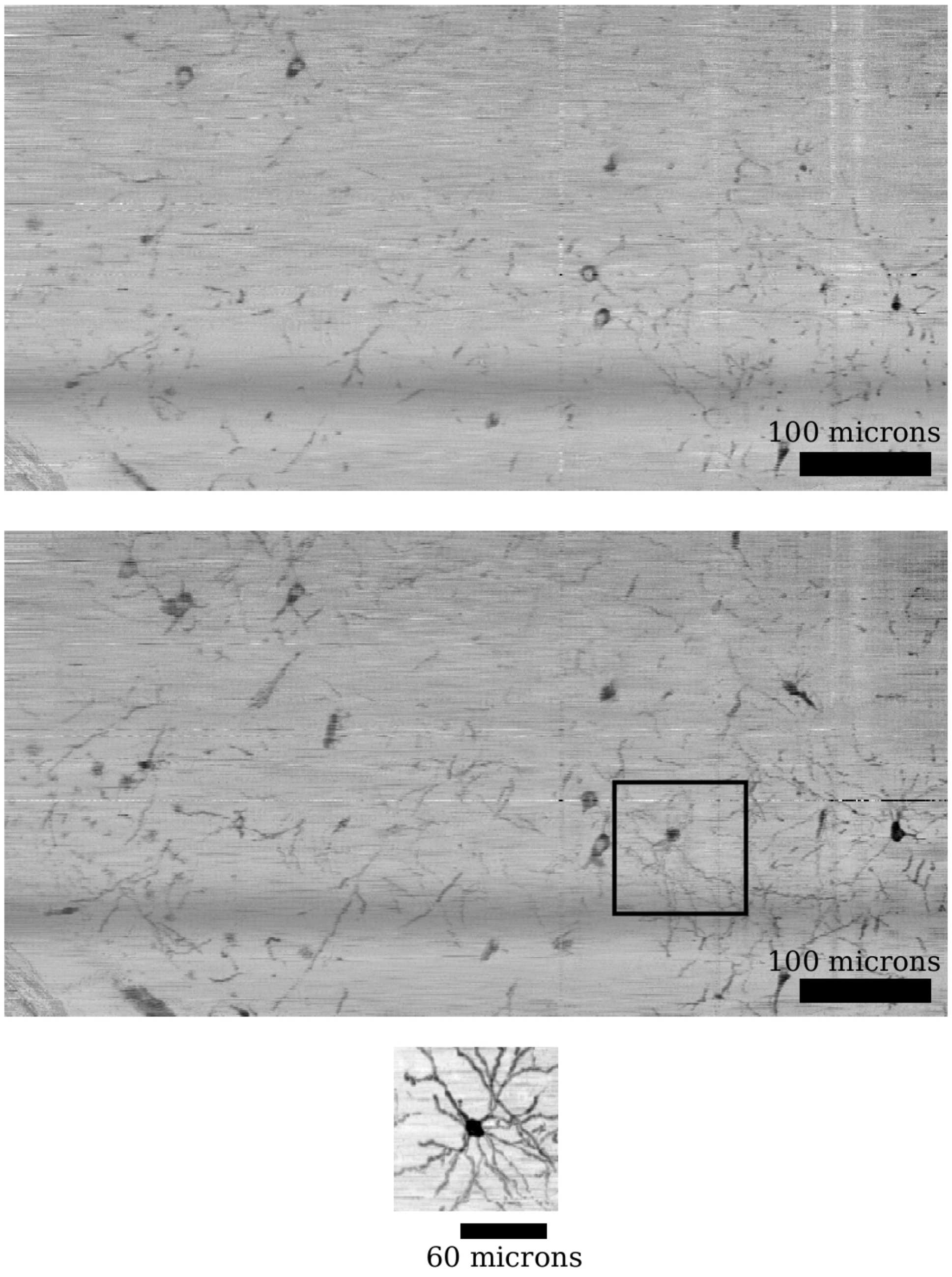


Fig. 8. Sagittal re-sectioning of the data set shown in Figure 7. A single slice is shown (top). Because Golgi-stained tissue contains sparse data, fibers can be more easily seen by combining several slices (middle). A single neuron is shown by combining 300 slices (bottom). The section thickness is 0.5μm (vertical axis).

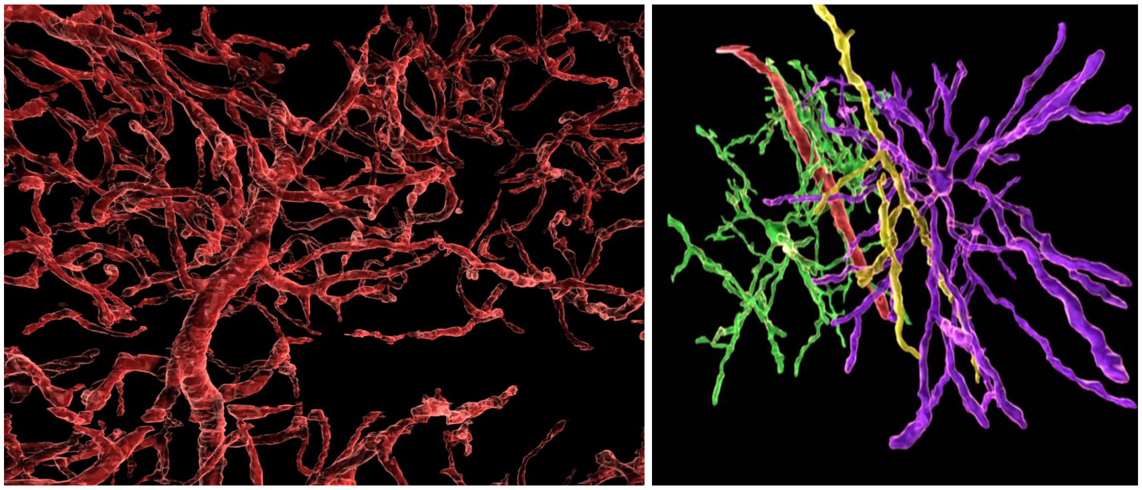


Fig. 9. Surface reconstructions of blood vessels (left) from the mouse olfactory bulb (Figure 5 and 6) and cells stained with Golgi-Cox (right) from mouse cortex (Figure 7 and 8). The neuron labeled in purple (with neighboring axon labeled in yellow) are also shown in Figure 8 (bottom). Nearby astrocytes (green) are shown supporting a blood vessel (red).

CHAPTER III

IMAGE PROCESSING

Captured KESM data sets are subject to various forms of noise and other imaging artifacts. While some of these are common to other forms of microscopy, most the artifacts that we consider are unique to KESM. In this chapter we will present methods for processing KESM images in order to remove imaging artifacts, thus creating more useful volumetric datasets for subsequent analysis.

3.1. KESM Volume Noise

KESM exhibits several types of noise, generally involving small lighting irregularities due to knife shape, illumination frequency, and knife vibration during the cutting process.

3.1.1. Lighting Defects

The most visible lighting defect is a variation in illumination across the *x*-axis. There are two main sources:

- Inconsistent illumination across the surface of the knife. This results in a steady change in the overall illumination along the *x*-axis of the image.
- Defects (e.g. chips and rough areas) on the surface of the knife edge itself. These defects cause refraction and reflection variations at the knife surface, also resulting in uneven lighting across the knife edge. This is visible in images as brighter or darker strips along the *x*-axis and extending along the *y*-axis (Figure 10).

Regular lighting defects are also visible along the y-axis. The major source is fluctuations in the illumination over time. This produces an oscillating fluctuation in illumination creating visible stripes in the image. Depending on the sampling rate of the image, the

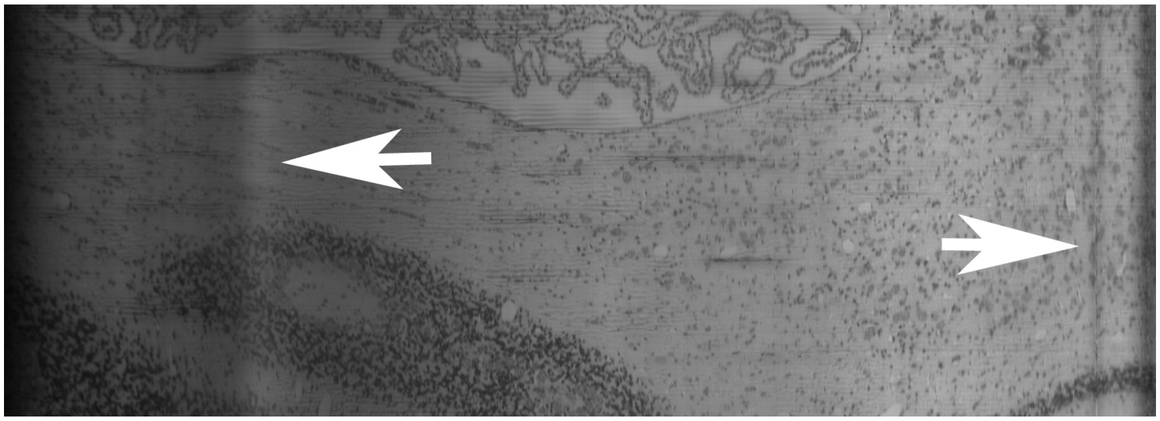


Fig. 10. The overall intensity shift along the x-axis is due to knife misalignment while two knife defects leave visible streaks in the image (arrows).

frequency of the fluctuations changes in the image space, however they are constant over the time domain.

3.1.2. Knife Chatter

Lighting defects such as those listed above form regular patterns in the output data that are easily removed from any given image. Unfortunately, irregular illumination tends to occur in KESM images due to knife vibration. Knife chatter is well known in machining [88], but usually is not an issue in imaging, since other imaging techniques do not image data as it is being cut. We use several mechanical techniques to reduce knife vibration, including increasing the mechanical stiffness of the specimen and cutting tool and randomizing cutting velocities in order to prevent reinforcement at any frequency. Even with these precautions, knife vibration is visible in the resulting image as changes in illumination along the y-axis (Figure 11).

When using KESM it is important to sample as close to the edge of the knife as possible in order to get the best alignment between sections as well as to ensure that the section is coherent and not warped or torn due to water current or knife friction. Sampling at the very

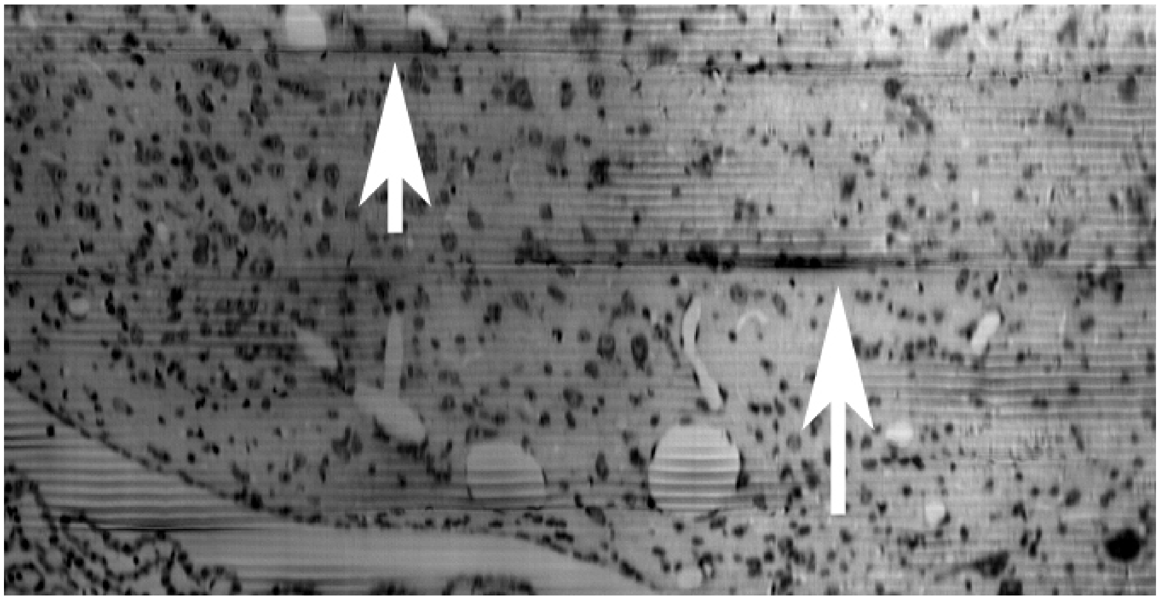


Fig. 11. High frequency chatter is visible throughout the image and particularly bad cases (arrows) may cause loss of information.

tip of the knife, however, makes knife vibration more visible since light intensity quickly drops past the edge of the knife. Knife vibration is therefore visible as dark horizontal stripes across the image which, due to misalignments described above, may or may not be continuous along the x-axis. In addition, particularly severe occurrences of chatter can cause loss of data.

3.2. KESM Image Processing Techniques

We have assembled several known imaging algorithms to help remove lighting irregularities. New methods are also demonstrated that take advantage of the unique noise found in KESM images in order to better prepare the volumetric datasets for segmentation. In addition, we specifically focus on image processing algorithms that require only local information in an image so that processing can be distributed across several systems for faster results. Such parallelism is important for maintaining the high data rate that gives KESM an advantage over other forms of microscopy.

3.2.1. Light Normalization

Equalizing the illumination across the x-axis of the image is the first step in removing noise. Since these irregularities are due to misalignment of the camera or artifacts in the lighting pipeline, they are repeated over every sample in every image. For example, if an artifact present in the image is due to a defect in the knife (such as a chip in the knife surface), this artifact will be present in every sample at a given x value in the image. This type of artifact is also visible even when tissue is not being sectioned. The constant presence of this type of artifact allows a base sample without tissue to be imaged at any time during the cutting process. Since these variations of lighting are constant during the sectioning process, each sample taken along the y-axis can be normalized using this base light vector. Note that such base information might need to be taken at regular intervals, since a knife might gradually develop surface defects, or gradually move out of alignment.

Cyclical illumination artifacts produced by high-frequency noise from the light source are less consistent between images. Since the initial sample can take place at any point during the illuminator's frequency fluctuation, a phase shift is observed along the y-axis. We compensate for this by sectioning slightly less tissue than the knife and objective allows. This leaves a small portion of the knife edge visible without any interfering tissue. A sample line of pixels along the edge of the image can then be used to normalize light frequency fluctuations along the y-axis.

3.2.2. Removing Chatter Artifacts

Knife chatter artifacts are more difficult to remove from KESM images. Although they are lines that extend along the x-axis, they tend to be discontinuous and not perfectly (though nearly) horizontal. This precludes scaling the line by any given scalar value, such as the mean of the current sample. Local smoothing is a method often used to eliminate high-frequency noise [25]. However, it is difficult to apply these techniques to KESM data since most of the information is high frequency and low contrast. Instead, we scale the value of a pixel in the sample by the mean of a small window of pixels surrounding the current pixel in the sample. As this window moves across the image, it removes streaks caused by tissue folding and knife vibration by scaling the intensity values up to match other pixels in the sample. This preserves the high-frequency details, providing that there was very little data lost due to the intensity shift, and removes the streaks making segmentation a simpler matter.

3.3. Results

We have applied these techniques to many different KESM data sets and have come to rely on them as a pre-processing step before segmentation. Here we present these methods for processing Nissl-stained mouse brain stem and cerebellum. This dataset was selected because of the high frequency chatter in the cerebellum and low contrast in the spinal cord (Figure 12 - 14).

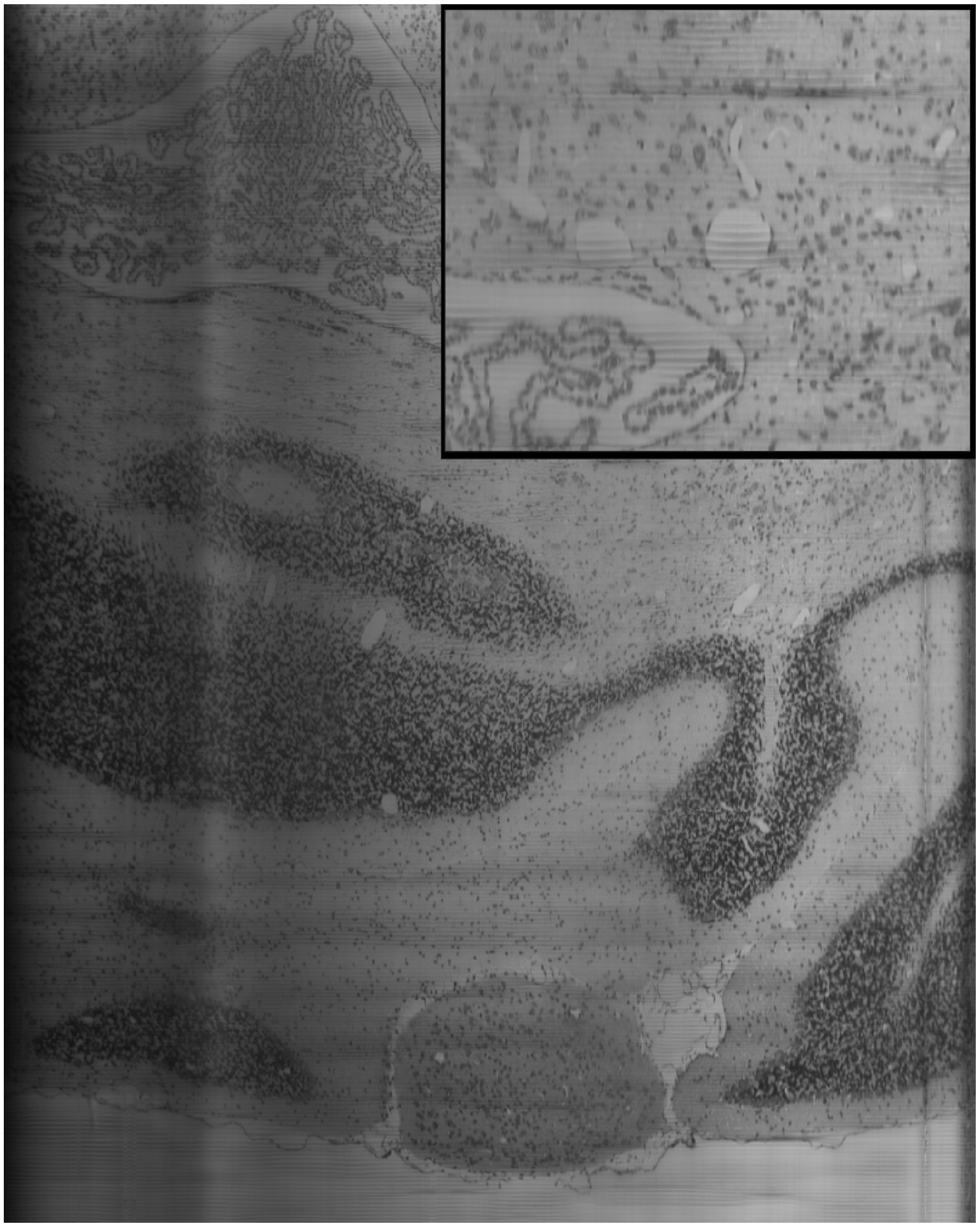


Fig. 12. Original mouse brain stem section with close-up.

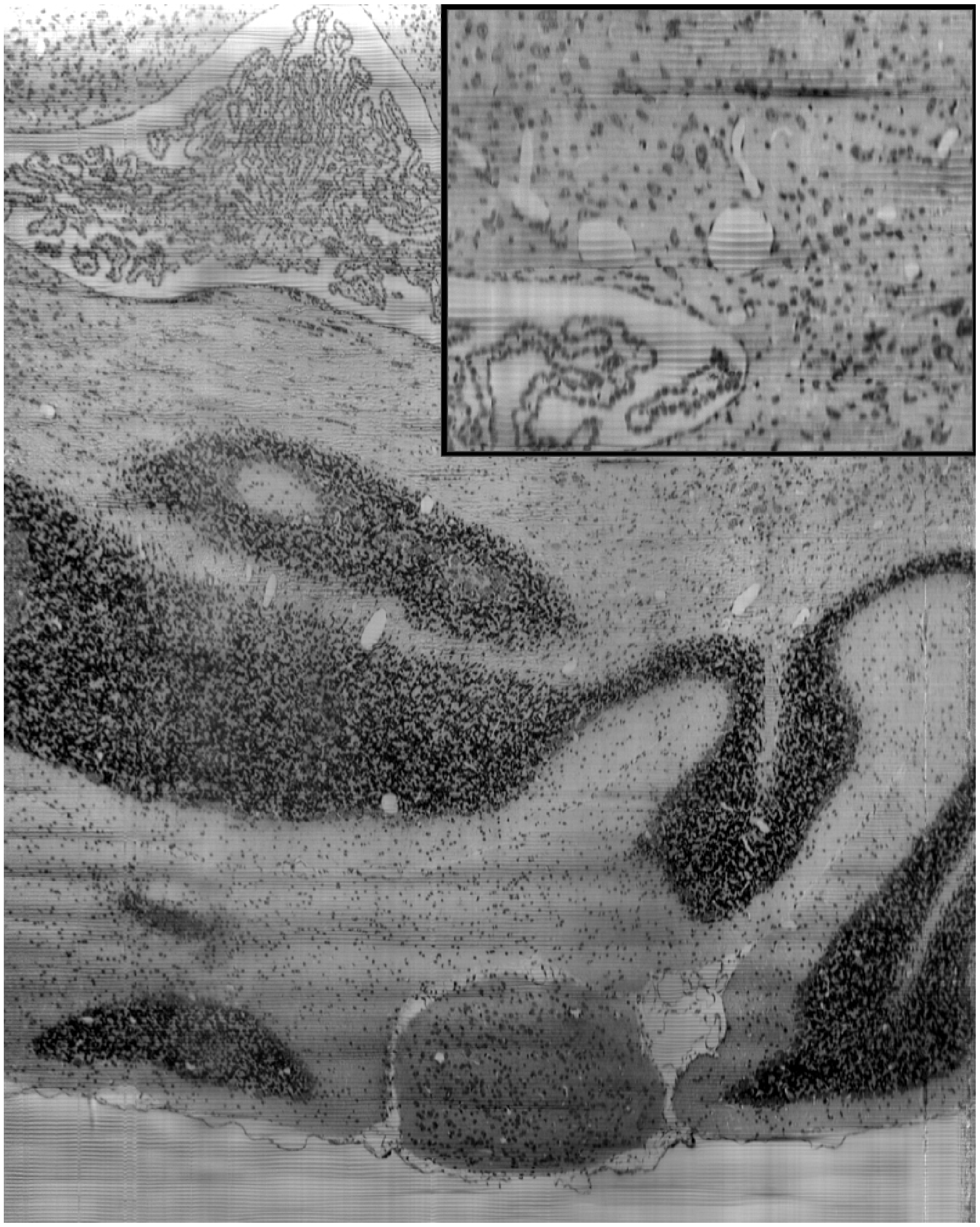


Fig. 13. Removal of lighting and knife irregularities.

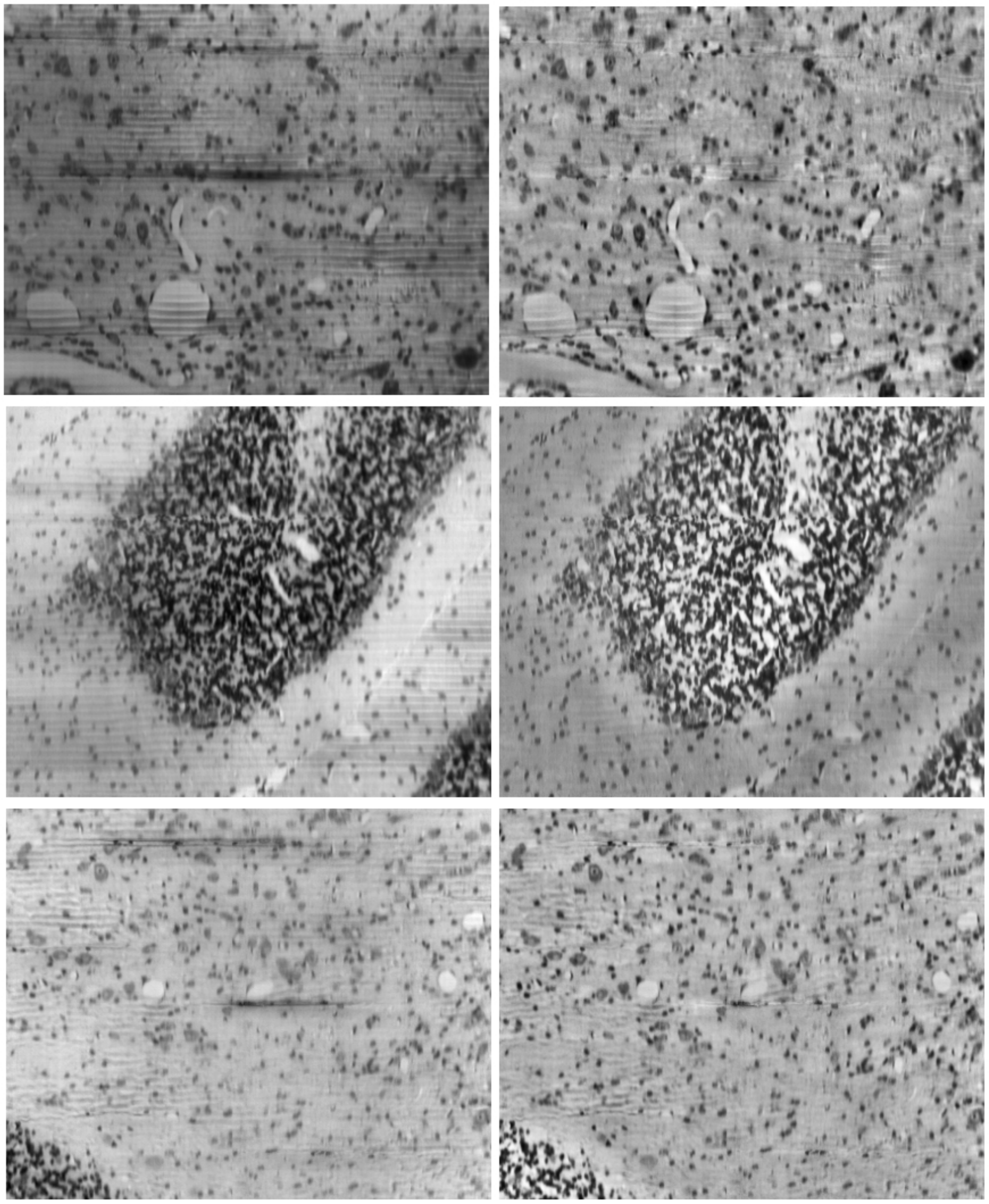


Fig. 14. Final pass for the reduction of knife chatter. High resolution views are taken from the same image as Figure 13. Images are shown after removal of light artifacts (left) and after removal of chatter (right).

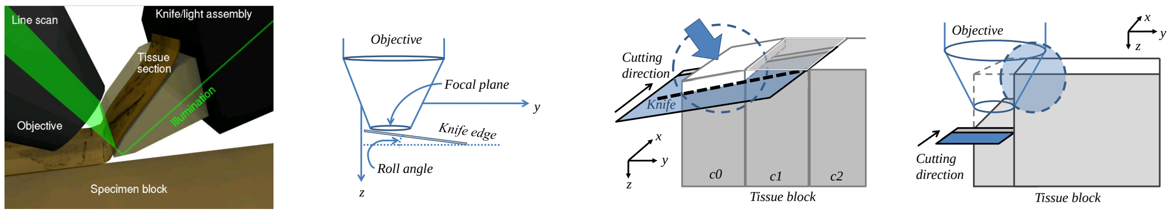
3.4. Block Alignment

Since the tissue in KESM imaging is destroyed during the imaging process, any tissue outside the field of view (FOV) is permanently lost, thereby limiting the section width that can be imaged using KESM.

In this chapter, we describe a technique for overcoming this constraint. We make two major contributions:

- We describe an automated sectioning algorithm that allows neighboring tissue, outside the FOV of the objective, to be imaged in later passes.
- We justify the use of simple affine transformations that compensate for deformations in the captured data set due to our sectioning technique.

3.4.1. Lateral Sectioning



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

Fig. 15. KESM lateral sectioning problems. (a) KESM performs imaging while cutting. (b) Small errors in the roll of the knife can (c) cause damage to un-imaged tissue. (d) Taking several consecutive sections stop tissue damage but may cause the objective to come into contact with the tissue.

In order to image larger specimens, we perform *lateral sectioning* across the specimen surface. In traditional KESM, two motor stages are used during the cutting process. One vertical lift stage is used to adjust the level of the specimen relative to the knife. The second stage moves the specimen under the knife to perform cutting. Our lateral sectioning technique requires the use of an additional stage that moves orthogonal to both of these axes. In our current KESM setup, this stage is already present and used for accurate positioning of the specimen under the knife.

There are two ways to perform lateral sectioning, however care must be taken so that un-imaged tissue is not damaged.

- Lateral sections can be taken across the surface of the knife. Due to small misalignments in knife *roll*, this can cause damage to un-imaged tissue outside the FOV of the objective (Figure 15b and c).
- This damage can be prevented by sectioning an entire *column* at a time. The distance between the knife edge and the objective limits the cutting depth because the side of the objective will come into contact with the un-cut specimen block (Figure 15d). In addition, as the cutting depth increases, there is more contact between the edge of the knife and the neighboring tissue. This can cause tearing of the tissue and induce knife vibrations [88].

3.4.1.1. Stair-Step Sectioning

We use a simple cutting algorithm that avoids both of these problems by cutting small stacks of images in a stair-step fashion (Figure 16). In order to avoid damage to the tissue or microscope, we must ensure that the cutting depth d is:

- large enough so that the overhanging part of the knife does not cause damage to un-imaged tissue
- small enough so that the microscope objective does not make contact with un-cut tissue

- small enough to keep knife vibrations and tissue tearing to a minimum

The first two constraints are easy to achieve because the focal distances of many optical objectives are 1mm to 2mm. We do, however, choose to minimize the column depth in order to minimize knife vibrations due to contact between the tissue and the knife.

The degree of knife misalignment is difficult to accurately measure. We can determine an upper bound for the knife misalignment based on the imaging constraints of the KESM. In order for the entire tissue section to be in focus, both points of the knife edge within the FOV of the objective must be within the focal depth. This means that the angle of the knife θ is constrained by:

$$\theta \leq \arctan\left(\frac{FD}{FOV}\right) \tag{3.1}$$

where FD is the focal depth of the objective and FOV is the field-of-view. If the roll angle of the knife is greater than θ, this can be detected by the user because part of the image will be out of focus. If θ complies with the above constraint, the angle of the knife cannot be detected using the imaging hardware available in KESM. Therefore, we compute a worst-case depth d based on the maximum possible un-detectible value of θ:

$$d = L_k \sin\theta_{max} - FD \tag{3.2}$$

where L_k is the length of the knife (Figure 17) and $\theta_{max} = \arctan\left(\frac{FD}{FOV}\right)$.

3.4.1.2. Implementation

We implement stair-step sectioning by maintaining a height field of the tissue surface. We can then constrain cutting so that the height difference between two columns never exceeds the calculated value d from Equation 3.2. Changes to the cutting parameters (e.g. column thickness) can be handled robustly by simply resampling the height field in order to insure that there is no loss in data. The algorithm used to constrain the sectioning process is shown

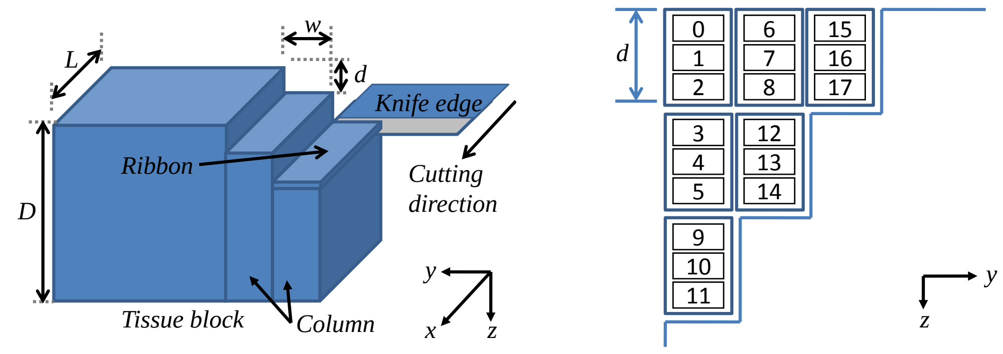


Fig. 16. Sections are cut in a stair-step fashion (left) and ordered so that neither the objective nor the knife comes into unwanted contact with the tissue (right).

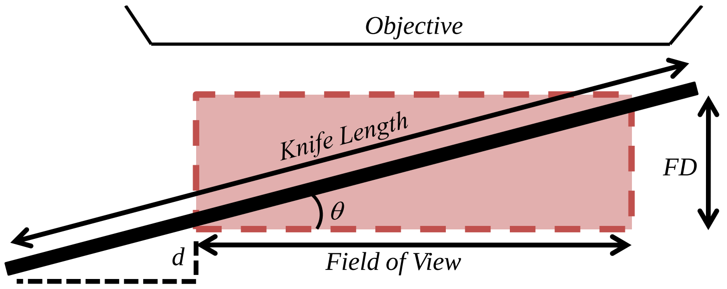


Fig. 17. Misalignment of the knife relative to the focal plane. θ represents the maximum undetectable misalignment. In practice, θ is quite small resulting in $d < 3\mu m$.

in Algorithm 1.

3.4.1.3. Tissue Damage

During the sectioning process, some tearing occurs at the interface of two neighboring columns (Figure 18). Our sectioning experience tells us that the damage is less than $5\mu m$ in width in most cases. For our 10X (0.3 NA) objective, the horizontal pixel resolution is $0.6\mu m$ and the FOV is 2.5mm. This results in an average data loss of 0.2%.

Algorithm 1 Stair-step sectioning

```
$nNumOfCols \Leftarrow (int)(dTotalBlockWidth/dColWidth)$
$nCurCol \Leftarrow 0$
Initialize $z$ positions in all columns
while $nCurColZ < nMaxBlockDepth$ do
    for $nIndexRibbon$ to $nPlankDepth$ do
        Section a tissue ribbon
    end for
    $nPlankThickness \Leftarrow nRibbonThickness \times PlankDepth$
    if $nCurColZ > (nNextColZ + nPlankThickness \times 2)$ then
        $nCurCol \Leftarrow nCurCol + 1$
    else
        $nCurCol \Leftarrow nCurCol - 1$
    end if
    Update $nCurColZ$
end while
```

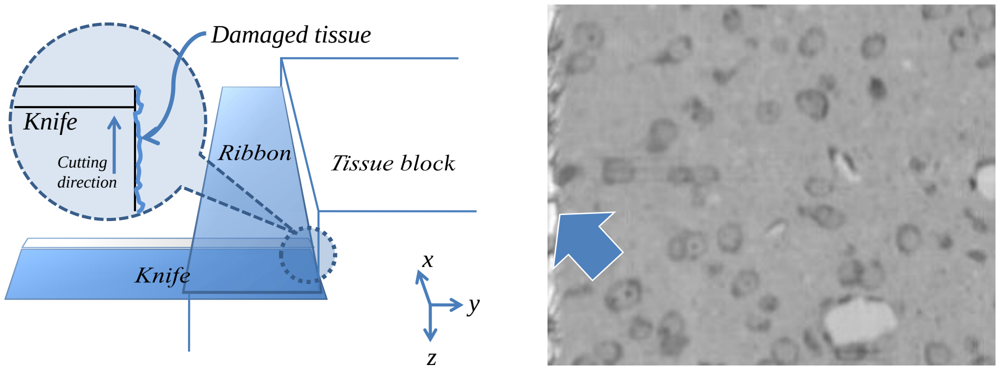


Fig. 18. Tissue damage due to lateral sectioning. (left) Damage occurs due to tearing at the interface of two columns. (right) This results in some data loss at the edge of the image (arrow).

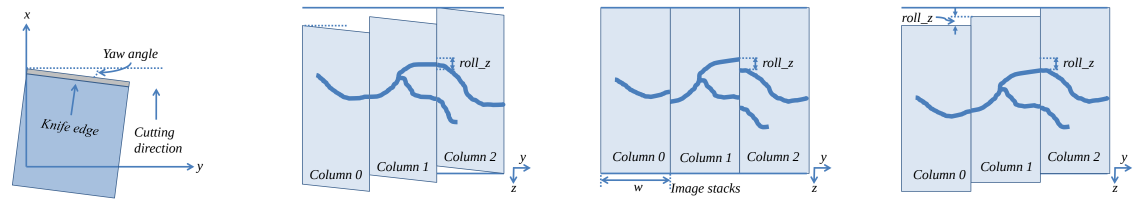
3.4.1.4. Distortion

As mentioned in the previous sections, small misalignments in the knife orientation are difficult to detect and result in the knife contacting the specimen surface at a slight angle θ. Although the roll angle of the knife is the only misalignment that can cause unwanted tissue damage, note that it is also possible for there to be a slight yaw misalignment (Figure 19a). Both of these knife angles cause each column to be imaged at a slight skew (Figure 19b).

When the image stacks are placed next to each other, this results in misalignment at the interface between columns (Figure 19c). We fix this misalignment by applying a translation to each column to compensate (Figure 19d). The direction of translation parallel to the plane created by the column interface. Each component of the translation (x and z) is proportional to the angle of misalignment. We note two important properties of these offsets:

- The offsets are based on knife misalignment along two axes. These angles are too small to be measured with the KESM optics but can produce noticeable distortion in the data set.
- Since the offsets are based on the knife angle, they are constant throughout the entire data set.
- Practical constraints on the knife angle (discussed above) limit these offsets to very small values (1-4 pixels).

In our initial experiments, we attempted to determine these offsets automatically by aligning images representing the interface between the columns. The major difficulty with this approach is that tissue tearing, and therefore data loss, occur at this interface. Although it is possible to acquire image data slightly ”inward” of the interface, this sampling was too coarse to allow effective alignment based on image data alone.



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

Fig. 19. Knife misalignments cause distortions. (a) Errors in knife yaw can also effect the data. (b) Both yaw and roll cause each column to be slightly skewed. (c) Aligning image data shows the misalignment between columns. (d) We can align neighboring columns by applying a translation along the interface plane.

Since the knife misalignments are constant throughput a data set, we found that the offsets need to be determined manually only once. This was done by selecting a small volume of tissue at the interface of two neighboring columns. These volumes were aligned by using filament structures, such as vasculature and neuronal processes, as fiducials. Many of the larger filaments have trajectories that can be interpolated through missing data at the interface. After an initial estimated alignment, we can then explore several other samples along the interface to evaluate and refine the offsets. Since imaging can occur uninterrupted for several hours at a high data rate (approximately 30GB/hour), one-time manual alignment (requiring only a few minutes) was an efficient method for determining offsets between neighboring columns.

3.4.2. Results and Conclusion

We have used these sectioning and alignment techniques to increase the volume of tissue imaged using KESM. We have tested these techniques using two data sets:

- Mouse brain microvasculature stained using India-ink perfusion. This creates a high-contrast data set containing a dense network of blood vessels (Figure 20).

- Rat somatosensory cortex stained with thionin (Nissl). This data set is lower contrast than the India-ink perfusion and contains significantly more data. In particular, cell nuclei of all neuronal and glial cells are visible. In addition, vasculature is visible as unstained filaments which were used as fiducials for alignment.

These techniques have allowed us to image large volumes of mouse spinal cord and rat cortex (Figure 21), well beyond the capabilities of optical sectioning techniques and single-column KESM.

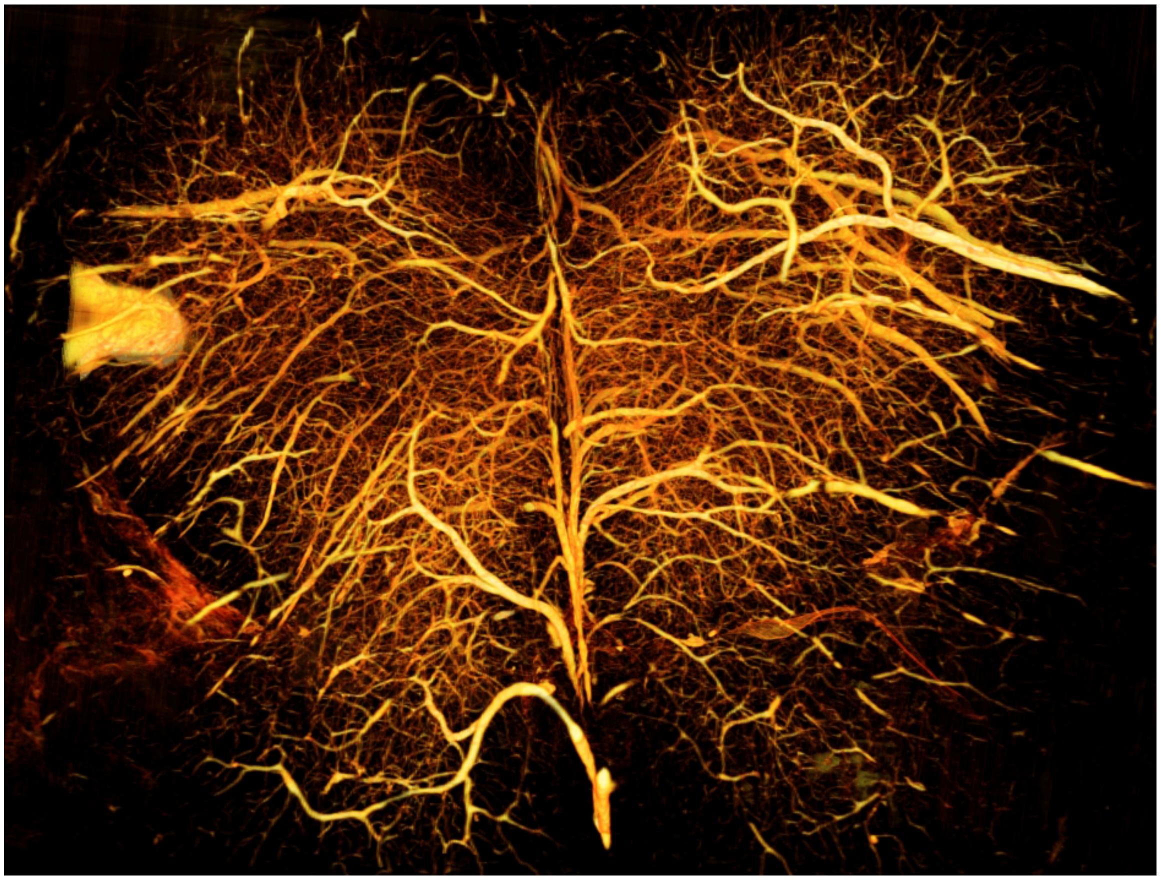


Fig. 20. Volume visualization of mouse spinal cord stained with India-ink. The tissue sample is approximately 2mm x 1mm in *x* and *y*. Several thousand sections were used to create the composite image (approximately 1mm deep).

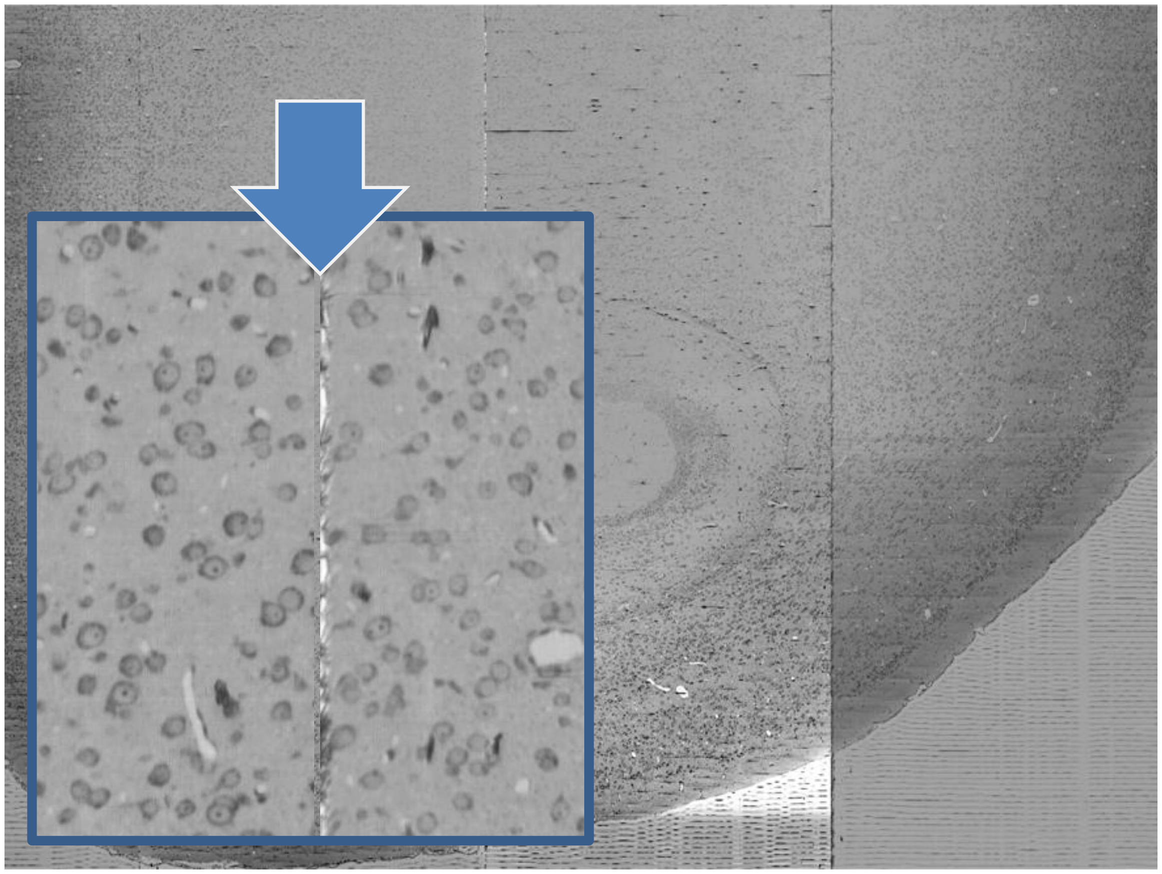


Fig. 21. Several columns of aligned rat brain. These sections include somatosensory cortex (left and right) and hippocampus (center). The inset shows a close-up of somatosensory cortex.

CHAPTER IV

FILAMENT SEGMENTATION

Filaments are common structures in biomedical imaging. Vascular trees are visible using many common imaging techniques such as MRI and CT, while cellular and sub-cellular structures are visible using microscopy [70]. The advent of new methods in high-throughput microscopy allows sub-cellular imaging on a much larger scale. These techniques create data sets that embed complex volumetric structures, such as neuronal and microvascular networks, consisting of vast numbers of interconnected filaments. These networks pose a unique problem in segmentation. The filaments are very thin and therefore must be imaged at a high resolution. In addition, they span large volumes of tissue. Volumetric data sets embedding comprehensive filament networks are therefore very large, even though the volume of the embedded network is often $< 6\%$ of the volume of the entire data set.

We present a framework for segmenting complex filament networks stored in volumetric data sets. We use a heuristic tracking method to create a model of the network. This model consists of filament centerlines, which provide an estimate of the internal medial axis of the network, including filament position and connectivity. We then use encoding based on Dynamic Tubular Grids [65] to store the volumetric data representing the network. In addition to providing significant compression, this technique can be used to eliminate data outside of the network, resulting in reduced noise and cleaner visualization. We show that our segmentation algorithm is highly parallelizable and can be run entirely on high-performance graphics hardware for fast results.

In this chapter we focus on segmenting data sets produced using high-throughput microscopy, since the structures embedded in these data sets are particularly difficult to segment using standard techniques due to their size and complexity. In particular, we will demonstrate our methods on data sets produced using Knife-Edge Scanning Microscopy

[52] and Array Tomography [61]. Both of these imaging techniques produce volumetric data sets embedding complex interconnected networks (Figure 22). We also test our algorithm on standard biomedical data sets by performing vessel segmentation in CT images and fibrin protein segmentation in confocal microscopy images.

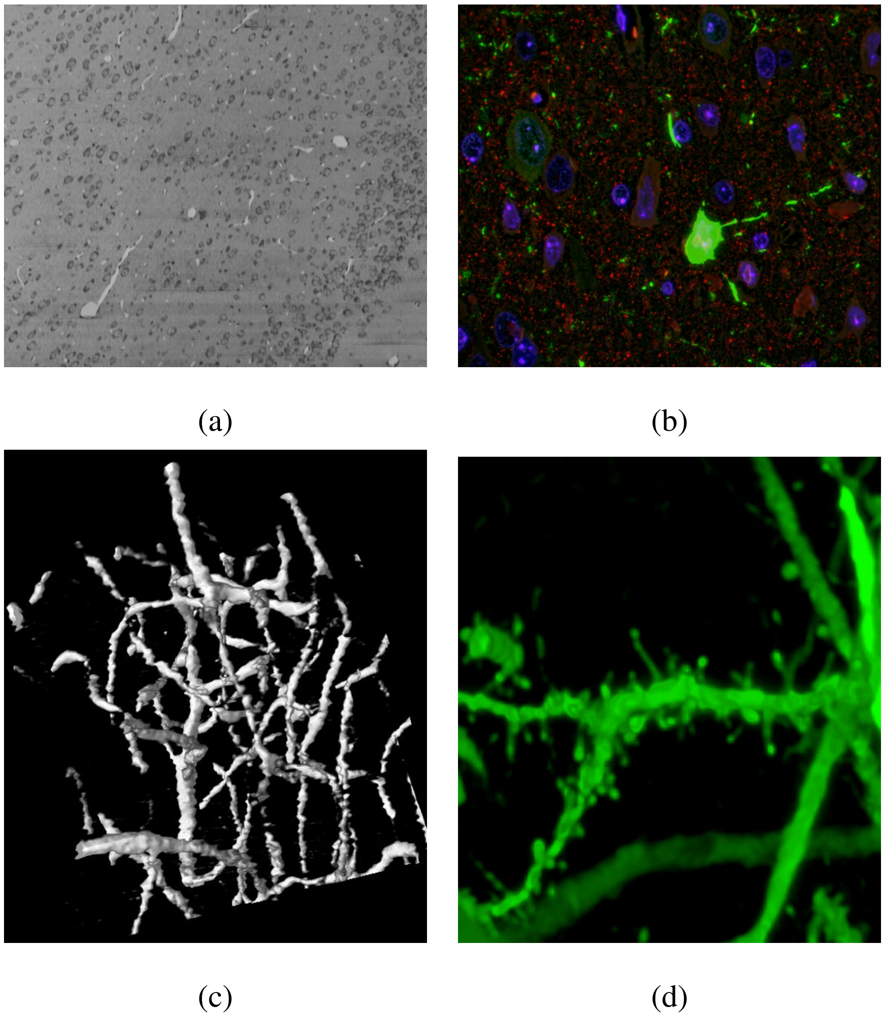


Fig. 22. Cropped sections of KESM (a) and Array Tomography (b) images. Imaging and staining artifacts, noise, and under-sampling cause gaps in microvessels (c) while neuronal filaments in AT images are have small surface details that interfere with tracking (d).

4.1. Previous Work

Many filament tracking algorithms are available for standard medical imaging applications such as CT and MRI image segmentation. An extensive review on the subject is given by Kirbas, et al. [35]. These focus on the segmentating large blood vessels, often using extensive image processing as an initial step [**?**, 93]. Multi-scale techniques [**?**, 16] can be used for feature detection, however most of the filaments in high-throughput microscopy data are thin, requiring the highest level of detail to resolve. This makes multi-scale segmentation impractical since the resulting down-sampling destroys filament information. Centerline detection [**?**, 86] and thinning [29] are generally based on selecting threshold values. These techniques work for high-contrast data, however finding useful thresholds in high-throughput microscopy data is difficult since noise often causes misclassifications. These misclassifications result in topological errors in the resulting network centerline, as well as false-positives due to over-segmentation. In addition, thin filaments often drop below the sampling resolution, creating gaps in threshold-based segmentations (Figure 22c). Region growing approaches [66, 81] require some initial surface, which is difficult to find given the complex topology of the embedded network. The methods that we describe in this chapter could potentially be used as an initial condition for region growing approaches. Template matching methods [80] are robust in the presence of noise but require a large template library containing oriented templates at multiple scales. In addition, filter matching is performed on every voxel even though only a small percentage of the volume contains structural data.

Vector tracking algorithms [3] are effective for continuous structures with a well-defined surface, but have trouble with low-contrast filaments or filaments with ill-defined or "fuzzy" surfaces (Figure 22d). Vector tracking methods also have some aspects in common with Lagrangian tracking methods used for flow visualization [73] and tractography in Diffusion Tensor MRI (DT-MRI) [6]. Although these methods perform particle-based vector tracking, they require an underlying vector field for particle advection. Our data sets contain embedded networks without an underlying flow field. We must therefore compute the path of these embedded filaments based only on scalar intensity information.

Many vessel tracking algorithms in the literature also take advantage of the tree-like structure of vessels. This assumption is applicable for MRI and CT, where only large vessels are imaged, but is not true at the microscopic level where the capillaries form an interconnected network. The algorithm that we propose works for both cyclic and acyclic structures and has aspects in common with both tracking and template matching methods. Also, we do not require pre-processing of the image data.

4.2. Filament Tracking

We are given a three-dimensional volume data set Λ of uniformly-spaced samples of a biological specimen containing a filament network. The biological tissue is stained such that samples on the network exhibit one intensity value I_n while samples outside the network exhibit some background intensity I_b. The algorithms described here assume that $I_n > I_b$. Images with dark filaments can be handled with minor adjustments to the given cost functions, or by inverting the input images.

If the data set Λ represents an ideal image of the network, we could set a threshold value I_t that specifies the boundary between the network and the surrounding tissue. However, the data set is generally corrupted by imaging artifacts and noise. These include artifacts caused by tissue preparation and staining. In addition, the sampling resolution is not always high enough to resolve all filaments in the network. We first create a model of the structure and connectivity of the filament network embedded within Λ.

We track individual filaments through the data set using a predictor corrector algorithm

(Figure 23). We estimate the centerline of a filament by determining the path of a single particle, which we refer to as a *tracer*, over time as it moves down each filament. At any given time t, the tracer has three properties:

- The tracer position $\mathbf{p_t}$ on the filament centerline.
- A vector $\mathbf{v_t}$ representing the estimated trajectory.
- A radius r_t defining the size of a template used to match the filament cross-section.

Given a tracer state at time t, we use an update algorithm that computes the next tracer state at time $(t+1)$ (Algorithm 2). The functions **PredictPath**, **CorrectPosition**, and **EstimateSize** are used to update the properties of the tracer over time. These functions are described in the following sections.

Algorithm 2 The predictor-corrector algorithm used to determine the next point along the axis of the filament.

```
Function TracerStep
Input: $\mathbf{p_t}$, $\mathbf{v_t}$, $r_t$
Output: $\mathbf{p_{t+1}}$, $\mathbf{v_{t+1}}$, $r_{t+1}$
$\mathbf{v}' = \textbf{PredictPath}(\mathbf{p_t}, \mathbf{v_t}, r_t)$
$\mathbf{p}' = \mathbf{p_t} + \Delta t \mathbf{v}'$
$\mathbf{p_{t+1}} = \textbf{CorrectPosition}(\mathbf{p}', \mathbf{v}', r_t)$
$r_{t+1} = \textbf{EstimateRadius}(\mathbf{p_{t+1}}, \mathbf{v}', r_t)$
$\mathbf{v_{t+1}} = \text{normalize}(\mathbf{p}' - \mathbf{p_t})$
```

4.2.1. Predicting Trajectories

Given the current tracer properties, we specify a heuristic method used to estimate the new trajectory $\mathbf{v}'$ such that an estimate of the tracer position $\mathbf{p}'$, where

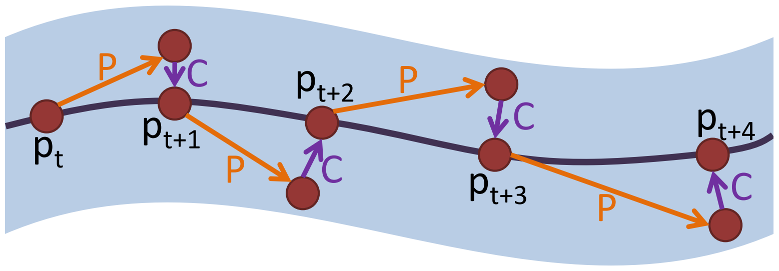


Fig. 23. A predictor-corrector algorithm. An initial prediction of the particle path (P) is made and the particle is advanced. A correction step (C) then refines the particle position to lie on the filament axis.

$$\mathbf{p}' = \mathbf{p_t} + \Delta t \mathbf{v}' \tag{4.1}$$

lies close to the filament centerline. In order to find the optimal direction vector, we choose a series of vectors $\mathbf{V} = [\mathbf{v_0}, \mathbf{v_1} ... \mathbf{v_n}]$ that lie within a solid angle α of the tracer trajectory $\mathbf{v_t}$. We then select the vector $\mathbf{v_i}$ that it is most closely aligned with the trajectory of the filament centerline.

For each vector, we take a volume sample $S_P(x,y,z)$ starting at $\mathbf{p_t}$ and oriented such that the z-axis for S_P is aligned with the vector $\mathbf{v_i} \in \mathbf{V}$ (Figure 24). We specify the cost function C_{pred}, which is minimized when these conditions are met.

$$C_{pred} = \sum_{x,y,z} w(x,y) \left| S_P(x,y,z) - f_{template}(x,y) \right| \tag{4.2}$$

Here, $f_{template}(x,y)$ is a template function, described below, and $w(x,y)$ is a weight function used to limit sampling outside of the filament surface. Note that S_P is the only function with a dependence on z. Thus, we consider S_P to be a series of images of the cross section of the filament. These images are compared to a template and then summed in order to create an integrated image representing the difference between the filament cross-section and the template. This integration process reduces the effects of artifacts and noise in the data set.

The function $f_{template}(x,y)$ is a user specified template that matches the filament cross-section. Most biological filaments have a cross-section that is mostly circular. In cases where filaments have fine surface details, integration along $\mathbf{v_i}$ tends to blur these details. Therefore, the filament cross section is generally well-represented using a Gaussian template:

$$f_{template}(x,y) = G(r_t) \tag{4.3}$$

where G is a Gaussian function scaled to the range $[0.0, 1.0]$ with a standard deviation equal to the filament radius. The weight function $w(x,y)$ is used to eliminate bias from other nearby filaments. This weight function also makes C_{pred} rotationally invariant along $\mathbf{v_i}$ since the actual sample image S_P is more easily collected as a cube. For $w(x,y)$ we use a circular Gaussian slightly wider than the template function. These functions are shown in Figure 25. An optimal direction estimate $\mathbf{v}'$ is selected by finding the vector $\mathbf{v_i}$ with the lowest cost.

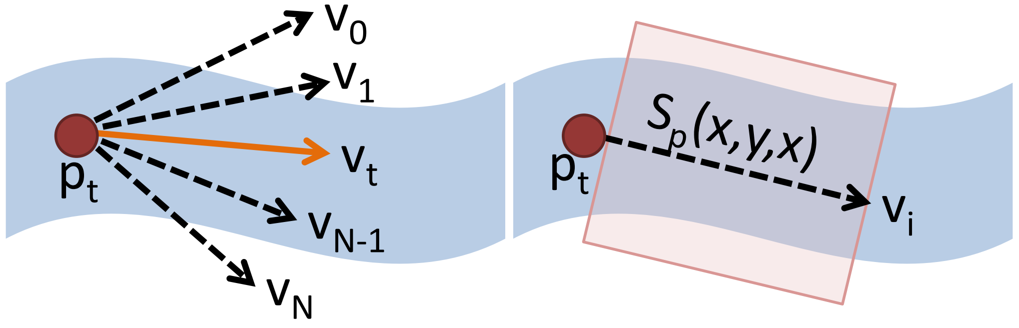


Fig. 24. Sample vectors used to find the optimal tracer orientation (left). For each vector, a volume sample $S_P(x,y,z)$ is taken and compared to a template in order to minimize the cost function C_{pred}. The volume sample is taken such that the z-axis of S_P lies along the vector $\mathbf{v_i}$ (right).

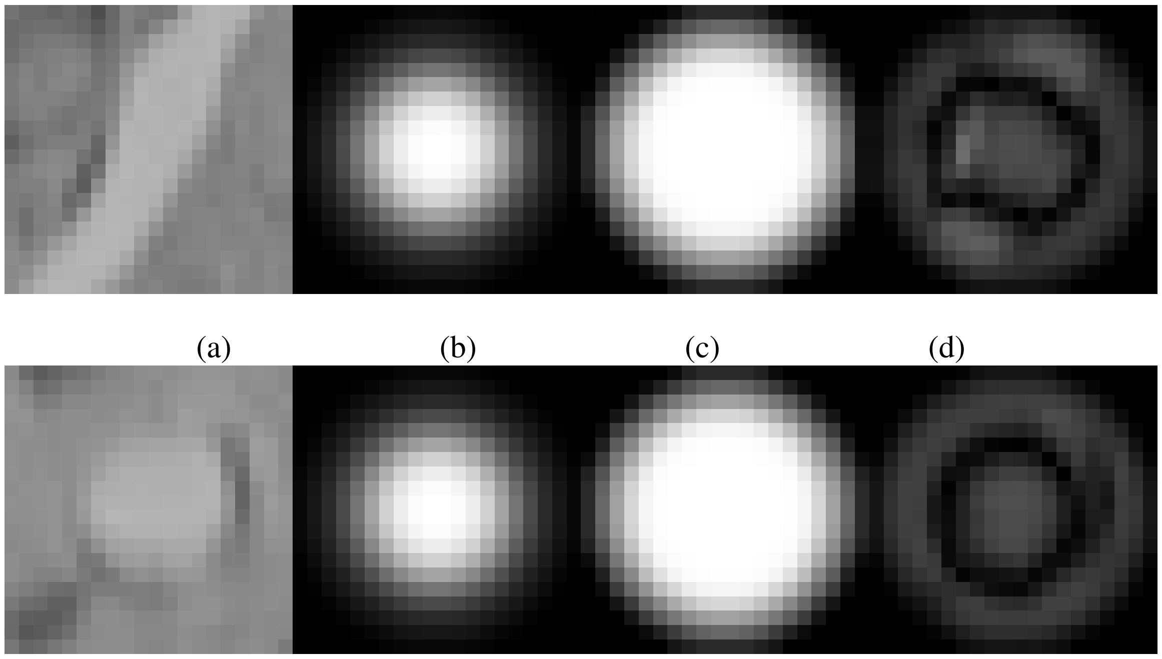


Fig. 25. Predictor evaluation of a low-contrast filament (unaligned [top] and aligned [bottom]). (a) S_P integrated over a small volume along $\mathbf{v_i}$, (b) $f_{template}$, (c) w, (d) C_{pred}. The average image intensity $\frac{C_{pred}}{n^2}$ is 31.2 (aligned) versus 37.8 (unaligned).

4.2.2. Position Correction

When moving the tracer along the estimated trajectory $\mathbf{v}'$ of the filament centerline, there is generally some error in the final position due to the finite number of vectors in $\mathbf{V}$ used for prediction. We correct the new tracer position by taking a second series of samples. We test a set of points $\mathbf{P} = [\mathbf{p_0}, \mathbf{p_1} ... \mathbf{p_m}]$ that lie in the plane defined by $\mathbf{p}'$ with normal $\mathbf{v}'$ within ε distance of $\mathbf{p}'$. For each point, we take another volume sample starting at $\mathbf{p_i} \in \mathbf{P}$ in the direction of $-\mathbf{v}'$ (Figure 26). The cost function used to evaluate each sample point is given by

$$C_{corr} = \sum_{x,y,z} [S_C(x,y,z)G(r_t)] \tag{4.4}$$

where S_C is the volume sample for $\mathbf{p_i}$ and $G(r_t)$ is a Gaussian scaled to the range $[0.0, 1.0]$ with standard deviation r_t. The new tracer position is selected based on the position that maximizes the value of C_{corr}.

After the final position $\mathbf{p_{t+1}}$ is computed, we set the tracer trajectory to the actual value

$$\mathbf{v_{t+1}} = \frac{\mathbf{p_{t+1}} - \mathbf{p_t}}{||\mathbf{p_{t+1}} - \mathbf{p_t}||} \tag{4.5}$$

which takes into account the final position $\mathbf{p_{t+1}}$ computed using both prediction and correction steps.

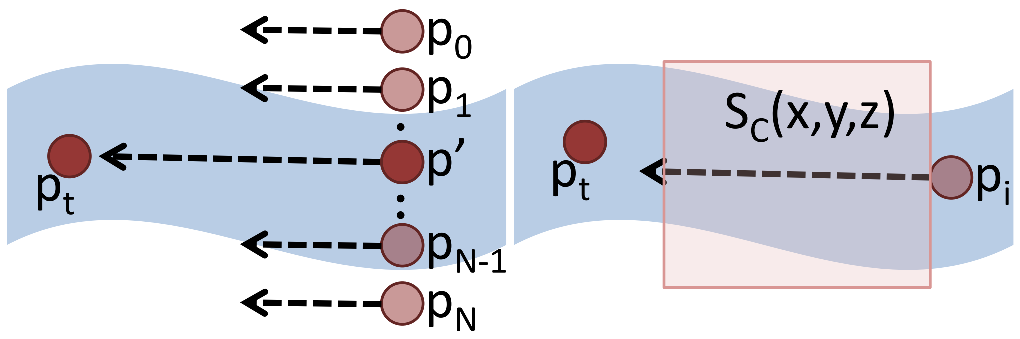


Fig. 26. Sample points used to correct the position of $\mathbf{p}'$ (left). The volume sample S_C is oriented along the direction of $-\mathbf{v}'$ (right).

4.2.3. Template Radius

The final step in our tracking algorithm updates the radius r_t of the template. Adjusting the size of the template is necessary to allow our algorithm to better match both the template and weight functions to the filament cross-section. This limits sampling to a region local to the filament and is a key advantage of our algorithm over standard template matching. Instead of processing the entire data set, we consider only regions local to the network.

Finally, this provides a rudimentary estimate of the filament radius, which we later use to extract filament information from the volume. Note that this estimate reflects a discrete estimate of how well the template size matches the filament, rather than an accurate measurement of the filament radius. Once the centerline for a filament is found, methods are available for more accurately computing the filament radius [53].

We estimate the new template radius by creating another series of volume samples of the region between $\mathbf{p_t}$ and $\mathbf{p_{t+1}}$. We specify a series of radius values $\mathbf{r} = [r_0, r_1, r_2...r_n]$. For each sample radius $r_i \in \mathbf{r}$, we compute the response of a template of radius r_i using C_{pred} (Equation 4.2). The template size that provides the best response (lowest value of C_{pred}) is the new template radius r_{t+1}. As described in the next section, this value is used to dynamically adjust the size of the volume samples, template function, and weight function.

4.3. Computation on Graphics Hardware

Our tracking algorithm is significantly more efficient than template matching since sampling is limited to voxels near the network. Our heuristic also ensures that orientations and sizes close to those actually describing the filament structure and centerline are tested. However, creating the volume samples necessary for prediction, correction, and resizing requires that a small region of the data set be reconstructed and re-sampled. Together with the large number of volume samples required for accurate evaluation, this becomes the most computationally expensive part of our algorithm. Other vector tracking methods deal with this by imposing strict limitations on sampling. For example, Al Kofahi et al. [3] sample a limited number of points along the filament surface, which reduces stability when the surface is not well defined or contains sharp features (Figure 22d).

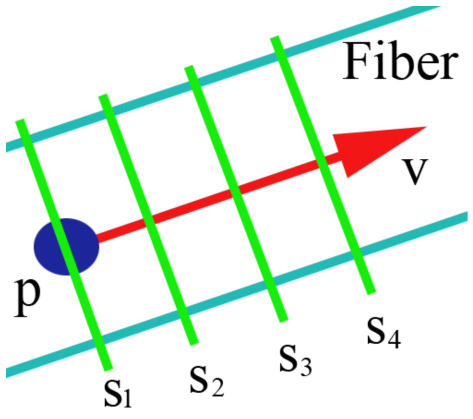
In this section, we show that testing the entire filament cross-section can be performed efficiently using graphics hardware. Cost function evaluation and branch detection can also take advantage of the shader pipelines and rasterization functions available on modern graphics hardware. By performing the computation completely on the graphics card, we also avoid introducing bottlenecks caused by data transfer across the system bus.

4.3.1. Sampling

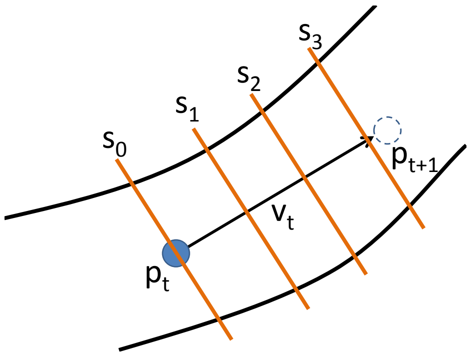
We take advantage of hardware accelerated texture look-up and interpolation by loading the volume data to the GPU as a 3D texture. Samples are taken by creating a stack of quadrilaterals specified in texture-space (Figure 27a). Each quadrilateral represents a single slice along the z-axis of the sample function S (Section 4.2.1). We then specify a texture matrix that transforms the points representing each quad to match the properties of the tracer $(\mathbf{p_t}, \mathbf{v_t}, r_t)$ at each vertex (Figure 27b).

For efficiency, we pre-compute all of the vectors, points, and scales used for prediction, correction, and sizing respectively. We then position stacks of quadrilaterals in texture space at the appropriate positions and orientations (Figure 27c). These pre-positioned sample planes are then stored in three separate display lists which can be rendered during the prediction, correction, and sizing stages. Sampling is performed by specifying a texture matrix describing the transformation to the current tracer state. We then render the appropriate display list for the prediction, correction, and sizing stages.

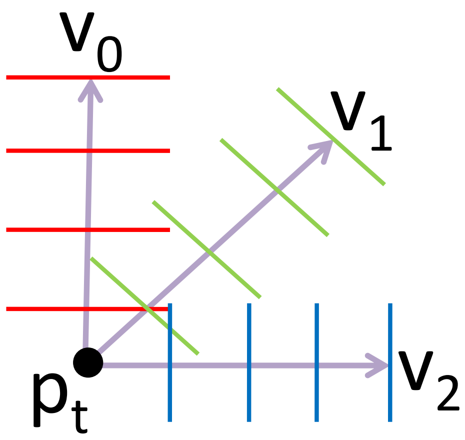
In order to facilitate the evaluation of cost functions, we store the results of the sampling stage in a two-dimensional texture map. We do this by specifying geometric coordinates for each of the vertices in the prediction, correction, and sizing display lists. We specify these coordinates so that the quadrilaterals are rendered in a two-dimensional array (Figure 27d). We store each z-slice of the volume sample as a row in the texture while each vector ($\mathbf{v_i}$), point ($\mathbf{p_i}$), or template radius (r_i) is stored as a column.



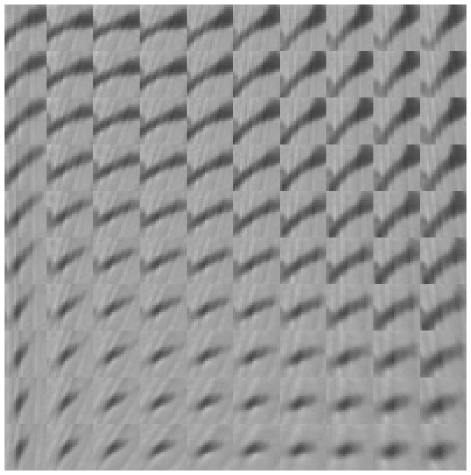
(a)



(b)



(c)



(d)

Fig. 27. Image stack for a single volume sample shown in texture space (a) and relative to a filament (b). We create a display list of several samples (c) so that they can be generated in parallel on the GPU. The resulting samples are rendered to a 2D texture for later evaluation (d).

4.3.2. Cost Function Evaluation

After rendering the samples to a texture, we use the render target to evaluate the cost function for each step of the tracking algorithm. The final result of each cost function is calculated in four passes:

- Render the sample geometry to a texture (Section 4.3.1).
- Evaluate the cost functions using a fragment program.
- Integrate the result for each sample using a reduction operation.
- Copy the final cost values from the GPU.

As an initialization step, we pre-compute both the template and weight functions and store them as textures (Figure 25b and 25c). After rendering the samples, we evaluate the cost function inside the summation (Equations 4.2 and 4.4). We use a fragment program to perform this computation on every pixel in the sample texture and store the result in a new texture. We then use a multi-step reduction operation [27] to integrate the pixel values representing each sample. The output of the reduction is a vector of values, each representing a result of the cost function for each sample.

4.3.3. Branch Detection

We determine network connectivity based on the proximity of two filament centerlines and their associated diameter. As each filament is tracked, we build a line strip from the history of all tracer positions. This line strip represents the centerline of the filament. After each filament is completely tracked, the centerline is added to a display list representing the entire network. As we track each subsequent filament, we check for an intersection with the existing network.

At each time step, we create an orthographic view volume between positions $\mathbf{p_t}$ and $\mathbf{p_{t+1}}$ along z, with x and y extents equal to the tracer radius r_t. We then render the network display list. Any rendered geometry that is not culled by the viewport transformation indicates an intersection between the current filament and the rest of the network. We then search through all filaments to find the exact branch point. Although we test for intersections at each time step, this operation can be performed quickly using the OpenGL selection buffer [89]. We perform the more complex search through all filaments only if an intersection with the network is detected.

4.3.4. Simulation

We now consider parameters that influence the effectiveness of our algorithm on a data set. As is the case with Lagrangian tracking techniques using particle advection, the time step $\triangle t$ affects the stability and accuracy of our algorithm (Equation 4.2 and Algorithm 2). Note that we are using an implicit predictor-corrector method for advancing the tracer position, so our choice of $\triangle t$ is less constrained than for explicit (prediction only) integration. We set $0 < \triangle t \leq 1$. Values of $\triangle t > 1$ should be avoided since this would advance the tracer across regions of a filament that have not been sampled.

Another important set of parameters specify the resolution of the volume samples. Provided that the resolution in the *xy*-plane is large enough to represent the filament cross section (≈ 10 pixels), this has little affect on the tracking results. The *z*-axis resolution, however, is an important consideration. This value is dependent on the noise in the data set and the smoothness of the filament surface. Increasing the value provides more integration along the filament length, and therefore more accuracy in noisy data, at the expense of greater sampling and evaluation time. High-contrast blood vessels with smooth surfaces required only two sample planes while we used up to $z = 5$ for fluorescence data in AT (Figure (22b and 22d).

The depth of the volume sample, representing the length of the filament along which the sample is integrated, depends on the average filament curvature. We used a smaller depth for high-curvature KESM microvascular filaments and Lung CT while longer samples were used to track networks containing low-curvature filaments found in AT, fibrin protein, and neuron data sets (Section 5.4). An overview describing how our parameters affect the tracking results is given in Section 4.5.1.

4.3.5. Stopping Conditions

The large size of high-throughput data sets and the memory constraints on graphics hardware limits our data size to 512^3 voxel blocks. We therefore stop tracking when filaments reach the edge of the data set. If larger data sets are desired we break them into multiple blocks that are tracked separately and joined at the interface.

Some types of networks, such as neural networks, have filaments that can terminate. In other cases, a filament may no longer be trackable due to noise over an extended region or because it has dropped below the sampling resolution. We end tracking by constraining the template radius r_t to some minimum value. When a filament in the data set terminates, the template shrinks below the specified threshold. Tracking also stops at filament intersections as mentioned in Section 4.3.3.

4.3.6. Seed Point Selection

We begin tracking from initial values of $\mathbf{p_0}$, $\mathbf{v_0}$, and r_0. We set these seed points using a method similar to the one proposed by Al-Kofahi et al. [3]. The data set is projected onto a plane and seed point positions are placed based on a conservative threshold and projected back into the data set. Based on the seed point position, the initial tracer state is set using a two-step process:

- The position is refined and an initial orientation and size are determined using a single prediction, correction, and sizing step. Although p_0 can be altered during the correction step, it is not moved along the predicted direction.

- Two tracers are created with identical position and size but with opposite trajectory (v_0 and $-v_0$).

This produces a list of tracers, where there are two tracers for each seed point. We iterate through this list,tracking each associated filament. In order for the data set to be fully tracked, every segment must contain a seed point. If a segment contains more than one seed point, successive tracers will immediately detect the resulting centerline geometry and terminate (Section 4.3.3). This termination requires less than 6ms on average to compute.

4.4. Data Storage

Our tracking algorithm produces a graph representing the structure and topology of the embedded network. We now show how this graph is used to classify volumetric data associated with the network. We then compress the network data, taking advantage of the small volume of the network relative to the embedding data set Λ. This also allows us to eliminate excess noise, improving the quality of volumetric visualizations. In addition to classifying each voxel as inside or outside of the network, we also associate each voxel with a single filament.

4.4.1. Network Bounding Volumes

We first determine if any given voxel is part of the network. We use an implicit representation of the network based on the information acquired during tracking (Section 4.2). Given the traced network, we create a bounding volume that allows us to classify each voxel. This bounding volume is based on the tracked filament centerlines and the template radius. In most cases, the template radius provides an upper-bound on the filament radius. In extremely noisy data sets where the filaments contain sharp surface features, the user may have to add an additional scale factor to increase the radius of the bounding volume around the filament.

One possible representation for the bounding volume is an L-Block Structure [58].

This structure uses a 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 requires a single position and size along each axis.

Consider a single filament segment specified by two points $\mathbf{p_n}$ and $\mathbf{p_{n+1}}$ with radii r_n and r_{n+1}. A filament segment is guaranteed to be bounded by an AABB around two spheres, each positioned at points $\mathbf{p_n}$ and $\mathbf{p_{n+1}}$ with radii r_n and r_{n+1} respectively (Figure 28a). As each segment is bounded, however, the L-Blocks overlap significantly (Figure 28b). Redundant information in L-Blocks is eliminated using a time-consuming iterative refinement algorithm [17]. In addition, L-Blocks are designed for archival storage and do not allow random access.

A tighter bound can be constructed using a series of truncated generalized cones (TGCs) [26]. Each TGC is defined by two adjacent points along a filament. The TGCs are connected end-to-end with the caps oriented by the direction of neighboring line segments:

$$\mathbf{n_b} = \frac{1}{2}\left(\frac{\mathbf{b}-\mathbf{a}}{||\mathbf{b}-\mathbf{a}||} + \frac{\mathbf{c}-\mathbf{b}}{||\mathbf{c}-\mathbf{b}||}\right) \tag{4.6}$$

where $\mathbf{a}$, $\mathbf{b}$, and $\mathbf{c}$ are three consecutive points on the medial axis of a filament. The radius of each end cap at point $\mathbf{p_n}$ is equal to r_n (Figure 30).

Once we have constructed a linked series of TGCs for each filament, we determine if a point is inside the bounding volume by implicitly defining the region within a TGC. Assuming that there is a point $\mathbf{p_x}$ under consideration, we 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 6.3, 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 29a)

$$\mathbf{n_{plane}} = \frac{\mathbf{p_{n+1}} - \mathbf{p_n}}{||\mathbf{p_{n+1}} - \mathbf{p_n}||} \times \frac{\mathbf{p_x} - \mathbf{p_n}}{||\mathbf{p_x} - \mathbf{p_n}||} \tag{4.7}$$

For each end cap, we find the end point of the line segment that represents the intersection of the end cap with the plane (Figure 29b):

$$\mathbf{p_{r,n}} = \mathbf{p_n} + r_n(\mathbf{n_{plane}} \times \mathbf{n_n}) \tag{4.8}$$

If $\mathbf{p_x}$ lies within the polygon formed by $(\mathbf{p_n}, \mathbf{p_{r,n}}, \mathbf{p_{r,n+1}}, \mathbf{p_{n+1}})$, it is also within the TGC.

We now determine if a point is part of a filament by testing the point against every TGC for every filament in the network. We accelerate this by computing an AABB around each TGC using the position $\mathbf{p_n}$, normal $\mathbf{n_n}$, and radius r_n of each end cap. Since the end cap is circular, we compute the minimum and maximum extent of the end cap n along any axis. This equation for the x-axis is

$$AABB_{X,n} = \mathbf{p_n} \pm r_n(\sqrt{1 - \mid \mathbf{n_n} \cdot \mathbf{X} \mid^2}) \tag{4.9}$$

where $\mathbf{X} = \begin{bmatrix} 1 & 0 & 0 \end{bmatrix}$. The position and size of the AABB along the X-axis are computed using:

$$AABB_{X,pos} = \min(AABB_{X,n}, AABB_{X,n+1}) \tag{4.10}$$

$$AABB_{X,size} = \max(AABB_{X,n}, AABB_{X,n+1}) - AABB_{X,pos} \tag{4.11}$$

By snapping each AABB to the embedding function Λ, we compare only voxels within an AABB with the associated TGC. We then label each voxel as either inside or outside of the network. We can also classify each voxel with its associated filament.

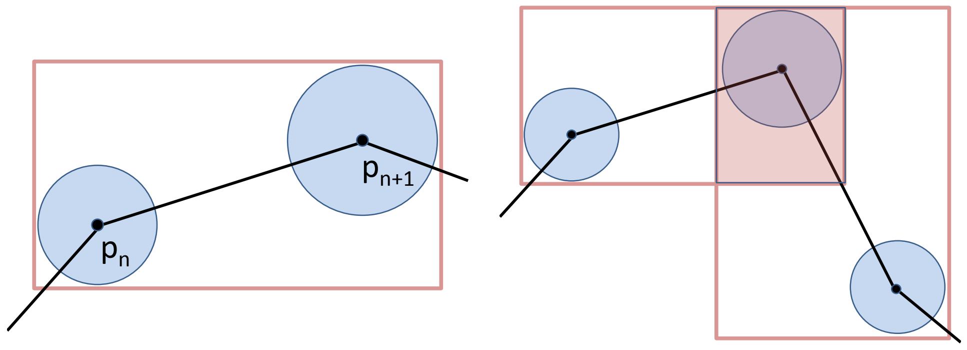


Fig. 28. Bounding two spheres of radii r_n and r_{n+1} (a) guarantees that all voxels belonging to the filament are included in the bounding volume. However, there will be significant overlap (b).

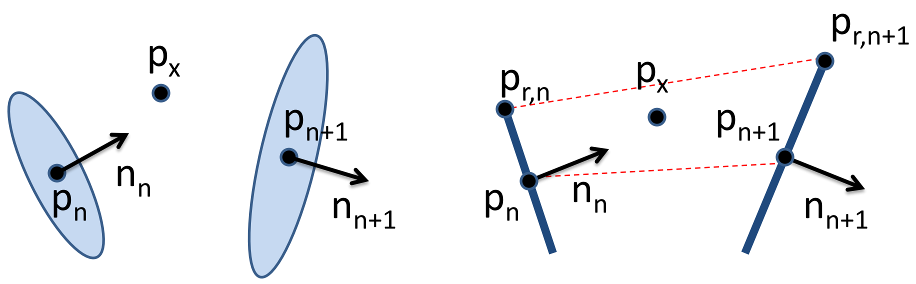
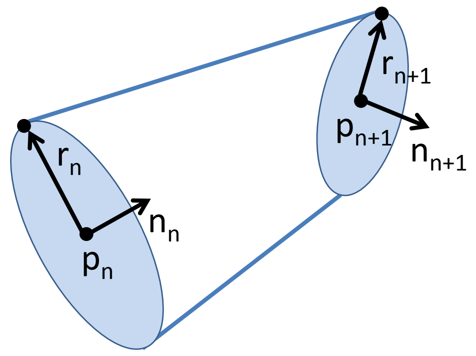


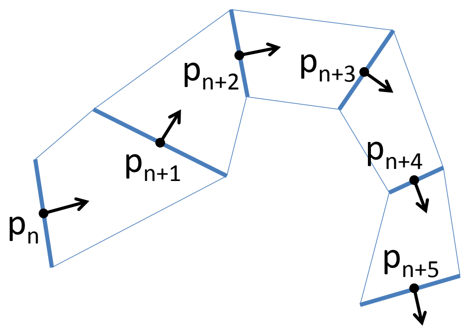
Fig. 29. (a) End caps of a TGC and (b) the intersection of the TGC with the plane defined by $\mathbf{p_x}$, $\mathbf{p_n}$, and $\mathbf{p_{n+1}}$.

4.4.2. Encoding

The volume of an embedded network is usually much less than that of the entire data set. We can therefore reduce the amount of space used to store the classified network by encoding each voxel into a space-saving data structure. In addition to acting as a bounding volume, L-Blocks (Linked AABBs) are efficient for storage. However, they must be uncompressed to a regular grid in order to allow random access. The most common structure for storing sparse three-dimensional data is the octree. However, octrees are inefficient for



(a)



(b)

Fig. 30. Truncated generalized cones (a) fit end-to-end to construct the bounding volume for a filament (b).

space-filling volumetric structures. We instead use a dynamic tubular grid (DT-Grid) [65]. This is a run-length encoding method for storing sparse volumetric data and has been shown to provide more compression and faster random access than octrees. DT-Grids provide significant compression by eliminating empty space between volumetric structures. Further details can be found in the paper by Nielson and Museth [65].

4.5. Results

We have tested our algorithm for tracking filaments in both high-throughput and standard biomedical data sets:

- Array Tomography (AT): High-throughput microscopy data set containing neural networks. These filaments are difficult to segment using standard techniques because they have an ill-defined surface, non-uniform intensity among filaments, uneven distribution of stain through the filaments, and standard imaging noise (CCD, photon-shot, and camera noise) (Figure 37a and 22a).

- KESM Vasculature: High-throughput microscopy data set containing cellular and

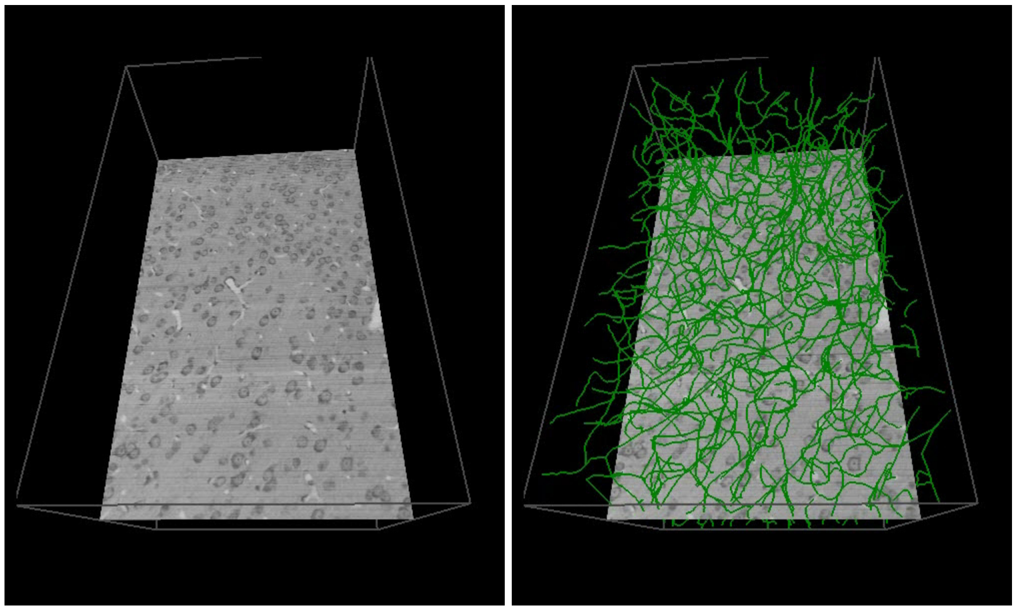


Fig. 31. Untraced (left) and traced (right) 512x512x512 voxel KESM vascular data sets.

vascular data. Filaments are low contrast and have high curvature. Thinning and isosurface segmentation results in over-segmentation due to cellular structures outside of the network while the high curvature and low-contrast of the vessels makes standard vector-tracking impractical (Figure 38 and 31).

- KESM Neurons: High-throughput microscopy data set containing neural networks. These filaments frequently drop below the sampling resolution of the microscope, resulting in frequent gaps in the network. The very low contrast and gradual changes in intensity also makes it impossible to select an isovalue or transfer function that completely classifies the network (Figure 39).

- Lung CT: Standard biomedical data set with complex structure. This data set is high contrast and contains very little noise, however the filaments are short and have high curvature (Figure 37b).

- Fibrin: Data set imaged using confocal microscopy. These filaments are low curva-

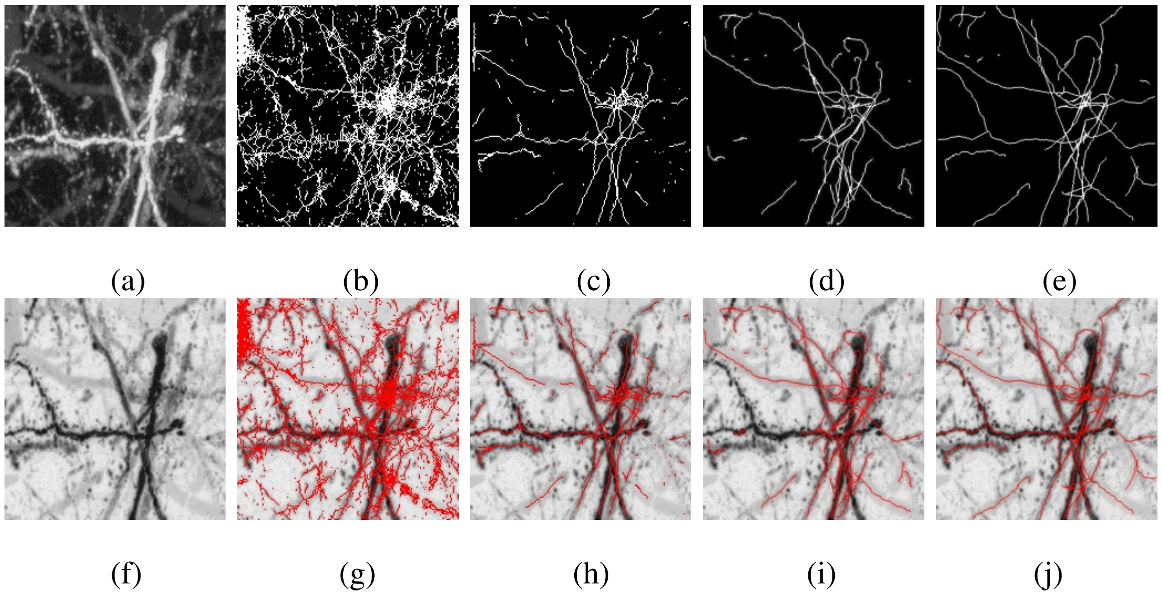


Fig. 32. Cropped volume of array tomography data containing neuronal dendrites. A maximum intensity projection of the data set is shown (a) and inverted for clarity (f). We then show several tracking results (b) - (e) and corresponding overlays (g) - (j). Skeletons produced using thinning algorithms (b) & (g) have noise and topological errors. These can be reduced by filtering the original data (c) & (h) (5x5 median filter) which is time consuming and damages low-contrast regions. Previous tracking algorithms (d) & (i) sample only the surface, which is often ill-defined in AT data sets while our tracking method (e) & (j) samples the entire filament cross-section.

ture but frequently drop below the sampling resolution (Figure 41). The images also contain artifacts, such as CCD noise, commonly found in confocal imaging.

4.5.1. Tracking

We first track each data set using the methods described in Sections 4.2 and 4.3. Using the AT data set, we provide a qualitative comparison of our tracking method with thinning [29], filtering, and vector-tracking using only surface sampling (Figure 32). Thinning algorithms have difficulty with noisy data and produce topologically incorrect skeletons with several extraneous segments due to over-segmentation of the original isosurface. These artifacts

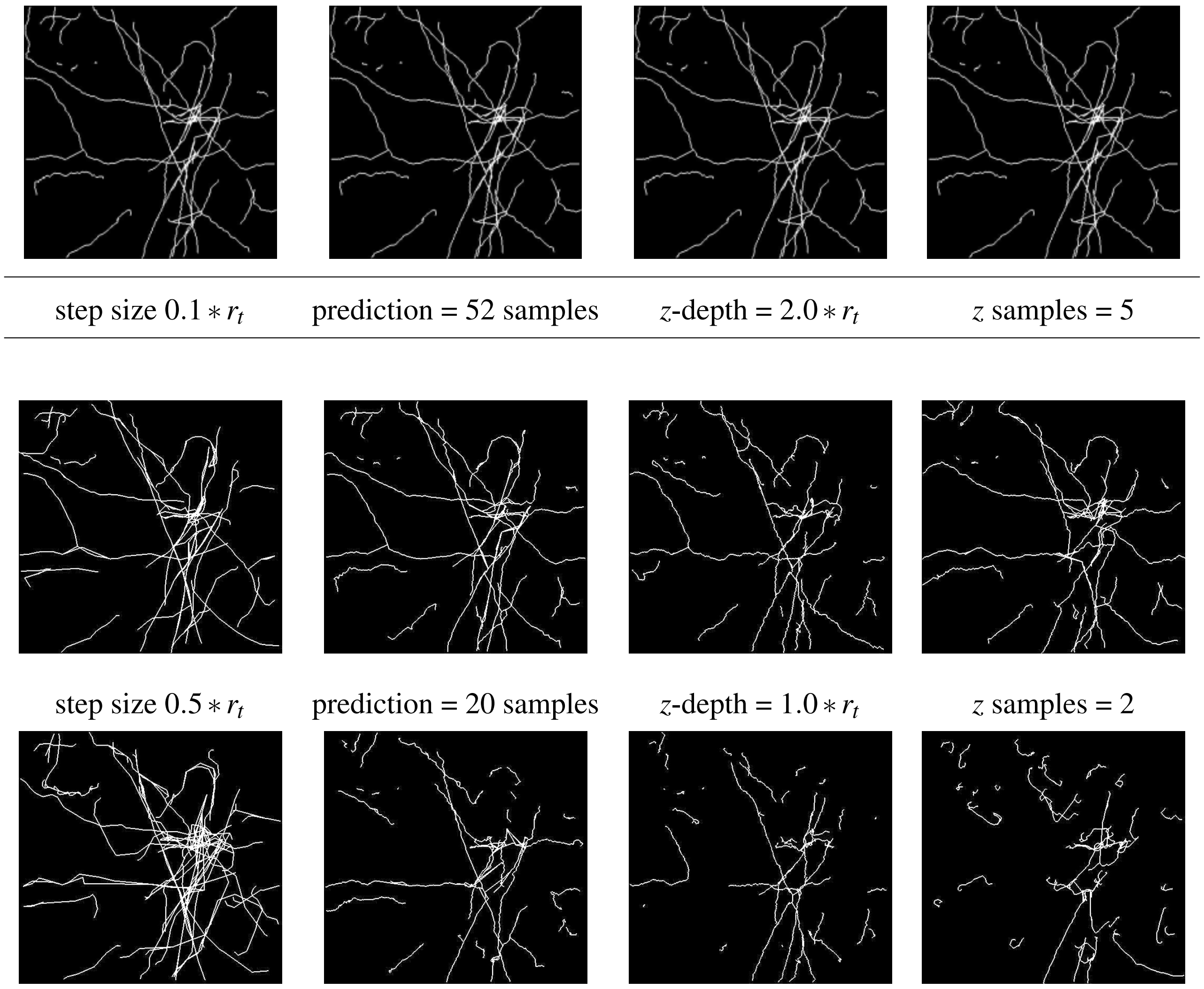


Fig. 33. Effects of parameters on tracking results for the Array-Tomography neuronal data set. Original tracking results are shown with standard parameter settings (top) and are identical. The remaining images are created by varying a single parameter and use the same seed points. From left to right: (1) Reducing the integration step size $\triangle t$, (2) reducing the number of prediction vectors tested, (3) reducing the length of the filament sampled (volume sample depth), and (4) reducing the number of samples along the z-axis of the volume sample.

can be reduced using filtering techniques, such as the median filter, however this is time consuming and causes additional breaks in filaments that are near the sampling resolution. Finally, previous methods sample only the filament surface, which is ill-defined due to high-frequency surface features, noise, and non-uniform staining. We also provide an overview of how our tracking parameters affect the resulting centerlines (Figure 33).

4.5.2. GPU Speedup

We have implemented both CPU and GPU based algorithms for comparison. We consider only the prediction step, which requires the most samples. We also compare our full GPU-based algorithm to an implementation using multiple CPU cores. The linear speedup reflects the high level of parallelism. We feel that this is an important result, showing future means of speedup for large data sets since we do not expect processor speed to increase at an appreciable rate. We compare four implementations in Figure 34.

- CPU-based implementation (single-core)
- CPU-based implementation (quad-core)
- CPU-based evaluation (GPU performs sampling)
- GPU-based implementation

The majority of time on all hardware tested is spent interpolating samples. As expected, using the GPU for sampling provides the largest factor of speedup. When using graphics hardware for sampling and the CPU for evaluation of the cost function, the transfer of data across the bus is the major bottleneck. When the GPU is fully utilized, performing all computation on the graphics card provides around a 4X to 5X speedup over only using the GPU for sampling (and a 20X speedup over the single-CPU implementation). As the number of fragment processors increases, we expect this factor to become larger.

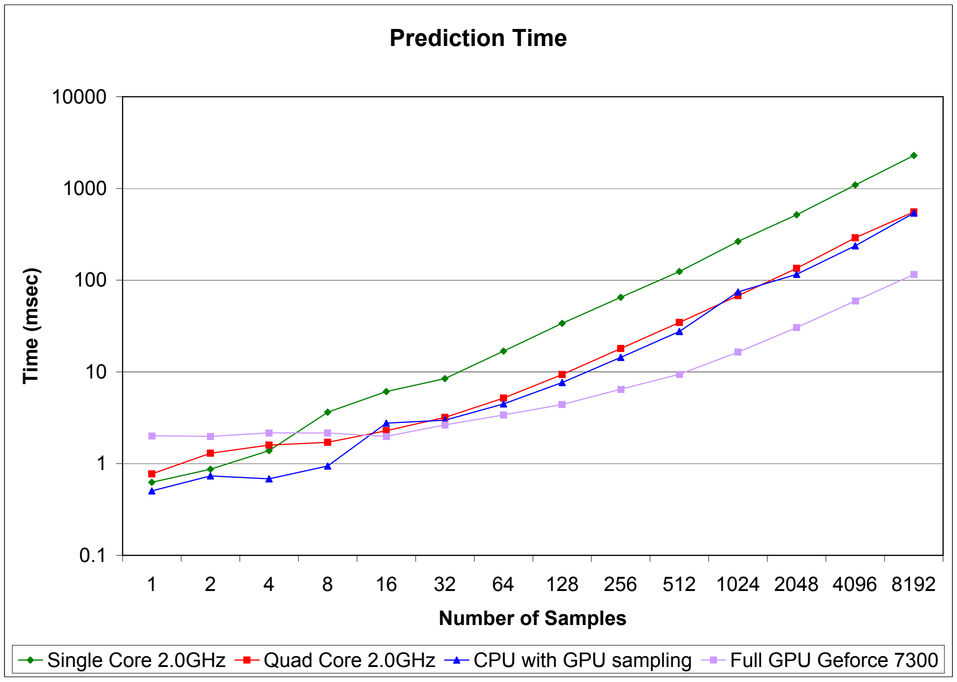


Fig. 34. Average time plotted, on a logarithmic scale, to make a single prediction based on the number of cross-sections.

We provide a breakdown of the time required to sample a 512^3 KESM Vascular data set (Table I). The data set and resulting centerlines are shown in Figure 31.

4.5.3. Compression

After tracking and classifying the network, we compare the memory required to store the classified data using L-Blocks and DT-Grids against that for the original uniform grid (Figure 36). In addition to providing better compression in all cases, DT-Grids allow random access into the data set and avoid storing redundant data. In all cases, a large percentage of the data representing the space between filaments was culled using the TGC bounding volume (Table II).

We compare results using volume rendering to display KESM data both before and

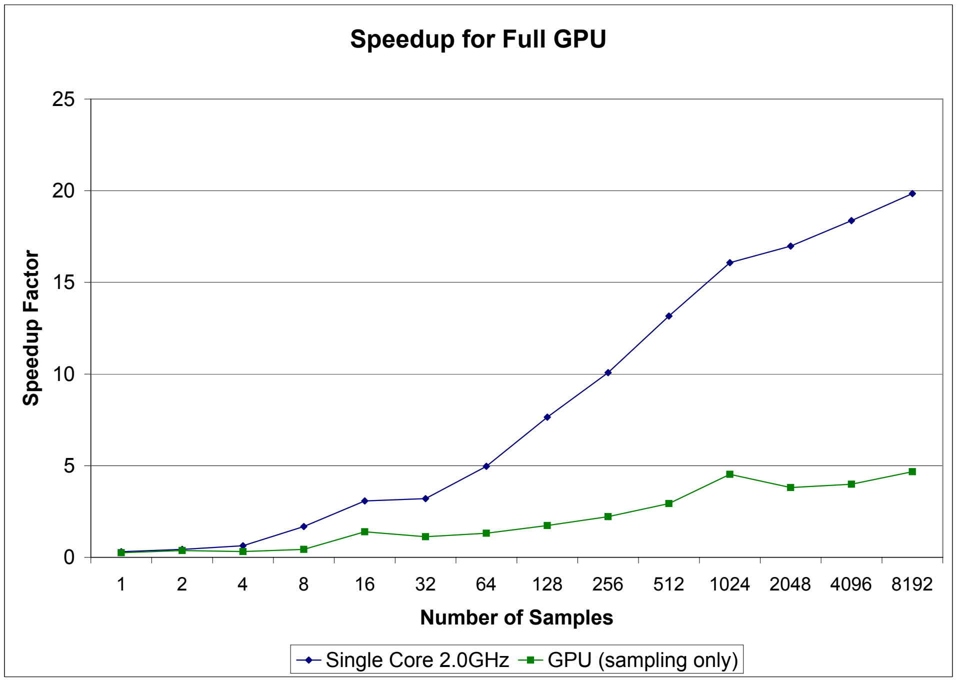


Fig. 35. The speedup provided by the full GPU implementation over (blue) a complete evaluation on the CPU and (green) a CPU evaluation that uses the GPU for sampling (bottom).

after filament tracking and encoding. Segmentation of the underlying network allows us to cull noise and excess data from the visualization, providing higher-quality images describing network structure (Figure 38). The underlying structure of the network also allows us to extract volume data associated with specified filaments. After tracing a large block of cellular data, we select an individual cell for visualization (Figure 39a and 39b). We also show an example of segmented and traced AT and Lung CT images (Figure 37).

Since each threaded series of blocks contains all of the volumetric data necessary to reconstruct the filament, individual filaments or filament networks can be rendered independently of the rest of the data set. This is done by selecting a single node and traversing the network in a breadth-first fashion (Figure 40). This technique can also be used to locate

Table I. Timing breakdown for tracking the KESM Vasculature data set (Figure 31). The total time required to track the data set is given (left) along with an average breakdown of the operations required for a single tracking step (center). Further breakdown of an average prediction step is also shown (right).

Total Timings	
Seed Positioning	0.011s
Seed Orientation	1.707s
Tracking Time	28.744s
Total	30.462s

Breakdown for a Tracking Step	
Predictor	1.248ms
Corrector	1.177ms
Sizer	0.994ms
Intersections	6.236ms

Breakdown for Predictor	
Sampling	0.265ms
Evaluation	0.230ms
Convolution	0.437ms
Compute Direction	0.021ms

Table II. Percentage of the volume culled from each data set based on the tracking and bounding segmentation scheme.

Data Set	Culled
Array Tomography	95.55%
KESM Vasculature	92.72%
KESM Neurons	97.53%
Lung CT	96.15%

independent networks by expanding a network until all valid nodes are reached. We use this to extract individual cells in neuronal populations (Figure 39c) as well as individual connected components in spinal cord microvasculature scanned using KESM (Figure 42).

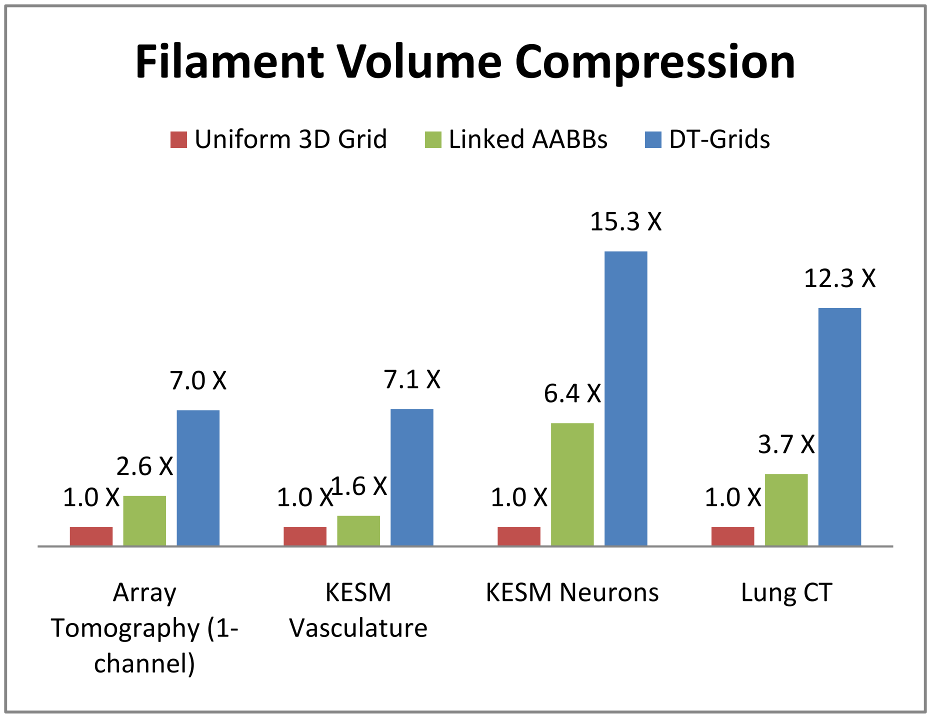


Fig. 36. Comparison of the compression achieved by storing the volume data on a uniform grid, as a series of AABBs, and using DT-Grids.

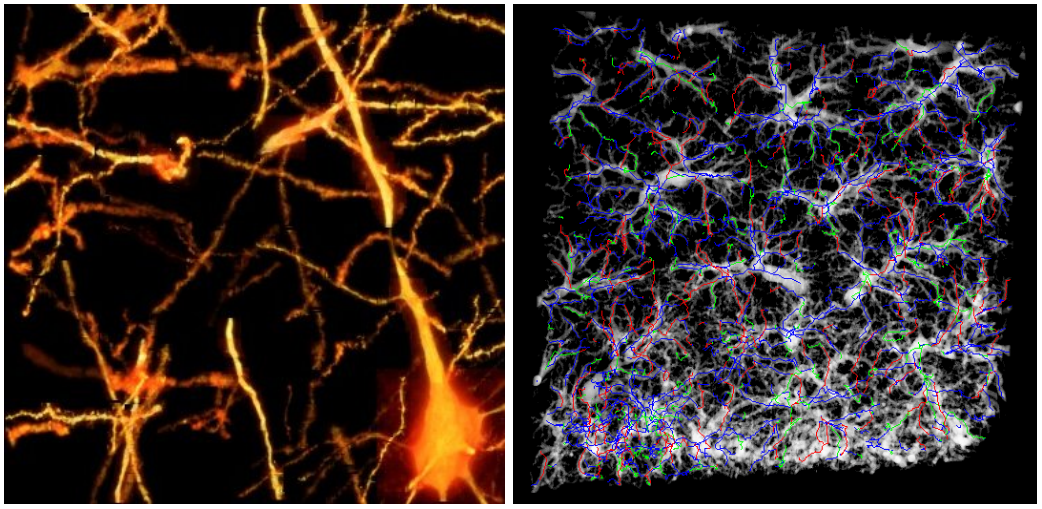


Fig. 37. (left) Visualization of segmented neurons in AT data. (right) Traced vasculature imaged using CT Lung overlayed over a volume visualization of the original data set.

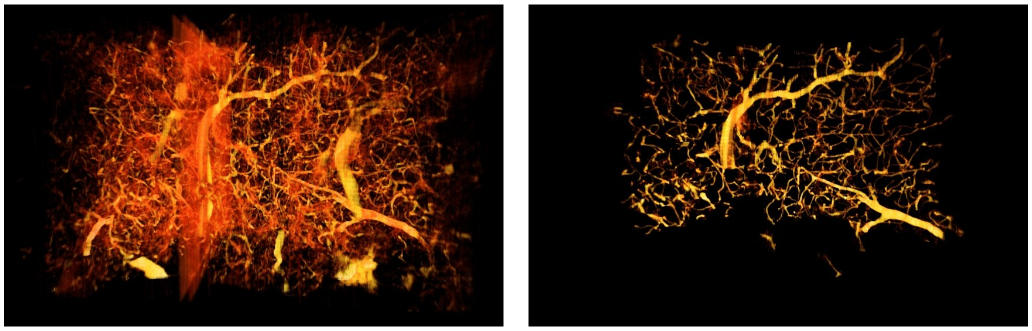


Fig. 38. The original volume visualization of the microvasculature in a mouse brain (left) contains background noise and parts of unwanted cellular structures. Tracing and segmentation extract the desired filament and surface information which can be visualized more clearly (right).

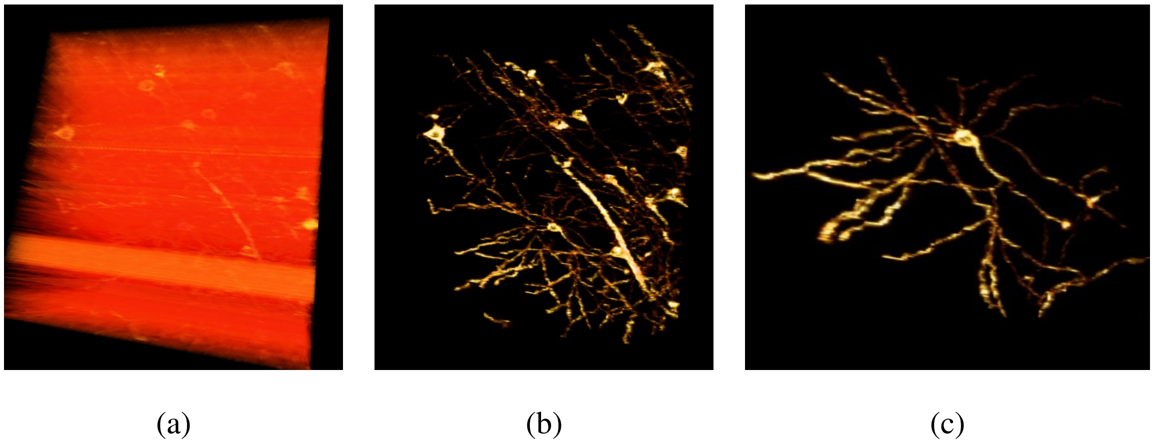


Fig. 39. Microscopic data set containing several neurons. The original volume visualization is completely inadequate and it is impossible to extract a contour surface using standard techniques (a). After tracing, blocking, and contrast enhancement of individual filaments, the structure becomes visible (b). Visualization of a filament network representing a single cell (c).

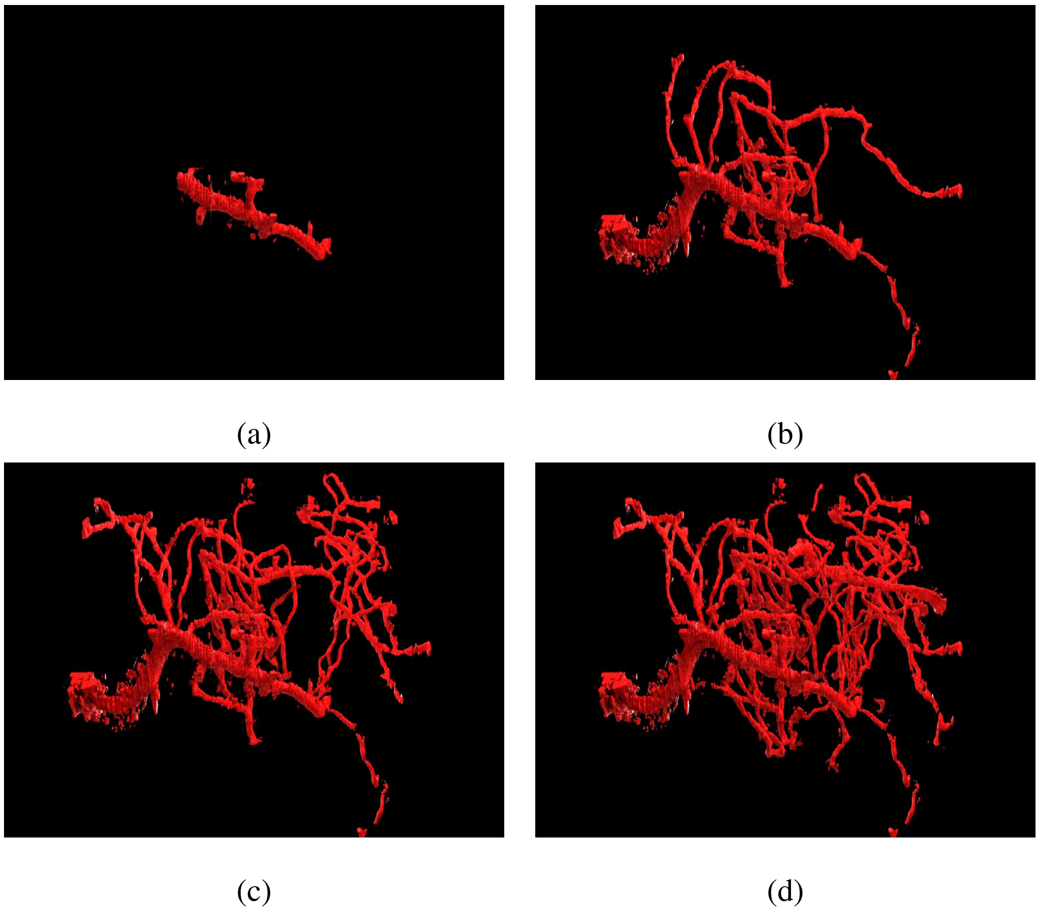


Fig. 40. Contouring of filaments selected using a breadth-first search of the network starting from a single user-defined filament(a) - (d).

4.6. Conclusions and Future Work

In this chapter we propose a framework for segmenting filament networks from volumetric data sets produced using high-throughput microscopy, enabling more effective visualization and analysis. This allows us to visualize structures, such as fine surface details, that would not be visible by rendering the filaments as imposters [60] or streamlines [85]. Our main goal is to provide a fast and automated segmentation algorithm, making it possible to classify volumetric data associated with a complex network in large data sets. Finally, we have shown that our framework is general enough to work with data sets from more

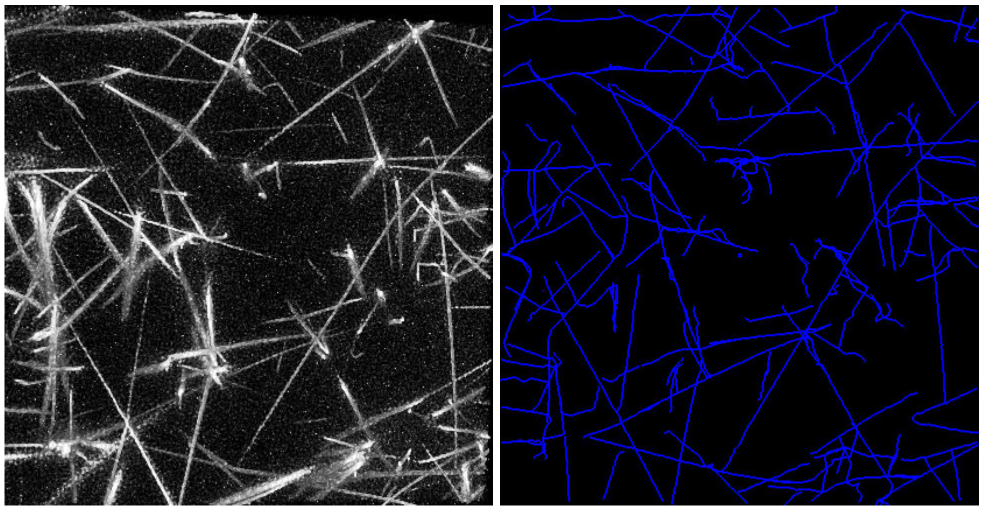


Fig. 41. Fibrin is a protein associated with blood clotting. (left) Volume visualization of a fibrin network and (right) associated tracing.

standard forms of medical imaging.

Our tracking algorithm gains significant benefits from the parallelism in modern graphics hardware, particularly through the use of GPU texture units for sampling. We can gain further efficiency by limiting the number of samples taken based on *a priori* knowledge of our data set. For example, the medial axis of a blood vessel tends to curve less as the radius of the vessel increases. We could therefore reduce the number of samples taken as the vessel radius increases. Since our samples are stored in a display list, it would be inefficient to do this for every filament. We could, however, sort seed points and trace them in order of ascending radius. This would allow us to increase the number of samples at a finite number of intervals while tracing an entire data set.

Although these algorithms provide a basis for segmenting and analyzing filament networks, we have not addressed methods for dealing with errors in the network structure. For

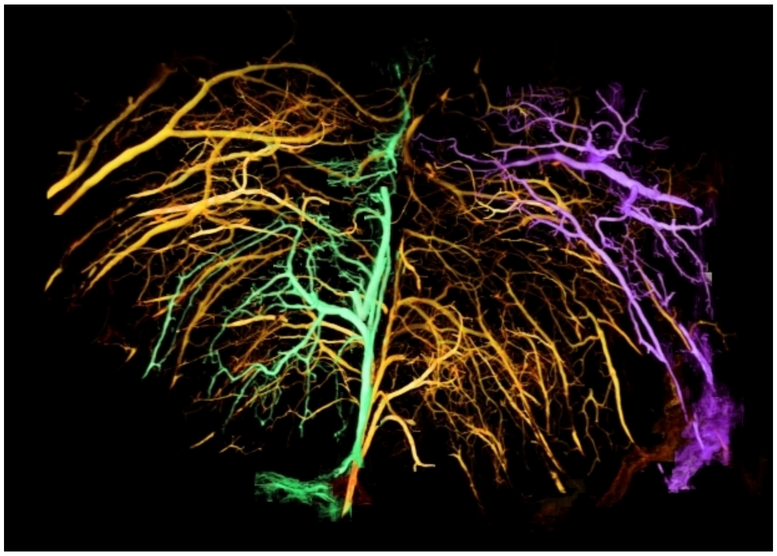


Fig. 42. Traced and segmented microvasculature of the mouse spinal cord. The entire network has been traced and displayed (orange) as well as several sub-networks (green and purple) represented as cliques within the network.

example, our algorithm determines network connectivity based on proximity, which is not always an accurate method. 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 two filaments passing close to one another.

Finally, there are several avenues of future work remaining for applying these techniques to interesting problems. For example, the structure of brain microvasculature has been shown to have an effect on neurodegenerative diseases. Our techniques can be used to create models of blood flow through microvessels and provide more accurate biological models of neural networks.

CHAPTER V

MICROVASCULAR MODELING

Complete models of vascular structure are important for understanding several medical conditions. Brain microvasculature has been shown to play an important role in disorders and neurodegenerative diseases including Alzheimer's Disease, Multiple Sclerosis, and Parkinson's Disease [95]. Capillaries, the basic components of the microvascular system, perform important nutritional functions and may also affect the neural response [62]. However, very little is known about the structure of microvascular networks. This is due both to their small size and extraordinary complexity.

Microvascular networks have several properties that make them difficult to both image and model. The capillaries are $\approx 5\mu m$ in diameter, requiring high-resolution imaging for reconstruction. Despite their small diameter, connected components in a capillary network span several cubic millimeters of tissue. Creating complete data sets representing microvascular networks requires imaging entire organs at a microscopic resolution. Advanced imaging methods and segmentation techniques are required in order to cope with these large and complex data sets.

In this chapter, we describe how KESM imaging and tracking algorithms can be used to create high-resolution microvascular models. First, we create a complete image of the mouse brain vascular system in high-resolution using KESM. We then use tracking methods (Chapter IV) to find the medial axis of capillaries that make up each network. In Section 5.3, we describe a method for combining topological information with an incomplete or damaged isosurface in order to create a model useful for statistical analysis and simulation. In Section 5.4, we use this model to perform a high-resolution statistical analysis of several important features of the mouse microvascular system. This chapter describes two major contributions. First of all, we create a whole-brain image of the mouse vascular system at the microscopic scale, as well as an image of the somatosensory cortex containing both vascular and cellular information (Sec. 5.1). We also develop a technique for using mutual information from both the network skeleton and isosurface to create compute radii for high-resolution structural models of the microvascular network.

5.1. Imaging

In addition to KESM, several three-dimensional techniques currently exist for imaging the vascular system at the macroscopic level. Magnetic Resonance Imaging (MRI) and Computed Tomography (CT) are commonly used for imaging vascular structure. Although these methods are non-invasive, the sampling resolution is insufficient for reconstructing a capillary network.

Micro-CT is often used for imaging capillary networks [42], however the resolution is generally limited to $8\mu m - 20\mu m$, while the average microvessel diameter in many mammals drops below $4\mu m$ [41, 63]. Methods such as synchrotron radiation Micro-CT (SRμCT) [30] have been used to reach resolutions of $1.4\mu m$, however the imaging time is prohibitive for entire organs such as the brain. Finally, CT vascular imaging cannot be used to image additional structures, such as cells, in the surrounding tissue.

Other microscopy techniques, such as confocal microscopy [70], are sufficient for resolving the three-dimensional structure of capillary networks but are dependent on light penetration into the tissue and are therefore limited to the specimen surface.

5.1.1. Imaging Methods

We created two large volumetric data sets based on different staining procedures for brain microvasculature. We first created a high-contrast whole-brain data set by perfusing India ink through the mouse circulatory system. This is a well-known stain that dyes the microvessels black (Figure 43). Although this method is high-contrast, it does not provide any information about the tissue surrounding the microvascular system.

As stated previously (Section **??**), physical sectioning destroys the tissue as it is imaged. Therefore, in order to gather as much information as possible during a single imaging pass, we create an additional data set by imaging tissue stained with Nissl [1]. This stain labels all cells and extracellular tissue in the brain while the vasculature remains unstained (Figure 43). This provides a means of associating cellular information with the vascular structure, although the contrast between the unstained capillaries and the surrounding tissue is greatly reduced.

The whole-brain data set took approximately two weeks to image and requires 2TB of uncompressed storage. We limited the Nissl-stained data set to a large region of somatosensory cortex. This region is important because it allows us to see the change in vasculature and cellular density between white and grey matter. For the whole-brain data set, we focus on regions of high vasculature, including the spinal cord and cerebellum (Figure 45a and b). For the Nissl stained brain, we examine the somatosensory cortex and underlying white matter (Figure 45c).

5.1.2. Image Processing

Both random noise and lighting artifacts are present in KESM images. This is due to both lighting conditions and mechanical vibrations. Lighting artifacts are removed using a median-based destriping algorithm [54]. This technique is fast and preserves features in the data set, but does not eliminate noise. Since the diameters of microvessels are near the resolving power of the microscope, noise and changes in contrast result in frequent gaps and artifacts in the network isosurface (Figure 44). In addition, standard image processing techniques such as median filtering and blurring [25] can further damage the network isosurface. Therefore,we elect not to use blurring for noise removal, which may cause

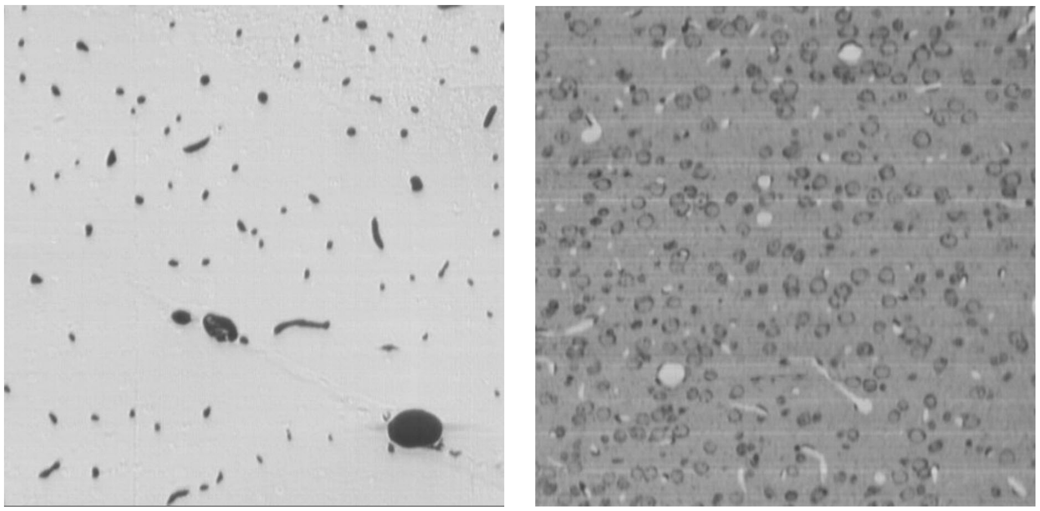


Fig. 43. Cropped samples of tissue stained with India ink (left) and Nissl (right). Vascular filaments are unstained when the tissue is prepared with Nissl, and stained black with an India ink perfusion.

data loss and introduce further breaks in the network. Instead we rely on segmentation algorithms that are robust in the presence of noise.

5.2. Extracting Network Structure

Given a three-dimensional data set representing a tissue sample, we create a graph describing the structure of the embedded filament network. We do this by finding the internal medial axis of each filament, including points where the filaments are connected.

5.2.1. Previous Work

There are several segmentation tools available for tracking macroscopic vascular structures, such as those found in MRI and CT data sets. An overview of these methods is presented by Kirbas et al. [35]. These techniques often rely on centerline extraction from an isosurface [86], region growing [64], or template-matching [25]. In addition, filtering techniques can be used to enhance the quality of segmentation for linear and curvilinear objects [80].

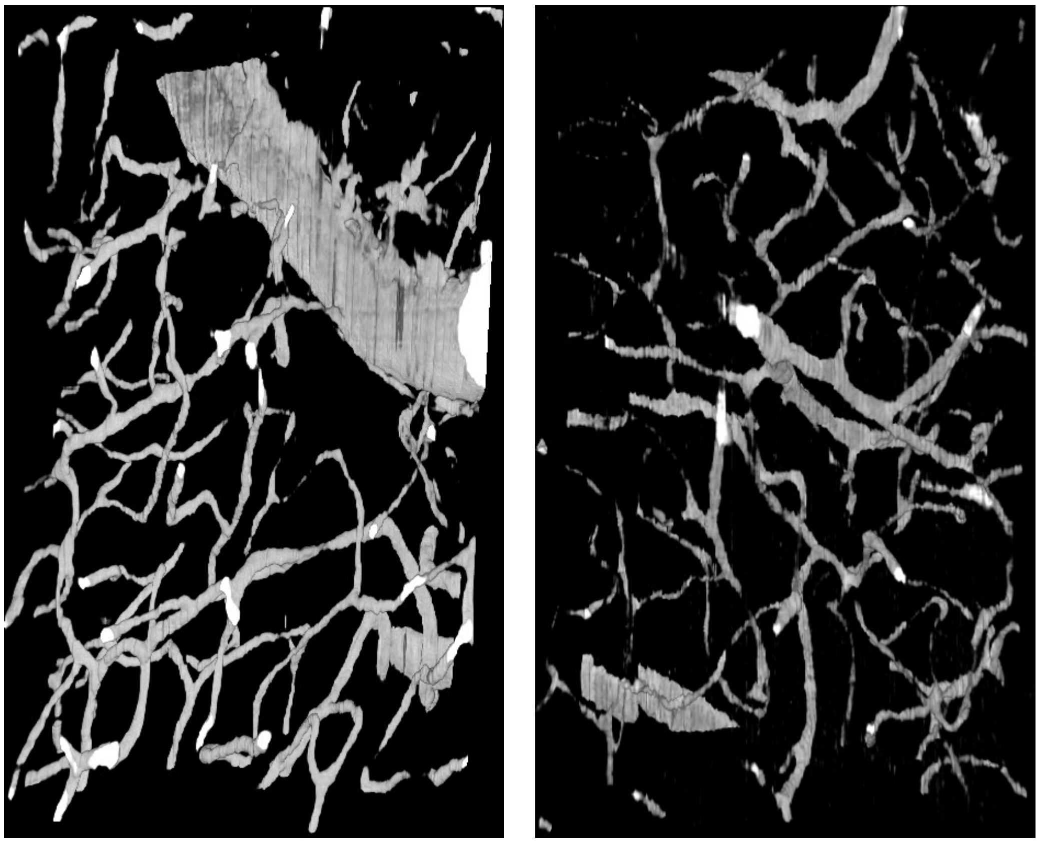
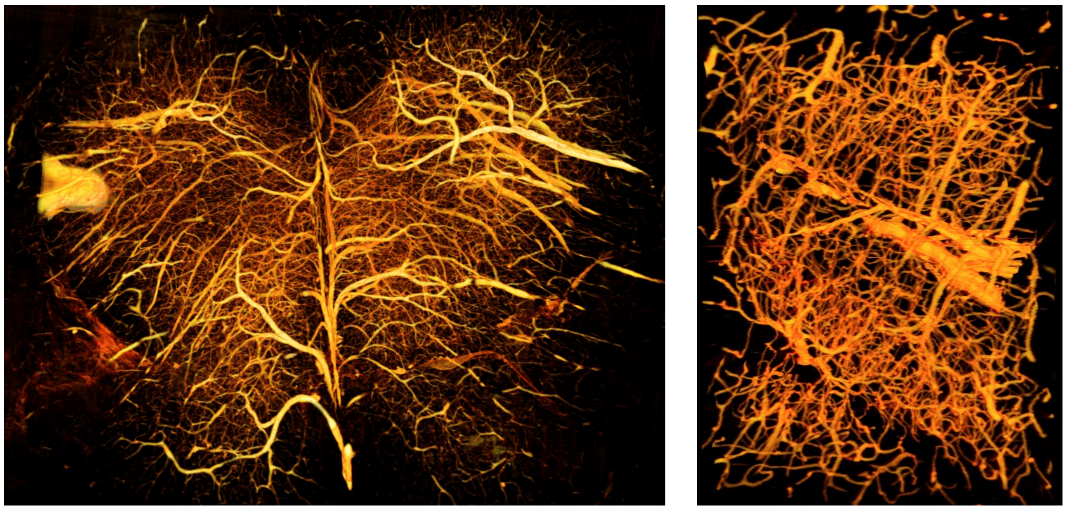
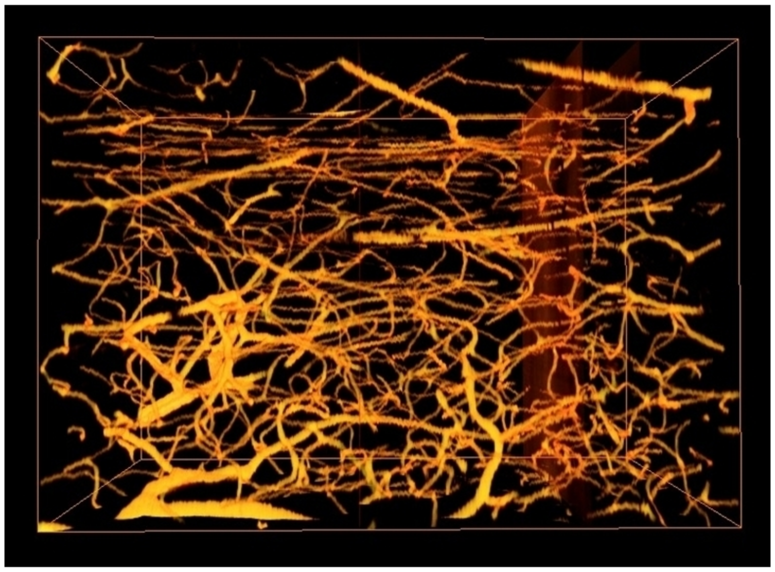


Fig. 44. Cropped isosurfaces from India ink (left) and Nissl (right). Because of noise and low contrast, the network isosurfaces contain frequent gaps, making it difficult to accurately reconstruct network connectivity and topology.

As mentioned previously (Section 5.1.2), the isosurface can be damaged due to noise while features in the surrounding tissue can produce over-segmentation when using thresholding. Although region growing methods are effective for finding the filament surface, they generally require some initial segmentation in order to converge to a meaningful result. Although we have found template matching methods to be robust in the presence of noise, these techniques involve testing various template sizes and orientations with each voxel in a data set. This is computationally expensive since the template must be tested in several positions, orientations, and sizes.



(a) (b)



(c)

Fig. 45. Maximum intensity projection of the mouse spinal cord (left, 1500 sections) and close-up views of a capillary network in the cerebellum (center, 512x512x512) and the neocortex (right, 512x512x512).

5.2.2. Segmentation

Vector tracking methods [3,12,38,90] rely on template matching local to the region around a filament. The degree to which the filament cross section matches the provided template is used to estimate the filament trajectory. This estimated trajectory is then used to reduce the number of template sizes and orientations tested.

Given a point that lies on a filament, vector tracking algorithms predict the trajectory of the filament by sampling a region around this initial point. We then step along the filament in the direction of the estimated trajectory, thereby traversing the filament axis. Vector tracking generally uses three-dimensional template matching and is therefore robust in the presence of noise and broken filaments. Unlike standard template matching, we use a heuristic to limit sampling to a region near the filament. Since microvasculature occupies a very small volume of the data set (Section 5.4), the number of samples required is greatly reduced. In this section, we describe our vector tracking methods, while further details can be found in our previous work [55].

5.2.2.1. Heuristics

Given a point p_i on the filament, we predict the next point p_{i+1} along the filament axis by sampling the region around p_i using a template. We look for the optimal transformation matrix T that minimizes the heuristic function

$$h(T) = \int_x \int_y \int_z |\Phi(T\mathbf{x}) - \gamma(\mathbf{x})| d\mathbf{x} \tag{5.1}$$

where Φ is the volumetric data set, γ is a template function, and the point $\mathbf{x} = [x, y, z]$ is a point on the template. The matrix T is composed of affine components that describe the position, orientation, and size of the template:

$$\mathrm{T} = \mathrm{Tr} \times \mathrm{R} \times \mathrm{S} \tag{5.2}$$

In this equation, Tr, R, and S are affine transformations respectively representing a sampled position, orientation, and scale of the template γ.

5.2.2.2. Tracking

We find the minimum value of the heuristic function $h(T)$ by sampling a discrete set of transformations. We construct Tr based on the initial position p_i:

$$Tr = \begin{pmatrix} 1 & 0 & 0 & p_x \\ 0 & 1 & 0 & p_y \\ 0 & 0 & 1 & p_z \\ 0 & 0 & 0 & 1 \end{pmatrix} \tag{5.3}$$

We then construct a set of rotation matrices $\mathbf{R} = [R_0, R_1 ... R_N]$ that orient the template along an associated set of sample directions $\mathbf{r} = [r_0, r_1 ... r_N]$. We use a template that is rotationally invariant along the direction vector r_i. Therefore, the associated orientation matrix R_i can be composed using any two other vectors orthogonal to r_i. Likewise, we construct a series of scale matrices $S_i = s_i I$ where $s_i = [s_0, s_1 ... s_M]$ and I is the identity matrix. By sampling combinations of R and S, we select the samples that minimize the cost function $h(T)$. The position of the template is updated by taking a step along the estimated filament trajectory based on the selected orientation vector r_i, which produced the minimum value of h.

We use a volumetric cylinder as a template since it is both rotationally invariant along r_i and accurately represents the structure of most capillaries. The orientation and size of the template is adjusted as it is moved down each filament, in order to provide an accurate match (Figure 46). Although the template size can be used as an estimate of the filament radius, it is highly dependent on how well defined the surface is relative to the background. As each step along a filament is taken, the current position p_i is tested against previously traced filaments in order to detect intersections. Further details on efficient filament tracking methods can be found in Al-Kofahi et al. [3] as well as our previous work on hardware-accelerated techniques [56].

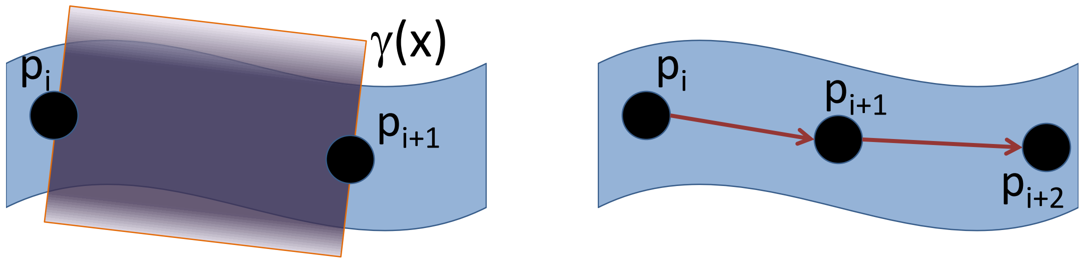


Fig. 46. (left) We track the filament axis by matching a template function γ to the filament. (right) After computing the optimal template orientation (Eq. 5.1), we take successive steps along the filament axis.

5.2.2.3. Seed Points

Filament tracking requires us to create initial seed points on each filament from which to initiate tracking. We place seed points using the method proposed by Al-Kofahi et al. [3]. We project a region onto a two-dimensional plane using a maximum intensity projection for Nissl or a minimum intensity projection for India ink. Seed points are then placed on the plane using a conservative threshold and projected back into the three-dimensional region. Note that we are not concerned with over-seeding since branch detection will remove excess seed points.

5.3. Refinement

Based on the filament skeleton found using vector tracking, we construct a graph G representing the structure of the filament network. Nodes connecting two edges in G represent samples along a single capillary. Branch points within the capillary network are represented by nodes in G with more than two edges (Figure 47a).

Filament radius can then be estimated based on the optimal size of the template estimated during each tracking step, or by segmenting the filament cross-section using methods such as active contours [42]. When basing the radius estimate on the template size, we are limited to the discrete number of sample sizes used during tracking. In addition, these radii are dependent on the estimated trajectory of the filament in both cases. Therefore, small errors in filament trajectory can result in a cross-section that is not orthogonal to the true filament trajectory. Another option is to evolve a level set surface outward from the skeleton [49], however both active contours and level sets require several parameters to optimize fitting the resulting curves or surfaces.

Our approach instead relies on the network isosurface. Since our sampling resolution is sufficient to resolve the smallest vessels, there is little fear of misclassifying large vascular segments due to undersampling. As mentioned previously, however, the network isosurface contains structural flaws that make it difficult to determine network topology and connectivity. However, the isosurface does contain useful information about the surface structure and diameter of each filament in the network. In this section we discuss how we refine the network with morphological information from the microvascular isosurface, providing radius information for anatomical studies.

Given our original data set represented as the scalar volume function Φ, we manually select an isovalue that most accurately represents the microvascular surface Γ embedded in Φ. When computing an isosurface from image data, noise and artifacts cause misclassifications of the volume in one of two ways:

- (A) External regions are incorrectly classified as interior regions (false positives). This results in over-segmentation, causing erroneous surfaces to be created in the tissue surrounding the filament network.
- (B) Interior regions are incorrectly classified as exterior regions (false negatives). These errors result in unusually thin filaments or gaps in the network.

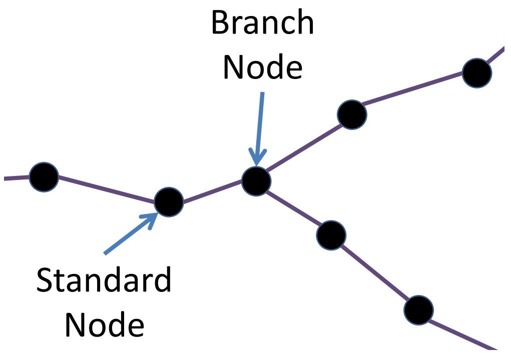
We compute the capillary radii for our vascular model by using mutual information between the extracted skeleton (Sec. 5.2) and the network isosurface. By mapping isosurface information onto the network skeleton, we can find many of the erroneous cases produced by misclassifications and edit them out of our final model.

5.3.1. Mapping

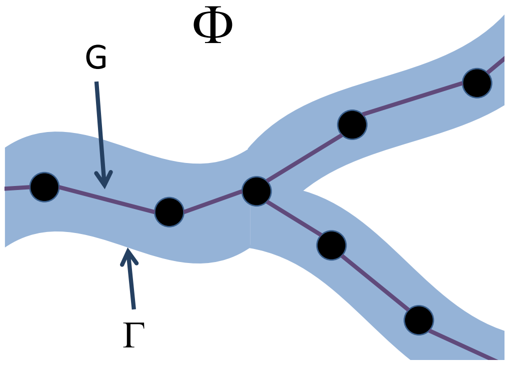
We first create a mapping between the network skeleton and the data set. By overlaying the graph G, representing the network skeleton, with the scalar volume function Φ, we create a direct mapping $G \Rightarrow \Phi$ where any point on a node or edge of G represents a three-dimensional position within the data set Φ (Figure 47b). We then construct an implicit signed distance function F_{sdf} based on the data set such that $\Phi \Rightarrow F_{sdf}$ and therefore $G \Rightarrow F_{sdf}$. We use the standard definition of a signed distance function:

$$F_{sdf} = \begin{cases} d_\Phi(x,\Gamma) & \text{if } x \text{ is outside } \Gamma \text{ and} \\ -d_\Phi(x,\Gamma) & \text{if } x \text{ is inside } \Gamma. \end{cases} \tag{5.4}$$

where x is a point on G and $d_\Phi(x,\Gamma)$ is the shortest distance between x and the isosurface Γ.



(a)



(b)

Fig. 47. (left) The traced graph G contains standard and branch points indicating positions on the filament axis. (right) We create a mapping from G to Φ by overlaying the traced graph onto the volumetric function.

Note that, if the function Φ is noise-free and all points in the graph G lie exactly on the filament axis, the radius r at any point x on G can be found by looking up the associated point in F_{sdf}:

$$r = -F_{sdf}(x) \tag{5.5}$$

We will first discuss efficient methods for creating the signed distance function F_{sdf} and then describe how G is refined in order to find the radius of each point in the network.

5.3.2. Computing a Distance Field

We compute an implicit signed distance function by solving the Eikonal equation $|\nabla F| = 1$ on a discrete grid. There are several efficient methods available, including Fast Sweeping [94] and Fast Marching [83]. Fast Sweeping provides an $O(n)$ solution, but requires that every voxel be evaluated. Fast Marching can be done in $O(n \log n)$ time but allows us to march outward from a surface and stop computation when necessary. We note that it is only necessary to solve the negative, or internal, portion of F_{sdf} since the radius information exists inside the surface Γ specified by the zero level-set of F_{sdf}. Since the volume inside Γ is usually much smaller than the volume of the data set (see Sect. 5.4), we use a single pass of Fast Marching to evaluate the distance field only inside the network isosurface.

We first initialize F_{sdf}. Grid points next to the surface Γ are initialized with the distance from the surface while all other grid points are set to some large positive value. We then march inward computing the distance function for all values inside Γ (Fig 48). A detailed description is given by Osher and Fedkiw [69].

5.3.3. Radius Computation

We create an initial estimate of the filament radius by resampling F_{sdf} at points that lie on G. By limiting sampling to points on or near nodes and edges in G, we greatly reduce noise

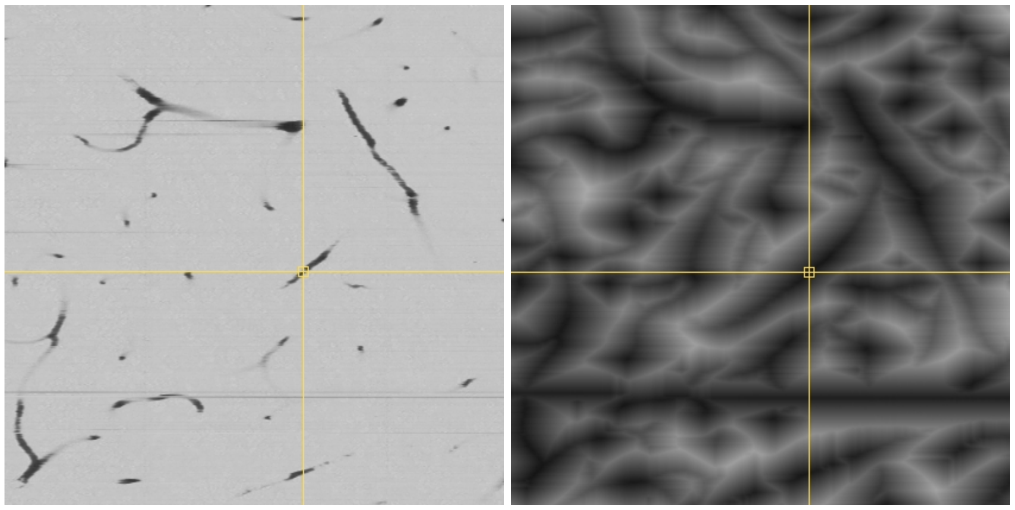


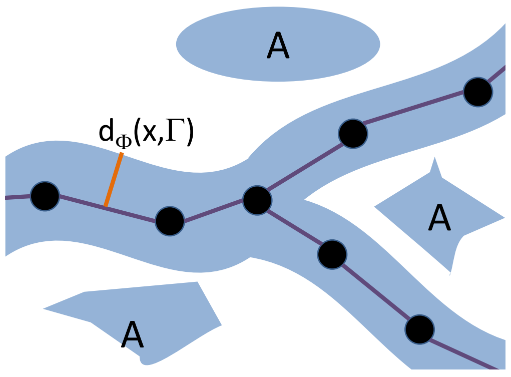
Fig. 48. Orthogonal section from the cortical data set (left) and equivalent section from the function F_{sdf} (right). For clarity, F_{sdf} has been fully evaluated and scaled to [0...255].

due to misclassifications of type A (false positives). This is because false positives, by definition, occur outside of the network while the distance values in F_{sdf} are always based on the closest point on Γ (Figure 49a).

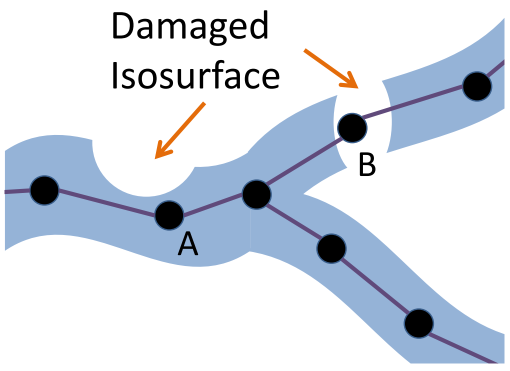
Misclassifications of type B (false negatives) give erroneous results by causing interior regions to be labeled as exterior. This causes breaks in the filament or makes the isosurface unnaturally narrow along a portion of a capillary (Figure 49b). When these regions are sampled, r is either very small or negative. These misclassifications are resolved by propagating known values from neighboring nodes in G.

Since the graph G is based on a heuristic estimate of the network embedded in Φ, it is unlikely that all points on G lie exactly on the filament axis. Therefore, sampling exactly on G leads to under-estimating the capillary radius since the corresponding point in F_{sdf} may lie closer to the filament surface. Note that the actual filament axis lies on the point inside the filament that is furthest from the filament isosurface. In F_{sdf}, this surface is represented by the zero level-set. Therefore, when sampling F_{sdf} using a point $\mathbf{x}$ on G, we instead sample a region in F_{sdf} that lies within ε of $\mathbf{x}$.

We set the radius in G equal to the maximum value of r found within ε of $\mathbf{x}$. We set ε to $5\mu m$, which is close to the known radius of microvessels. Since microvessels are sparsely arranged relative to their diameter, it is unlikely for sampling to cross into other capillaries using this value for ε.



(a)



(b)

Fig. 49. We use the graph G to interpolate through errors in the isosurface. (left) False positives cause erroneous regions to be formed outside the isosurface. These regions are avoided by only sampling F_{sdf} near the graph. (right) Points in G with very small (A) or negative (B) values of F_{sdf} indicate isosurface damage. The radius for these points are computed by propagating valid information about F_{sdf} from neighboring points.

5.4. Results and Discussion

We first imaged an entire mouse brain stained with India ink at a resolution of $0.6\mu m \times 0.7\mu m \times 1.0\mu m$. Images were processed to remove lighting artifacts (Sect. 5.1) and stored as 512x512x512 raw image files. We then specified several brain regions of interest for analysis. We particularly focused on the cerebellar cortex and spinal cord (Figure 45a and b). We also imaged a large region of mouse cortex stained with Nissl, processing and storing it in the same manner (Figure 45c).

5.4.1. Computational Anatomy

We used vector tracking (Sect. 5.2) to perform automated segmentation of the network. We then collected several anatomical statistics on the structure of the network, such as the number of segments and branch points per unit volume of tissue (Table III). We also focus on obtaining local anatomical information such as the direction of travel of capillaries (Figure 50). Statistics and anatomical information are of interest to anatomists since very little information is available on the three-dimensional structure of microvasculature. Although we are unaware of any studies performed at this scale and resolution, our results are similar to morphometric studies that have been performed using samples of SRμCT-collected data for the mouse cortex [30] and confocal microscopy images of human brain [41].

Table III. Computed statistics for the capillary network model based on a refined vascular network. We define a *segment* as a length of capillary between two branch points. All statistics are specified per cubic millimeter of tissue.

Region	Segments	Length (mm)	Branches	Surface (mm^2)	Volume (mm^3)	Volume (% of total)
Neocortex	11459.7	758.5	9100.0	10.40	0.0140	1.4%
Cerebellum	34911.3	1676.4	19034.4	20.0	0.0252	2.5%
Spinal Cord	36791.7	1927.6	26449.1	22.2	0.0236	2.4%

5.4.2. Modeling

Using the structural information available from tracked filaments, we refine the network based on the filament isosurface. This involves manually selecting an isovalue that accurately describes the vascular surface (Figure 51a). Although the surface is noisy and contains misclassifications, we map the isosurface information onto the network skeleton. Using our refinement techniques (Sec. 5.3), we create a structural model of the vascular

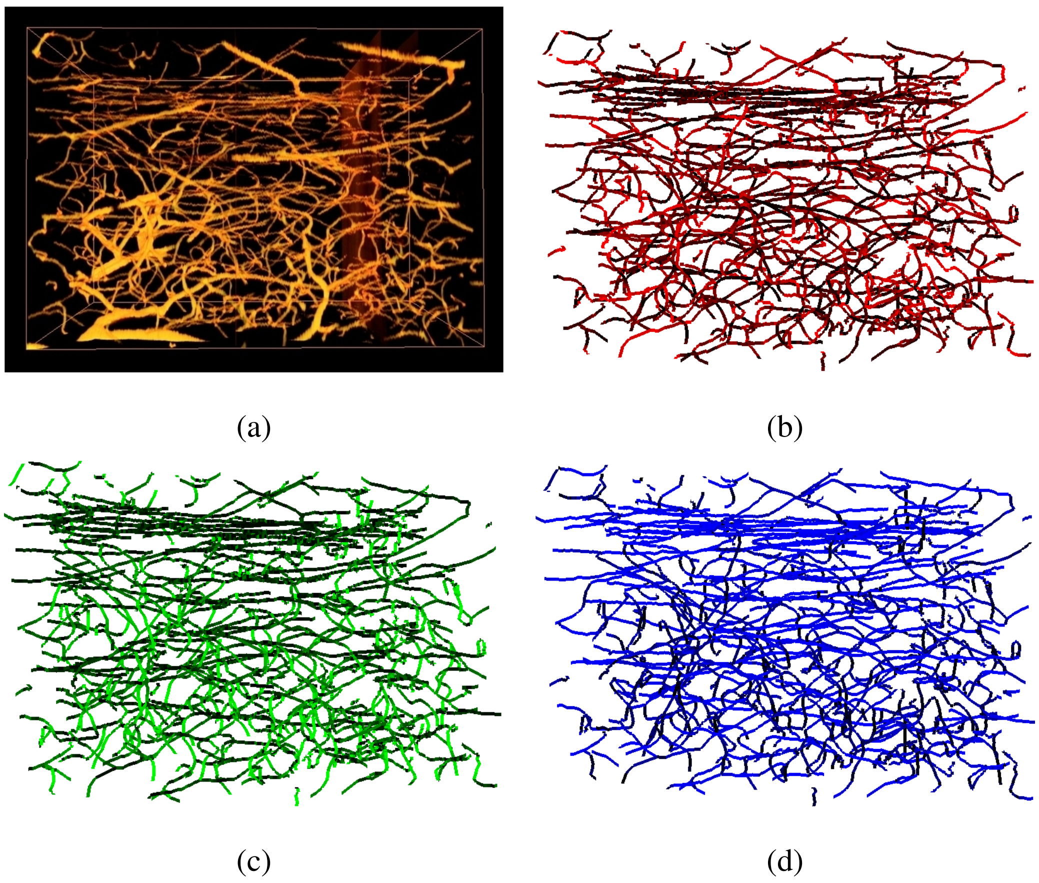


Fig. 50. Vascular trajectory in the neocortex. The tracked vascular network from a single block (Figure 51) is shown with filaments colored based on their direction of travel (red = sagitally, green = horizontally, blue = coronally). The highly oriented blue filaments are running coronally through white matter.

network that describes filament radius as well as position and connectivity. We then use the refined model to compute the microvascular volume and surface per unit volume of tissue (Table III).

5.4.3. Evaluation

We have limited means for comparison and evaluation due to a lack of similar high-resolution data sets. We instead focus on the evaluation of vector tracking as a means of segmentation. Although our algorithm performs well under a visual evaluation, the

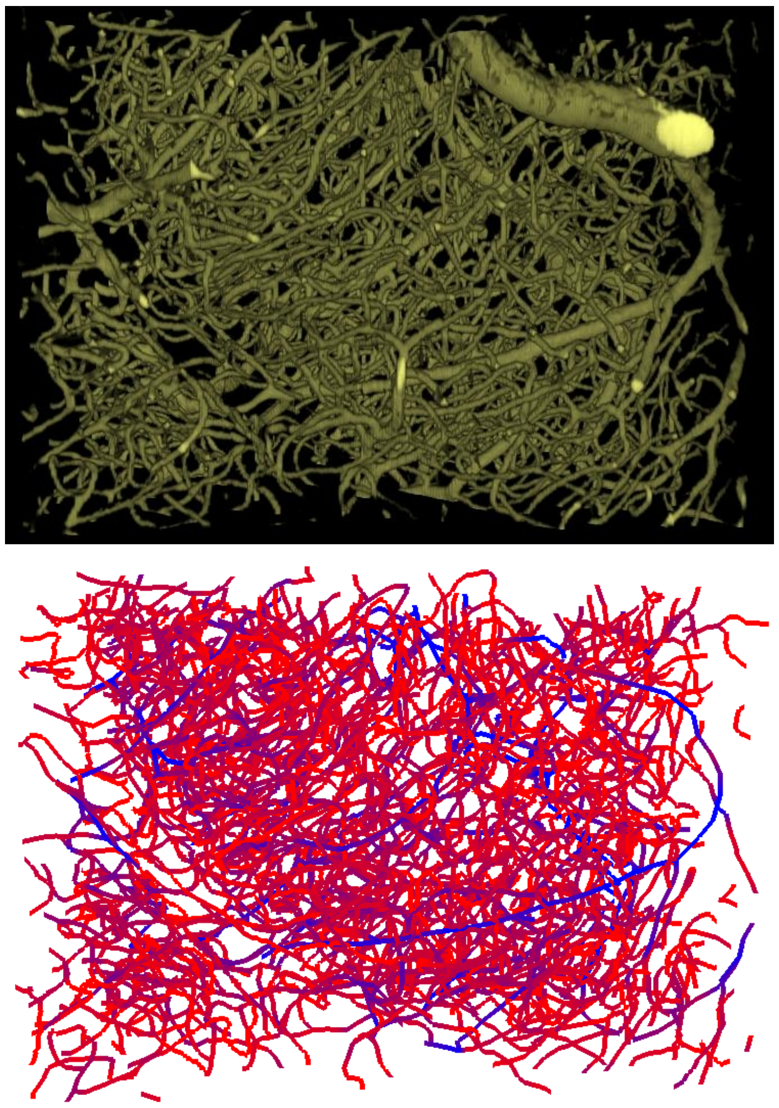


Fig. 51. Cerebellar cortex. Vascular isosurface (left) used for refining radius measurements. Traced vascular network (right) colored based on radius (red $< 2\mu m$, blue $> 5\mu m$).

complexity of large microscopy data sets makes errors difficult to find visually.

Because of the nature of microvascular networks, all segments must be connected in order to allow blood flow. We consider any traced capillaries that terminate without branching to be errors. These terminations, accounting for $\approx 3.2\%$ of the network, could be due either to errors in tracking or inadequate staining. However, changes in brain microvasculature are known to occur and some portion of these may be developing or degenerating microvessels.

5.4.4. Discussion

In this chapter we describe a framework that provides high-resolution information about the structure of microvasculature spanning large three-dimensional regions of tissue. Our analysis focuses on mouse brain microvasculature, however similar staining methods have been used on other tissue samples from other species.

Further work can be done by extending these models to pathological tissue. This would allow high-resolution analysis of blood-flow patterns in different disease models. Although frameworks for fluid simulation in microvessels have been developed [74], these methods have not been extended to larger scales due to a lack of high-resolution structural information. Finally, very little research has been done in the area of simulating cellular and vascular relationships, which can be observed in Nissl-stained tissue. Modulation is known to occur between neurons and microvasculature. Simulation of these relationships would require detailed cellular morphology as well as microvascular anatomy.

CHAPTER VI

FILAMENT VISUALIZATION

6.1. Selective Volume Visualization

6.1.1. Introduction

The vertebrate vascular system allows transport of oxygen and other small molecules to cells throughout an organism. Several techniques exist for constructing volumetric images of the vascular system at the macroscopic scale. Magnetic Resonance Imaging (MRI) and Computed Tomography (CT) are often used to image vasculature in living tissue. However, these imaging modalities are incapable of resolving *microvasculature*. Microvasculature is the network of capillaries that complete the loop between arteries and veins. In contrast to the tree-like topology of larger-scale vascular structures seen in traditional biomedical data, microvasculature tends to have a network structure. Also, while large-scale structures serve primarily a transport function, microvasculature more directly exchanges molecules with surrounding tissues.

New developments in microscope technology allow imaging of these complex microvascular networks. A major difficulty with visualizing complex volumetric network information is that the density of the network is prohibitive in understanding its structure. For example, visualizing a small region of a microvascular network reveals little about its structure. In addition, features outside of the microvascular network, such as cell bodies, add to the complexity (Figure 52). Our goal is to selectively visualize the network and associated tissue data. We do this by constructing a graph describing the structure and connectivity of the network. We then encode the volumetric data describing the network into the graph, which can then be queried for interesting anatomical structures. These structures are then selectively displayed to the user.

In this chapter, we make two major contributions. First, we develop a framework for visualizing dense microvascular networks. This framework allows biologists to selectively visualize filaments based on anatomical queries, such as connectivity and distance. We then extend our framework to incorporate the visualization of features in the surrounding tissue, such as cell positions and diameters. These tools provide a novel way of exploring densely packed data sets with large numbers of anatomical features and relationships.

6.1.1.1. Motivation

The vertebrate vascular system is formed by a cyclical network of arteries and veins, carrying oxygen rich and oxygen poor blood respectively. Starting from the heart, they branch into smaller and smaller vessels. At the microscopic scale, the smallest blood vessels, known as *capillaries* (or microvasculature), form a meshwork that connects veins and arteries and delivers nutrients to nearby cells.

The structure and function of microvasculature is important, particularly for neurological tissue. Changes in brain microvasculature have been linked to several chronic conditions such as Alzheimer's Disease, Parkinson's Disease, and Multiple Sclerosis [95]. The relationship between cellular and vascular tissue is also of interest in tumors and other chronic tissue.

The primary interest in all of these cases is the degree of access tissue has to blood-borne molecules. In the brain, the diffusion of these molecules is limited by the Blood-Brain Barrier (BBB), a layer of cells that tightly control molecular diffusion in the brain. Although there are means of active transport through these cells, the primary means through which tissue receives chemical input from the vascular system is through diffusion of molecules from capillaries (Figure 53). Therefore the distance between the capillary surface and surrounding tissue is an important factor describing the effect a capillary has on tissue.

Finally, high-resolution surface details are important for understanding microvascular structure and its relationship to cells. In particular, rendering the capillary surface allows us to explore regions of angiogenesis, where microvasculature is undergoing changes through the formation of new capillaries or the destruction of existing ones. There is also a close relationship between the capillary surface and certain types of cells. For example, endothelial cells play an important part in defining the capillary surface. Visualization of this mutual relationship is difficult using simplified representations of vasculature commonly used for macroscopic data [8, 21, 57].

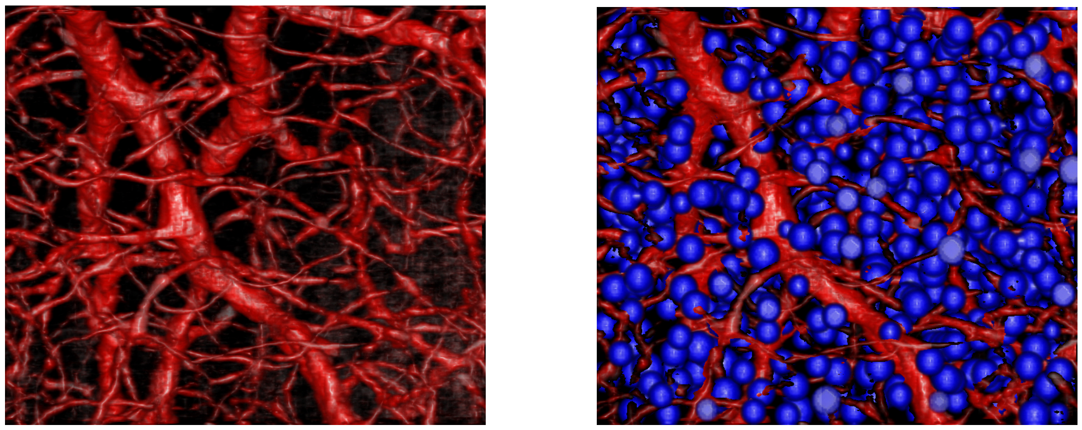


Fig. 52. Despite the density and complexity of the vascular isosurface, the volume occupied by the network is less than 3% of the total volume. Much of the remaining volume is occupied by cell bodies (right) and extracellular tissue.

6.1.1.2. Imaging

In this section, we briefly discuss the imaging methods used to create microvascular and cellular data sets. The advent of high-throughput microscopy allows researchers to quickly produce large volumetric data sets representing high-resolution biological tissue. Knife-Edge Scanning Microscopy (KESM) [52] is capable of imaging large specimens at microscopic resolution producing data at a rate in excess of 100MB/second. Imaging entire organs such as the mouse brain produces several terabytes of data. At this resolution biological tissue is highly complex, containing densely packed high-frequency structures.

In this chapter, we use data acquired from KESM scans of rat brain. In order to image tissue using light microscopy, the tissue samples are stained and then imaged. In particular, cell bodies (soma) are stained (dark). During the staining process, all blood is flushed from the capillaries in the tissue. The microvasculature is therefore visible as unstained filaments (light) (Figure 54).

6.1.2. Segmentation

Although our primary focus in this chapter is the visualization of filament networks, we briefly discuss the segmentation methods used to extract the capillary network and cells. Filament structures such as microvasculature are inherently difficult to segment. This is due to low overall contrast, significant interfering (cellular) data, and image noise. In addition, thin filaments often drop below the resolving power of the microscope, causing gaps in the network. Naive methods such as isosurface reconstruction yield misclassifications, causing gaps in the network and erroneous surface details (Figure 55).

Many filament tracking algorithms have been developed for medical imaging methods such as X-Ray and MRI. An extensive review on the subject is done by Kirbas, et al. [35]. Methods that rely on the isosurface of the network [?, 86] work well for thick structures but frequent gaps and misclassifications make locating the medial axis difficult since there is no easy way to locally differentiate between noise and structure. Template matching [80] is effective in the presence of noise and gaps in the network, however this requires matching scaled and oriented cylinders in a three-dimensional space, which is computationally expensive for large data sets.

The methods that worked best for tracking these types of thin filaments are medial-axis (or *vector*) tracking algorithms such as those designed for tracking neurons in confocal [3] and serial [55] microscopy. These methods rely only on information local to the filament and can offer significant speedups over template matching. Fast algorithms are important since processing time often exceeds the amount of time required for imaging.

We first extract a skeleton of the microvascular network using a vector tracking algorithm. Full details can be found in Al-Kofahi et al. [3], while techniques for accelerating these algorithms can be found in our previous work [55]. This allows us to establish the positions of capillary centerlines and connectivity. We initialize vector tracking by placing seed points throughout the data set based on a conservative threshold. A template is then placed at each seed point and rotated and scaled to minimize a heuristic

$$h(T) = \int\int\int |\Phi(T\bar{x}) - \gamma(\bar{x})| d\bar{x} \tag{6.1}$$

where Φ is the data set, γ is a template function, the vector $\bar{x}$ is a point on the template. The transformation matrix T is constructed from the position, orientation, and size of the template:

$$\mathrm{T} = \mathrm{Tr} \times \mathrm{R} \times \mathrm{S} \tag{6.2}$$

where Tr transforms the template to the initial position and R and S define the orientation and scaling of the template respectively.

The minimum value of h is found by sampling discrete sets of transformations. We construct Tr based on our initial position. We then sample a series of orientations, searching for a minimum value of h. Finally, we update the size of the template by sampling a series of sizes, continuing the minimization of h. The position of the template is then updated by taking a step along the estimated filament trajectory based on the estimated orientation. In order to segment capillary networks, we use a cylindrical template. The orientation and size of the template is adjusted as it is moved down each filament. Intersections are detected based on the proximity of two segments.

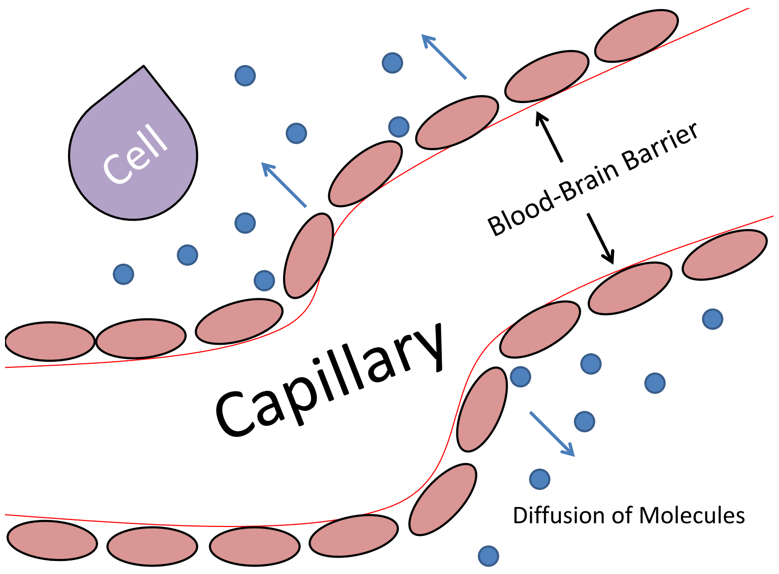


Fig. 53. Small molecules (such as oxygen) used to sustain cells are diffused through the capillary wall and into the surrounding tissue.

Unlike network data, cells vary much less in size and cell bodies are mostly rotationally invariant. Therefore, we can locate cell positions using standard template matching with a small Gaussian correlation kernel [19]. Although this technique can be error-prone, we note that cells exist in vast numbers in biological tissue and highly accurate information is not necessary to demonstrate our visualization methods. We therefore manually select cells in our sample data set in order to provide input to our visualization framework. The segmentation of cellular structures is a well understood problem and several other automated methods are available [46, 59].

6.1.3. Volumetric Encoding

Using the vector tracking techniques described previously (Section 6.1.2), we construct a graph G describing the position (through estimates of the central axis of capillaries) and connectivity of the microvascular network. In addition, the tracking algorithm provides an estimate of the filament radius at each node. Using standard graph analysis algorithms [9],

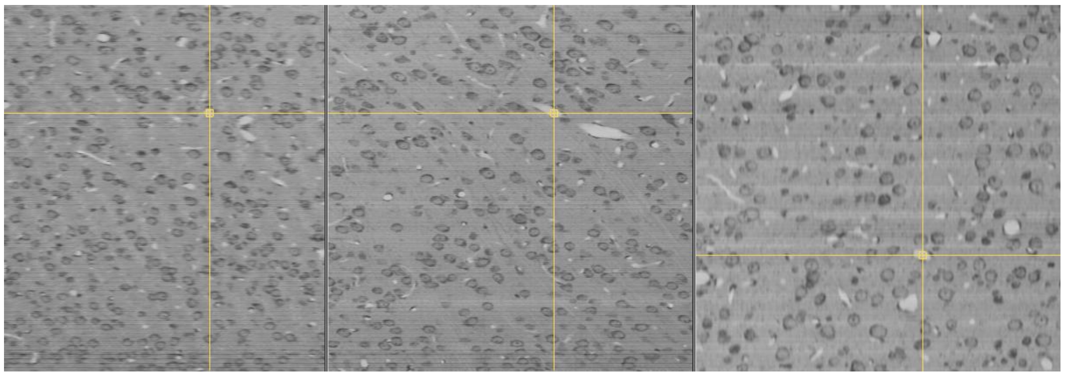


Fig. 54. Orthogonal cross-sections from a 512x512x512 voxel data set. Cross-hairs indicate the same position (within a capillary) in each section. Cell bodies and cell nuclei are dark while surrounding tissue is light gray. Capillaries are unstained.

we now query *G* for structural and statistical information.

Although some basic visualization of the network can be provided using the information in the graph [7], *G* does not contain any volumetric information and the surface structure can only be estimated from the radius of each node. This surface information can be particularly important in physically-based simulations of fluid dynamics through

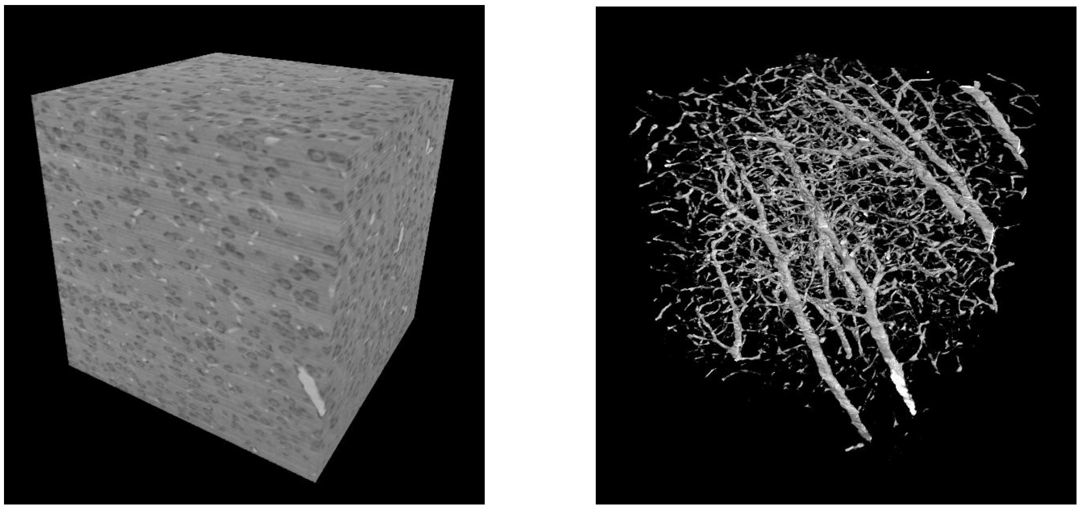


Fig. 55. Orthogonal sections from Figure 54 shown in context (left) and an isosurface representing the microvasculature.

the network. The ability to store volumetric information also extends to other types of networks, such as those created by neurons, which have important surface features. Finally, the volumetric information associated with the vascular network provides more accurate and interesting visualization.

6.1.3.1. Prior Work

Encoding volumetric network data is a unique and challenging problem. Although there are many techniques for creating a bounding volume around the network using structures such as cylinders [50], spline models [**?**], and various other primitives [75], these methods are primarily used for simplified visualization and do not provide any means of encoding the volumetric data. We are aware of only one method [58] for storing data associated with filament structures as a series of attached axis-aligned bounding boxes, although this is primarily used as a means of compression and there are no efficient algorithms for accessing the volumetric data.

6.1.3.2. Dynamic Tubular Grids

The method that we use is based on Dynamic Tubular (DT) Grids [65], which were designed to encode a narrow band of volumetric data around an implicit surface computing advancing fronts [69]. In our case, we use DT-Grids to encode the region surrounding the network skeleton defined by G.

DT-Grids store an array of points representing the volume data in lexicographic order in each dimension $x_1, x_2, ... x_n$. When constructing a DT-Grid, the points must be inserted in this order. Insertions and deletions are not allowed elsewhere in the structure. The data is represented recursively as a series of projections onto a lower dimension. The data itself is stored in a single array in lexicographical order while an underlying data structure keeps track of the coordinates of each value.

DT-Grids have several algorithmic features. Some of the most important for our visualization techniques are:

- Constant time ($O(1)$) access when values are accessed in lexicographical order.
- Constant time access to neighboring voxels when using a template.
- Logarithmic time ($O(\log n)$) random access.

Additionally, DT-Grids provide significant compression when used to store large sparse data sets. Despite the complexity of our volumetric networks, the scalar data describing the network occupies a small percentage of the entire volume. Provided that we limit ourselves to operations on DT-Grids that can be done in constant time, this provides significant speedup over processing the entire volumetric data set. An extensive implementation of DT-Grids and a discussion of their algorithmic properties is given in the chapter by Nielson, et al. [65].

6.1.3.3. Mapping Volume Data to a Graph

Our segmentation and tracing has given us a graph, G, describing the topology of the capillary network. This graph G consists of a set of nodes and connecting edges. Each node in the graph represents a sample point on the network skeleton while each edge describes the connectivity between neighboring points. We first define some terms that are used to refer to anatomical structures in G:

- a *branch* point is a node with three or more incident edges. These points represent branches in the network.
- a *termination* point is a node with only one incident edge. These represent a capillary termination, which can be due to an error in the tracking algorithm, a developing capillary, or a capillary leaving the bounds of the data set.

- a *segment* is a series of edges between branch or termination points. These edge sequences represent a single capillary.

For any point in G, we can look up a corresponding point in the volumetric data, Φ. However, what we actually need is a way of associating larger volumetric regions with the individual components (vertices and edges) of G. This can be a complicated computation, and so we precompute the associations, and use DT-Grids to store this correspondence. We will describe this preprocessing (i.e. storing the correspondence) here.

We apply a method proposed by Mayerich and Keyser [55] to construct a bounding volume around each segment using a series of truncated generalized cones (TGC) (Figure 56). Each TGC is defined by two adjacent nodes along a segment and has a medial axis defined by the edge connecting the points. The TGCs are connected end-to-end with the end caps oriented using normals defined as

$$\mathbf{n_b} = \frac{1}{2}\left(\frac{\mathbf{b}-\mathbf{a}}{||\mathbf{b}-\mathbf{a}||} + \frac{\mathbf{c}-\mathbf{b}}{||\mathbf{c}-\mathbf{b}||}\right) \tag{6.3}$$

where **a**, **b**, and **c** are three consecutive points on the estimated medial axis of a filament. The radius of each end cap is equal to the radius stored at each node in G.

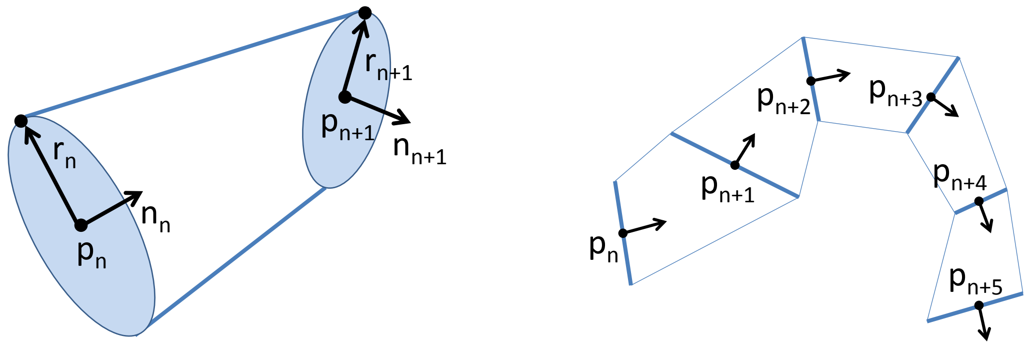


Fig. 56. Truncated generalized cones (left) fit end-to-end to construct the bounding volume for a filament (right).

In order to determine if a given point exists within the bounding volume, we first define the region within a TGC. Assuming that there is a point $\mathbf{p_x}$ under consideration, we want to determine if it 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 6.3 and r_n is the end cap radius (Figure 57). We first find the plane that passes through points $\mathbf{p_x}$, $\mathbf{p_n}$, and $\mathbf{p_{n+1}}$ with normal

$$\mathbf{n_{plane}} = \frac{\mathbf{p_{n+1}} - \mathbf{p_n}}{||\mathbf{p_{n+1}} - \mathbf{p_n}||} \times \frac{\mathbf{p_x} - \mathbf{p_n}}{||\mathbf{p_x} - \mathbf{p_n}||} \tag{6.4}$$

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:

$$\mathbf{p_{r,n}} = \mathbf{p_n} + r_n(\mathbf{n_{plane}} \times \mathbf{n_n}) \tag{6.5}$$

we know that the point $\mathbf{p_x}$ is within the TGC if it lies within the polygon formed by $(\mathbf{p_n}, \mathbf{p_{r,n}}, \mathbf{p_{r,n+1}}, \mathbf{p_{n+1}})$.

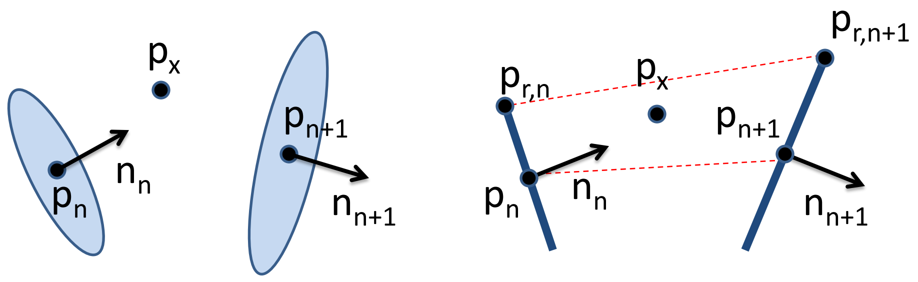


Fig. 57. (a) End caps of a TGC and (b) the intersection of the TGC with the plane defined by $\mathbf{p_x}$, $\mathbf{p_n}$, and $\mathbf{p_{n+1}}$.

Having constructed the TGCs, we now associate all values in the volume image Φ as either inside or outside the bounding volume of G. We wish to avoid comparing all N voxel positions with the TGCs of all M edges, since both N and M are large numbers. Instead, we “rasterize” each TGC onto a new grid, Λ of the same resolution as the original volume Φ. Each point of the grid Λ contains a unique number identifying the segment that the corresponding point in Φ is associated with. For each point inside the bounding volume of a filament, we write an identifier indicating the associated segment in G. Due to possible overlaps between neighboring filament bounding volumes, particularly at intersections, the values of Λ may be overwritten multiple times. Since the vascular volume is relatively small, this generally only happens at intersections and poses no problem for visualization and storage. In fact, this eliminates redundancy in the encoding scheme since each voxel in Φ is associated with at most a single network segment. We therefore ensure that the minimum amount of data is stored in the resulting DT-Grids based on the bounding volume of the network.

After rasterizing all TGCs, we iterate through all elements of Λ in lexicographical order. If a value was written to $\Lambda(x,y,z)$, we push the associated volumetric value $\Phi(x,y,z)$ into the DT-Grid representing the segment identified by $\Lambda(x,y,z)$. Each value in Φ is either ignored or added into at most one DT-Grid representing a single segment in G.

At this point, we now have a graph G containing the original medial axis, connectivity, and radius information. Additionally, each segment in G has an associated DT-Grid containing the original scalar volume values in Φ that make up that segment. Even with the large overhead of the DT-Grid data structure, our experiments still show significant compression over storing the original scalar data Φ as a three-dimensional grid (Figure 69).

6.1.4. Visualization

The key to being able to visually convey complex data is to selectively render related structures, thereby limiting the amount of visual information a user sees at one time. We now describe methods for selectively visualizing the volumetric data associated with our net-

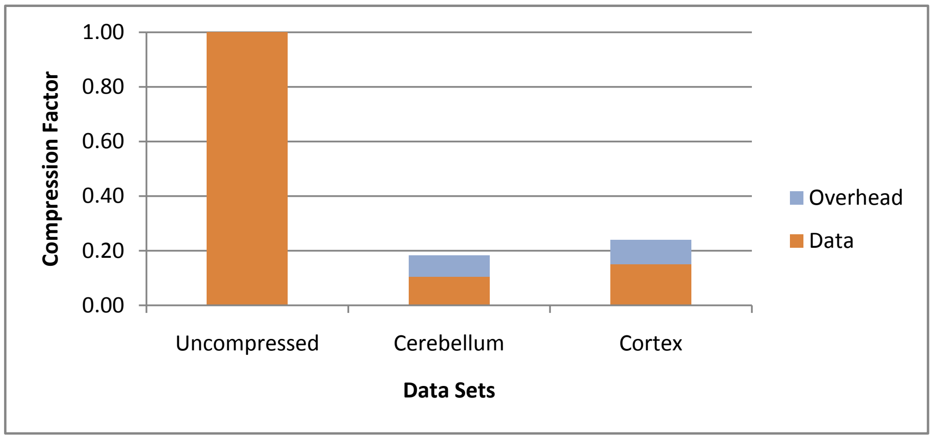


Fig. 58. Compression achieved after processing the volume function Φ into a series of DT–Grids associated with each segment. All volumetric data describing the network is stored while $\approx 90\%$ of the volume data representing surrounding tissue is culled. We compare the size of a full volume to sample volumes in two regions of the rat brain.

work structure. We also note that the overhead size is constant and independent of voxel size. As the voxel size increases (e.g. by storing colors or gradients), the compression provided by a DT-Grid relative to a uniform grid increases.

6.1.4.1. Previous Work

Several techniques exist for the visualization of filament and fiber data that can also be applied to vascular networks. Several surface construction methods discussed previously (Section 6.1.3.1) are designed for visualization. A large body of work is also available for estimating vascular surfaces [8, 21, 57] for visualization. However, these techniques do not capture surface features.

Oriented imposter methods such as stream lines are very good at representing highly organized fibers, such as those found in Diffusion Tensor Imaging [71, 82, 85]. These techniques have little applicability, however, in complex networks that have highly uncorrelated orientations as are found in microvasculature. Although these techniques can be adapted to selectively highlight filaments [60], unselected data still interferes with visualization. Additionally, these visualization techniques do not capture surface features, which can be important when exploring relationships between microvasculature and surrounding tissue.

6.1.4.2. Volume Visualization and Isosurfacing

Both ray-casting and isosurface rendering are important for volume visualization but do not offer many opportunities for selective visualization. Isosurfaces provide a mesh that can be rendered efficiently using hardware acceleration while direct volume rendering provides a way to visualize fine surface details that would be lost when selecting a single isovalue. The easiest way to selectively visualize structures in volume visualization is to cull away regions, focusing on local areas of interest. This is of little help in microscopy data sets since vascular structures can span the entire data set. Isosurfaces, in contrast, are often represented as triangles and therefore independent surfaces can be selectively visualized. This still causes problems with microvascular networks, however, since the structures are highly connected, requiring some means of separating the surface into useful components. In addition, selecting an isosurface value that fully captures the structure while limiting misclassifications can be impossible. We will now discuss how to adapt our algorithms to selectively visualize complex networks using standard volume visualization and isosurfacing techniques while eliminating most problems with noise.

The total volume of brain microvasculature is known to make up only a small percentage ($< 3\% - 6\%$) of the total volume in biological tissue. Since Φ is on a discrete grid, we must store all voxels that intersect the vascular surface. The estimated radius of our vector tracking algorithm (Section 6.1.2) and subsequent encoding based on a bounding volume (Section 6.1.3) therefore retains a slightly larger portion of the data set ($\approx 10\%$). This prevents important capillary features from being culled from the final image. Although noise in microscopy is highly correlated from section to section [54], it tends to occur randomly throughout a volume. Therefore, by culling over 90% of the volume, we have eliminated large quantities of volume noise that would interfere with isosurfacing and volume rendering methods. In the case of isosurfacing, we can now use more liberal isovalues without generating surface artifacts elsewhere in the data set.

Isosurfacing algorithms are straightforward to implement directly on a DT-Grid. For example, implementation of the Marching Cubes algorithm [48] requires that each element in the grid be visited, along with its neighbors. Although random access to any given neighbor requires logarithmic time (Section 6.1.3.2), the use of a 3x3 template across all elements allows the isosurface to be reconstructed in time linear to the number of values in the DT-Grid. This provides a net gain in efficiency since the DT-Grid only contains values near the network. Since geometry can be sent arbitrarily to the graphics card, the isosurface can be updated interactively based on the user's selection criteria.

The major problem encountered when rendering DT-Grids using volume rendering methods such as ray-casting is that random access into a DT-Grid requires logarithmic time. This is further complicated by our structure since G contains a separate DT-Grid for each segment. One way to combat this problem is to create a union of all selected segments and render them as a single DT-Grid. Creating the union of all segments requires time linear to the number of voxels in the union, which can be very efficient. However, the resulting structure is still a DT-Grid and requires logarithmic time for random access. We note that Nielson et al. [65]. have shown that, for large data sets, the logarithmic time random access is often faster than the constant time random access provided by a uniform grid. This is due to cache coherence. Unfortunately, there are no predefined algorithms to use DT-Grids with graphics hardware, so such methods would be limited to CPU-based rendering.

In order to implement hardware-accelerated volume rendering of our networks, we use a more direct approach. We store a uniform grid as a three-dimensional texture in graphics memory. When the user makes any updates to the selected filaments, the DT-Grids for each selected segment are copied to a 3D “canvas” in main memory. This canvas is then copied to the graphics card, updating the rendered texture. Copying the DT-Grids to the canvas is a highly efficient operation and can be performed interactively. In this case, updating the texture map in graphics memory is the bottleneck and causes some frame stuttering, depending on bus speed, when the user changes the selection criteria.

6.1.4.3. Results

These techniques allow us to perform structured visualization of complex network data. We are able to visualize the actual volumetric or isosurface data interactively based on user-selection. Examples of user selection operations include:

- Picking filaments from the network,
- Expanding a list of components using a breadth first search,
- Selecting filaments based on statistical queries of G (such as radius, branch angle, etc.),
- Visualization of the shortest path between two points in the network,
- and visualization of the diffusion of a substance based on simulated flow through the network.

Operations such as breadth first search allow us to understand the high level of connectivity that has been theorized to exist throughout the microvascular system. By selecting an initial sample of filaments based on these techniques, we use a breadth first search to explore network connectivity (Figure 59). By dynamically expanding and contracting the network, a user can view capillaries in the context of the entire network and then refine the visualization to include only regions of interest.

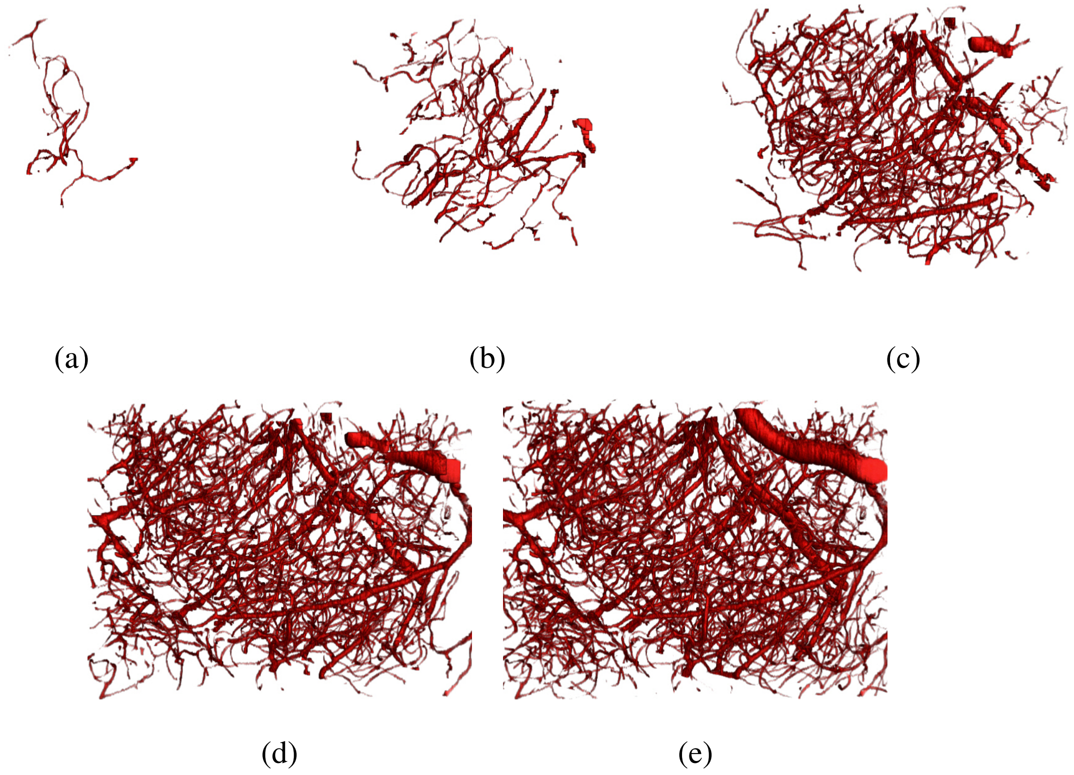


Fig. 59. Network searching and selection. A small bundle of filaments was picked by the user (a). Successive expansion of the network using a breadth-first search (b-d). The complete network (e) contains filaments at the edge of the data set that are unconnected to the main network.

6.1.5. Cellular-Vascular Relationships

One of the most interesting relationships between microvasculature and the surrounding tissue is between capillaries and nearby cells. The ability to explore these relationships would provide a valuable tool for scientists interested in the effects of microvasculature on cell nutrition and modulation.

6.1.5.1. Distance Metrics

Nutrients and other chemicals travel from capillaries to nearby cells using both diffusion and active transport (where mediating proteins carry molecules). Both of these transport methods are highly dependent on the distance of the cell to the capillary surface. The distance of the cell surface from the surface of nearby capillaries is therefore an effective metric for measuring the relationship between a cell and capillary.

It is straightforward to compute the distance from any point in the tissue to the central axis of any capillary, since this axis is extracted during segmentation. However, since capillaries vary in size, we should also take the radius of the capillary into account. We do this by using the distance to the computed bounding volume of the network (Section 6.1.3). In order to compute the distance from any specified point p_x interactively, we compute the distance between p_x and the TGC surrounding each edge in G. In order to determine the distance between a point p_x and a TGC, we use Equation 6.4 and Equation 6.5. We then compute the distance between p_n and the line formed by $p_{r,n}$ and $p_{r,n+1}$ (Figure 57).

6.1.5.2. Visualizing Distance Queries

We use this metric to visualize a network of capillaries closely associated with a selected point in the surrounding tissue as follows. By specifying a point in the volume Φ, we can select nearby components of G based on their distance from the specified point. Since often the effects of individual capillaries (not just portions thereof) are of interest to biologists, we can further adjust the selection to identify segments based on the distance of their nearest components in G (Figure 60). For measuring the distance from points to capillaries, we define the segment distance as the distance from the closest component edge (the minimum edge distance).

We show how this can be effectively used to cull excess microvascular information from a volume visualization (Figure 61). In this figure, the same region of tissue is visualized. The first image shows the entire region while the following images show only segments near the selected point. As the “radius of interest” is reduced, the structure of the surrounding capillaries becomes more clear.

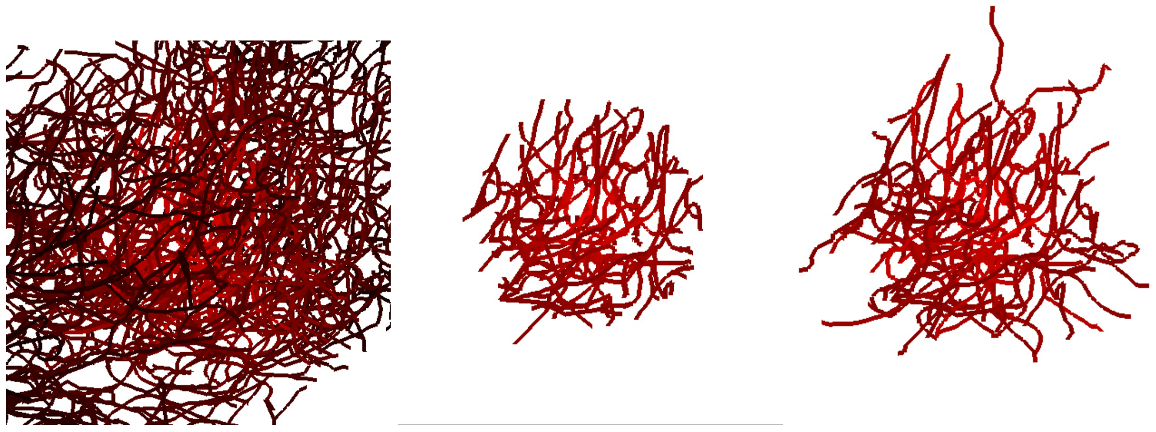


Fig. 60. Vertex and filament distance from a point. A region of cerebellar cortex is rendered using oriented billboards. Segments are colored based on the distance of the closest point to a cell (left). We then clip edges to a region $\approx 100\mu m$ from a cell. We can either select only edges within that distance (center) or any segment that contains an edge/vertex within that distance (right).

When selecting specific cells in order to determine their distance to neighboring capillaries, we must also take the radius of the cell into account. Since our cell segmentation is limited to a center point and radius, we simply subtract the cell radius r from the computed distance between the vascular surface and the center point p.

6.1.5.3. Voronoi Diagrams

Although the selection described above allows us to interactively explore the structure of vessels that support and modulate a single cell, many theories concerning the function of microvasculature concern the number and positions of cells surrounding microvessels. Of primary concern are cells that are most directly associated with a capillary. Using the techniques described in the previous section (Section 6.1.5.1) would involve computing the distance between every cell and blood vessel and sorting all values to find the closest relationships. Since both the number of cells and the number of edges in G are large, this process can be time consuming. Additionally, other aspects of the surrounding tissue may

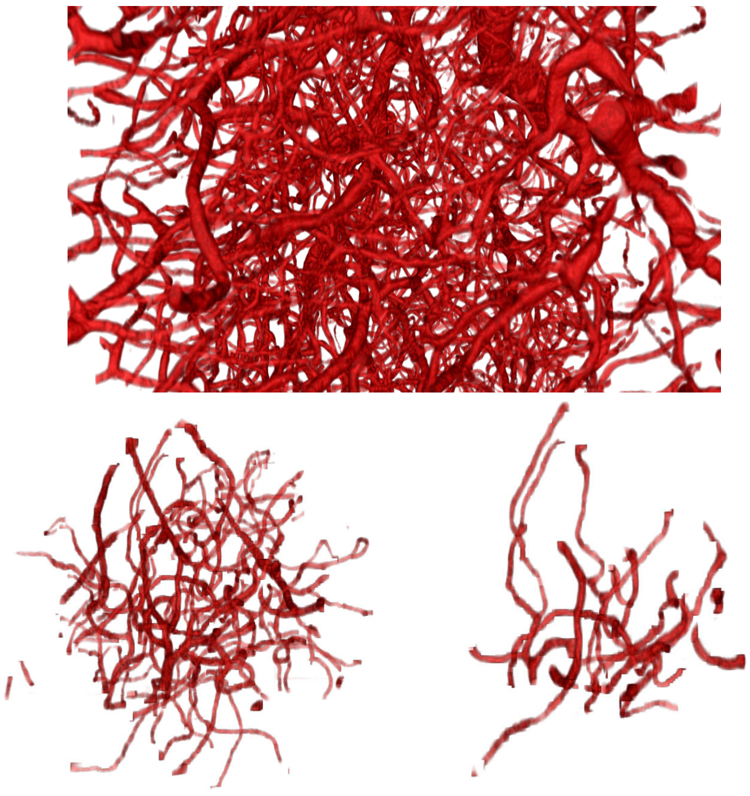


Fig. 61. Volume rendering of the region in Figure 60. The entire volumetric network is shown (top) as well as a cradle of vessels $\approx 100\mu m$ (a) and $\approx 50\mu m$ (b) from a specified point.

be of interest besides cell positions. For example, understanding the size and shape of a region closely related to a capillary may be interesting in studying tumor growth. This would require comparing each voxel in Φ to the bounding volume of G. The following method allows us to efficiently associate regions of tissue in Φ with the closest segments in G.

We label these regions by constructing an implicit Voronoi diagram based on G. Several methods are available for constructing Voronoi diagrams [5, 67] but most are designed to work with point sets rather than implicit or geometric surfaces. Fast methods have been proposed for computing implicit Voronoi diagrams from geometric primitives [32], however these are algorithmically complex, requiring $O(PN)$ where P is the number of primitives and N is the number of voxels in Φ.

We compute the Voronoi diagram by propogating an identifier for the Voronoi region while constructing a signed distance function (SDF). We use the standard definition of a SDF

$$\Psi = \begin{cases} d_\Psi(\bar{x}, \Gamma) & \text{if } \bar{x} \text{ is outside } \Gamma \text{ and} \\ -d_\Psi(\bar{x}, \Gamma) & \text{if } \bar{x} \text{ is inside } \Gamma. \end{cases} \tag{6.6}$$

where Γ is the vascular surface and $d_\Psi(\bar{x}, \Gamma)$ is the distance between $\bar{x}$ and Γ.

There are several well understood methods for constructing signed distance functions on a discrete grid. The most commonly used algorithm is the Fast Marching Method [83] which requires $O(n \log n)$ time to evaluate. The Fast Sweeping algorithm [94] converges in $O(n)$ time, although it requires that every point in the implicit function Ψ be visited multiple times. Since we wish to evaluate every point in Ψ, we use Fast Sweeping for efficiency.

Fast sweeping requires that we specify initial boundary conditions from which the final SDF is iteratively computed. We first initialize Ψ to some large value $+\infty$. We then rasterize each edge of G into Ψ using a three-dimensional line drawing algorithm [11]. At each sample point in Ψ intersected by an edge, we write

$$\Psi(\bar{x}) = -r_G(\bar{x}) \tag{6.7}$$

where r_G is the radius measured at the edge and $\bar{x}$ is the position in G along the edge. Both values are computed by interpolating between the graph nodes defining the edge. We then use Fast Sweeping to propagate the distance values through the rest of the function (Figure 62). Implementation details as well as the Ψ update function for uniform grids with cubic voxels are provided in the paper by Zhao [94]. Additional details describing the construction of SDFs can be found in Osher and Fedkiw [69].

It should be noted that the voxels of many actual biological scans (including ours) are not cubic. This means that a more complicated update function is required to correctly perform fast sweeping, or the data must be resampled to a uniformly spaced grid ($dx = dy = dz$). If a more complex update function is used, this involves solving the quadratic equation:

$$\left(\frac{\bar{x}_{x,y,z} - \bar{x}_{x-1,y,z}}{dx}\right)^2 + \left(\frac{\bar{x}_{x,y,z} - \bar{x}_{x,y-1,z}}{dy}\right)^2 + \left(\frac{\bar{x}_{x,y,z} - \bar{x}_{x,y,z-1}}{dz}\right)^2 = F \qquad (6.8)$$

for $\bar{x}_{x,y,z}$, which is the value in Ψ at point (x,y,z).

We construct the implicit Voronoi diagram by propagating a unique identifier for each filament along with the distance function. This is done during the Fast Sweeping procedure. In addition to the function Ψ, we maintain a separate implicit function Ψ_{VOR} that stores an identifier at each voxel indicating from which segment the distance value at that point originated. After completing all iterations of Fast Sweeping (eight for $\Psi(\bar{x})$ if $\bar{x} \in \mathbb{R}^3$), Ψ_{VOR} contains the complete Voronoi diagram where each point contains an identifier to the segment closest to that point.

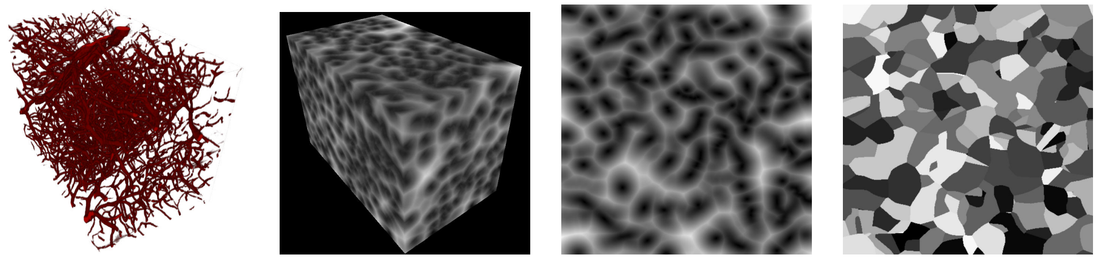


Fig. 62. Vascular network (a) and associated three-dimensional implicit signed distance function (b and c). The associated Voronoi diagram is constructed by propogating an identifier for each segment along with the distance value during computation of the Eikonal equation.

Finally, we use the computed Voronoi diagram to assign each cell to the appropriate capillaries (segments in G). Frequently, the cell will overlap multiple Voronoi regions. Depending on the desired visualization, the cell can be associated with one or many capillaries. In our visualizations, we elected to simply associate the cell with the region in which the cell's center was positioned. We can then render cell positions along with capillaries as either simplified spheres (representing the cell position and radius) or using volumetric data from the original data set (Figure 63). In the latter case, we stored the volumetric data in a DT-Grid associated with each cell as described in Section 6.1.3, using a sphere as a bounding volume.

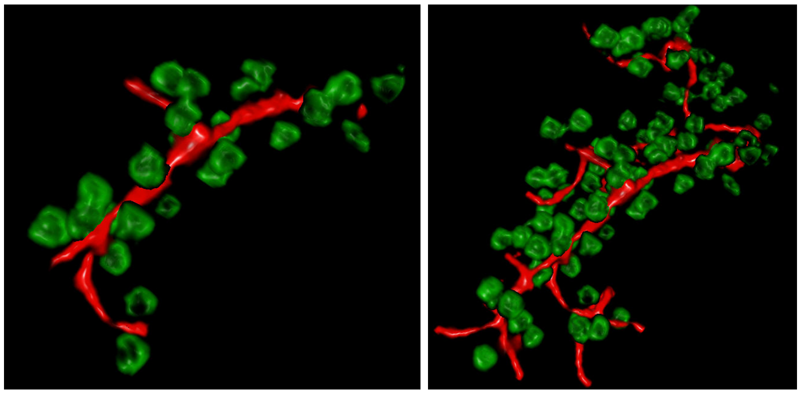


Fig. 63. A series of connected capillaries in the mouse somatosensory cortex along with their associated cell bodies.

6.1.6. Discussion

Although there are several methods available for visualizing vasculature on a macroscopic scale, microvasculature is a significantly more complex and interconnected structure. In order to interpret complex networks, biologists have been limited to statistical models describing features such as branch angles and vessel direction. Although these features are important for comparing biological models, they do little to elucidate local characteristics, such as the positions of cells around individual capillaries. We provide a tool that allows biologists to explore the connectivity and complexity of tissue microvasculature without sacrificing important high-resolution details, such as surface structure. This provides a tool for biologists to explore several potential areas, including:

- visualization of *angiogenesis*, or the formation of capillaries in the microvascular system
- visualization of the surface structure of capillaries, particularly around bifurcations
- analysis and visualization of microscopic aneurisms and other abnormalities visible in the microvascular surface
- detailed visualization of differences in microvascular structures between two types of tissue (control and chronic)

These features would be impossible to visualize using simplified primitives, such as cylinders or billboards. In addition, our selective visualization techniques allow a user to exclude excessive amounts of data and focus on local regions of interest.

We also associate the tissue surrounding the microvascular network with individual capillaries. This is of particular interest to biologists studying the relationships between capillaries and individual cells. This allows researchers to visualize several features that are not yet well understood, including:

- the number and types of cells directly associated with a single microvessel
- the regions of tissue that may be affected by aneurisms in either the macrovascular or microvascular system
- differences in the volume associated with a microvessel based on tissue type and region

We are unaware of any methods in the literature that allow biologists to explore the relationships between a network and its surrounding tissue. We have therefore provided a new valuable tool that allows researchers to visualize information that is not well understood.

6.1.7. Conclusion

In this chapter, we discuss a framework for visualizing the complex structure of a microvascular network. We then extend this framework to visualize the network's anatomical relationship to the surrounding tissue.

Due to the large size and high data rate of KESM, we were careful to limit our visualization algorithms to those with time complexity no greater than $O(N)$ where N identifies some large number of features. In our framework, this includes the number of cells, the number of nodes and edges in G, and the number of voxels in the data set Φ.

We provide the time required for several pre-processing and interactive stages in our framework (Table IV). We did not implement any automated methods for cell segmentation, since this is a well-understood problem and many methods are available in the literature [**?**, 19, 59]. In order to test our visualization methods we manually labeled cells in the data set. Segmentation of the microvascular network is performed automatically and approximately 97% of the network is labeled correctly while 3% of the capillaries were missed by the tracking algorithm.

All data sets were 512x512x512 voxels and rendered using a Geforce 7900 graphics board. We were able to maintain interactive frame rates while looking at any specified network. When the user changed the network, the appropriate DT Grids are copied onto a pre-allocated 3D canvas and sent to the GPU. This time required to perform this copy is highly dependent on the size of the network but was less than 0.2 seconds in most cases. This results in a small amount of stuttering when the network is changed, but the interactive frame rate returns to after the copy.

6.1.8. Future Work

Filament networks occur frequently in other high-throughput microscopy data sets. In particular, networks of neuronal fibers found in brain tissue can be visualized using techniques similar to those discussed in this chapter. Storing the volumetric data associated with neuronal fibers is particularly important since surface features play an important role in identifying their function (Figure 64).

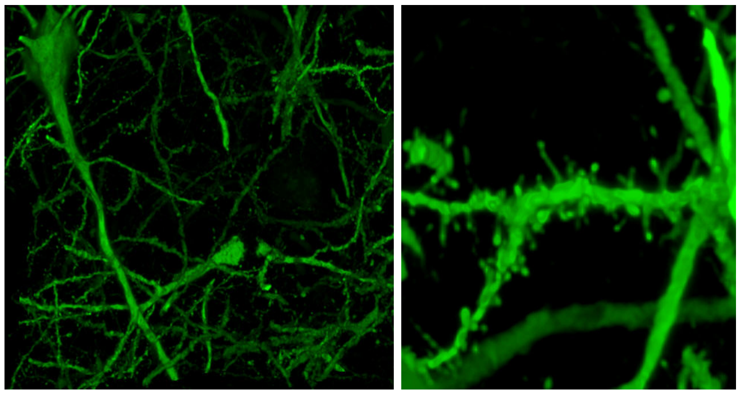


Fig. 64. Neurons imaged using Array Tomography [61] (left). Storing the network using DT-Grids preserves important surface details (right).

Finally, as data sets created using high-throughput microscopy become more frequently available, we expect to see many anatomical structures that are significantly more complex than those found at the macroscopic scale using traditional imaging methods. The density and high-frequency nature of these structures will require the use of underlying data structures similar to ours that can be used to query and selectively visualize anatomy.

Table IV. Time required for various stages of our framework using a 512x512x512 cerebellar data set (Figure 59) on an NVIDIA Geforce 7900. An updated filament volume is sent to the GPU only when the user changes the selection criteria.

Stage	Time
Preprocessing	
Vascular Segmentation	229.248 s
Cell Segmentation	Manual
Create DT-Grids	42.192 s
Interactive	
Update Filaments on the GPU	0.188 s
Render Time	0.047 s

6.2. Interactive Rendering

6.2.1. Introduction

The ability to directly render volumetric data provides unique insight about the structures embedded in these data sets. Although the extraction and rendering of isosurfaces [48] provides a way to look at specific values in a data set, fine details are often lost due to smoothing. Direct volume rendering [18] provides the best means of visualizing structures with fine-scale detail. The disadvantages of direct volume rendering are that the data sets are often stored as uniform three-dimensional grids which often require large amounts of memory, while rendering time is often wasted on empty space. Sparse data sets, such as vascular networks at the microscopic scale (Figure 65), require a very large embedding data set, even though the structures of interest may occupy $< 6\%$ of the actual volume.

In this chapter, we present a method for efficiently storing and rendering sparse volumetric structures. Using a data structure known as a Dynamic Tubular Grid (DT-Grid) [65], we achieve both storage and rendering time linearly proportional to the number of voxels that make up the embedded structures. DT-Grids provide two algorithmic properties important for volume rendering:

- Constant time ($O(1)$) access to each element when the elements are accessed in lexicographical order.
- Storage requirements linear ($O(n)$) to the number of values stored in the grid.

The primary result of this chapter is to demonstrate that DT-Grids provide a more efficient representation than both octrees and uniform grids for rendering sparse volumetric network structures. We first develop an iteration scheme that allows us to use DT-Grids effectively for volume visualization. We show that DT-Grids are particularly efficient for rendering sparse data structures, such as biological networks commonly found in biomedi-

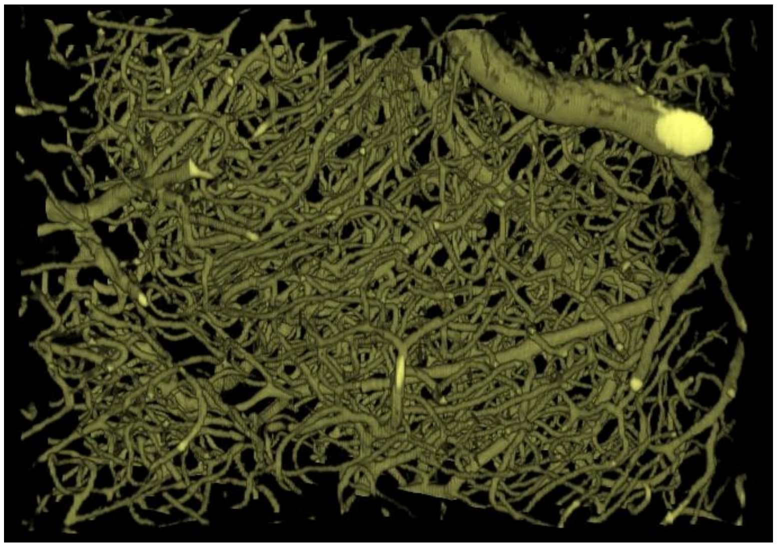


Fig. 65. Despite the density and complexity of the vascular isosurface, the volume occupied by the network is less than 6% of the total volume.

cal imaging. We then use splatting to render these structures in time linear to the volume of the embedded structure. Finally, we show that random access into a DT-Grid storing sparse volumetric data is significantly more efficient than octree methods, making DT-Grids an excellent candidate for both software and GPU-accelerated ray-casting.

6.2.2. Previous Work

A significant body of work has been published on volume rendering. Using three-dimensional grids for ray-casting [40, 97] has become popular since they provide constant-time access to any data element. When rendering sparse structures, however, grids are extremely inefficient and often require many times more storage space than the volume of the embedded structure.

Grids are also the most common method used for software and hardware-based ray-casting [20, 36, 78]. In addition to space inefficiencies introduced by grid-based data structures, ray-casting algorithms can waste large amounts of time looking into empty space surrounding volumetric structures. Some research has been published on improving render times for software-based techniques [**?**, **?**, 13, 44, 68] while more recent work has looked into using BSP trees [47] for GPU-based methods.

In order to overcome some of these space inefficiencies, structures such as octrees [37, 43] are used to aid in storage. While octrees provide efficient storage when high resolution is needed in a few local regions, they become highly inefficient when complex data is spread throughout a volume. Additionally, speed is often sacrificed for memory efficiency since octrees do not provide constant-time random access.

Another method of storing these data sets is as a series of points, which eliminates the need for storing intervening space. Each element of the data set has a position and a value (often a scalar). A large body of work has been published on rendering point-based data sets [4, 45, 72, 76]. Splatting algorithms [96] are often used to render point-based data sets and provide a way to render interesting volumetric structures while omitting empty space in the data set.

Although point-based data sets provide storage that is linear ($O(n)$) to the size of the structures in the volume, they have two major disadvantages. Template-based functions, such as gradient computations or convolution, are difficult to compute since no continuous function is defined. In addition, rendering high-quality point-based representations requires sorting so that the voxels are rendered in a back-to-front order to prevent popping and depth-related artifacts.

Our method using DT-Grids focuses primarily on efficient linear-time splatting, however we show that this data structure takes advantage of sparse data sets in a way that makes random access much more efficient than a standard space partitioning scheme. This is due to the coherence that exists within the structure embedded in the volume data set.

6.2.3. Dynamic Tubular Grids

In this section, we will provide implementation details for Dynamic Tubular Grids (DT-Grids). This structure was initially proposed by Nielson et al. [65] for solving level-set based surface evolution. While several other features and implementation details are described in the cited paper, we will limit our discussion to those that we have adapted and applied to volume visualization.

The DT-Grid data structure is defined recursively in that an N-dimensional DT-Grid is composed of an $(N-1)$-dimensional DT-Grid sub-component. We will therefore outline the one-dimensional structure and demonstrate some of its important properties. We will then describe how this technique is extended into higher dimensions.

6.2.3.1. One-Dimensional DT Grids

Assume that we are given a one-dimensional array of elements Λ. We call Λ *sparse* if elements of interest in Λ are interspersed with large regions of low-intensity values or zero-values that are of little interest. In regular grids, these regions serve an important function, providing a spatial context within which the data is embedded. A DT grid stores only values of interest embedded in the grid. We refer to a series of neighboring elements of interest in Λ as a *connected component*. These values are stored in sequential order (from least to greatest values of x) in an array we refer to as the `value` array. Instead of using empty space to position connected components inside the array, a second array is used, storing an integer triple defining each connected component's starting and ending coordinate as well as its position in the `value` array (Figure 66).

Although the `coord` array adds additional overhead to the structure, volumetric networks in biomedical data frequently take up a small percentage of the overall volume. The additional overhead of the DT-Grid is often eclipsed by the amount of space saved by eliminating excess empty space. This is particularly true when each voxel contains several components such as color values, density, and normals. Compression statistics are discussed in detail in Section 6.2.4.1.

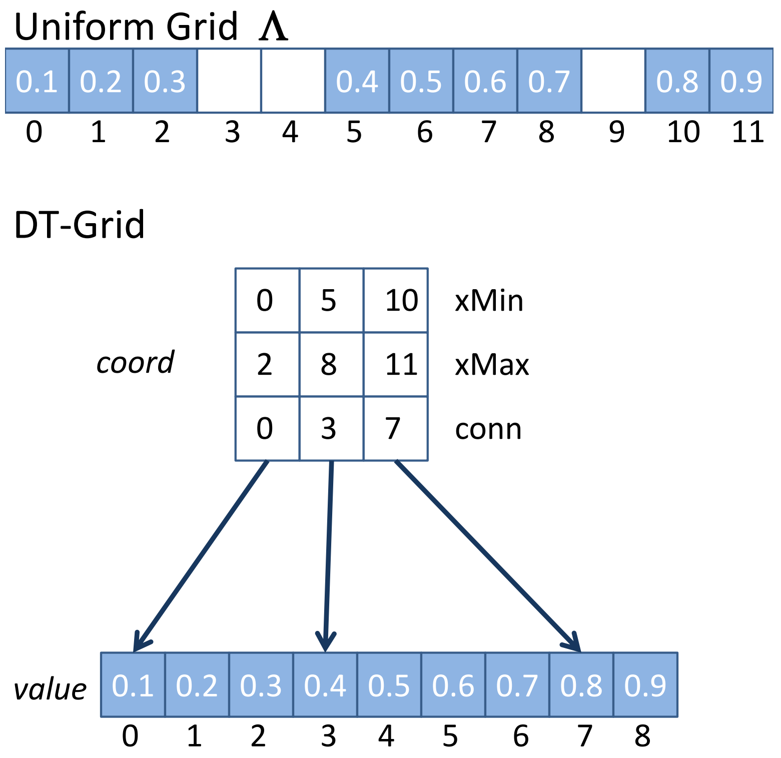


Fig. 66. Encoding a one-dimensional uniform grid Λ as a DT-Grid. Grid elements of interest (blue) are stored in a new `value` array, eliminating empty space (white). An additional three-element array `coord` is used as overhead in the DT-Grid to track the coordinates of the grid points. `xMin` and `xMax` record the original coordinates of the minimum and maximum values of each connected component. The variable `conn` provides a pointer to the first voxel of each connected component in `value`.

6.2.3.2. Algorithmic Properties

We will now discuss several algorithmic properties of this data structure that can be adapted to aid in volume visualization.

6.2.3.3. Construction

Adding an element into a DT-Grid is a constant-time operation, provided that the elements are inserted in the proper lexicographical order. The construction time of a DT-Grid is therefore linear in the number of elements inserted.

As stated previously, a DT grid is composed of two arrays. The `value` array is composed of scalar and/or vector data. In higher-dimensional DT-Grids, these values will be associated with each voxel (color, normals, opacity, etc.). The `coord` array is an array of integer triples that index into the `value` array (Figure 66).

When an element x_n is inserted into the DT-Grid, the value is placed next to the previous value (x_{n-1}) in the `value` array. If x_n was next to (x_{n-1}) in Λ, we simply increment the value of `xMax` associated with the current connected component. If the current value x_n was separated from $(n-1)$ in Λ by some empty space, we instead create a new connected component by adding an additional element to the `coord` array and initialize it with the starting coordinate of the new connected component.

A fundamental feature of note when creating this data structure is that all elements must be inserted in lexicographical order. That is, all elements must be inserted with increasing values of x, since values can only be inserted at the end of the `value` array.

6.2.3.4. Iteration

Iteration through a DT-Grid can also be done in constant time by keeping track of:

- the current x coordinate
- the current connected component (using an index into `coord`)
- the current position in the `value` array

Together, these components can form an *iterator* structure (Figure 67). As each value in the data structure is visited, the current x-coordinate is incremented until the end of a connected component is detected. The index to the current connected component is incremented and the x-coordinate is then set to the new `xMin`. The index into the `value` array is always incremented by one.

We note that iteration can be done in either lexicographical order (from the beginning of the data set) `or` reverse-lexicographical order (from the end of the data set). We build on this concept in order to perform correct object-order splatting (Section 6.2.4.2).

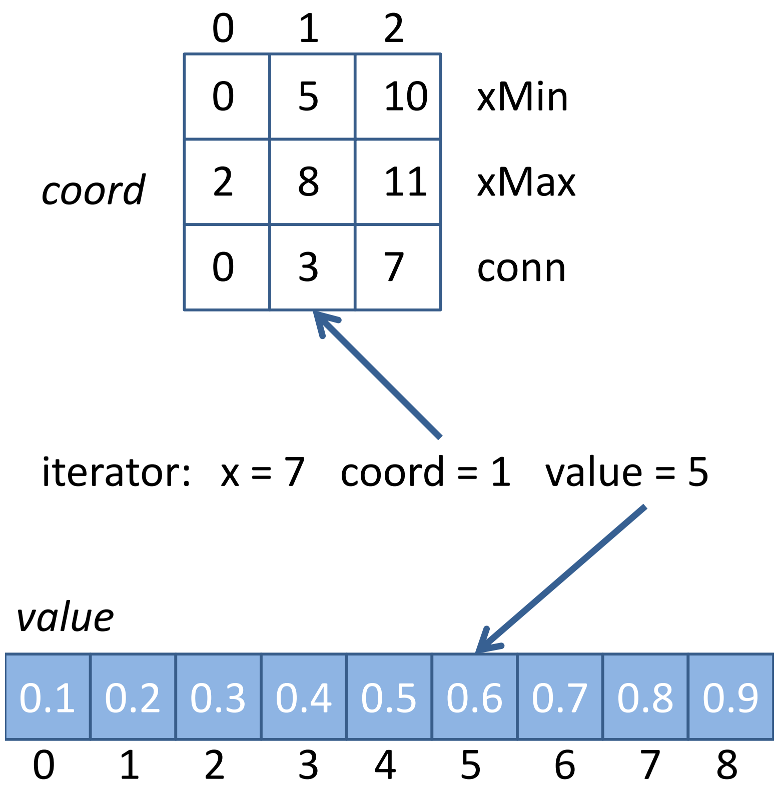


Fig. 67. Iteration through the one-dimensional DT-Grid defined in Figure 66. The iterator keeps track of the current x-coordinate as well as pointers to the `coord` array representing the current connected component and the `value` array representing the voxel at the current position.

6.2.3.5. Random Access

Given a specific x-coordinate, we would like to retrieve the associated value from the DT-Grid D. We do this by first performing a binary search through the `coord` array for the connected component n for which $xMin[n] \leq x < xMin[n+1]$. Once the appropriate column is identified, we then determine if the value is stored in the grid ($x \leq xMax[n]$). The complexity of the binary search is $O(\log N)$ where N is the number of connected components. After the correct connected component is identified, the appropriate value can be determined in constant time:

$$D(x) = value[coord[n] + x - xMin[n]] \tag{6.9}$$

6.2.3.6. Higher-Dimensional DT-Grids

As stated above, an N-dimensional DT-Grid contains an $(N-1)$-dimensional sub-grid. We now describe how to create a two-dimensional DT-Grid. Implementing the structure in higher dimensions follows the same pattern.

A two-dimensional DT-Grid contains the same components as the one-dimensional version with the addition of a `projection` component. The `projection` component is a one-dimensional sub-grid describing how the image is projected onto the x-axis (Figure 68a).

The two-dimensional DT-Grid keeps track of connected components along the y-axis while the one-dimensional component `projection` keeps track of connected components in the projection of the image onto the x-axis. The `projection` grid is used to keep track of the x-coordinate for the image. Instead of storing pixel/voxel values, each element in the `value` array of `projection` stores a two-component value:

- **to_coord** is an index into the two-dimensional `coord` array pointing to the first connected component in each x-axis column

- **to_value** is an index into the two-dimensional `value` array pointing to the first voxel in each *x*-axis column.

Every time an entire *x*-axis column has been inserted, a new component in `projection` is created for the next column. Insertion must still be performed in lexicographical order and all values are stored in the 2D DT-Grid's `value` array.

Iteration for *N*-dimensional DT-Grids is performed by maintaining iterators to the appropriate positions in each sub-grid. For example, a two-dimensional iterator maintains a pointer to the appropriate column in `projection` while each consecutive value along the *y*-axis is visited. When all *y*-axis values are visited in the current *x*-axis column, the index into `projection` is moved to the next column (thereby changing the current *x* coordinate).

6.2.4. Efficient Volume Visualization

In this section we will demonstrate the efficiency of DT-Grids for storing sparse volumetric structures. In addition, we will discuss modifications to the DT-Grid structure that allow us to perform efficient volume splatting.

6.2.4.1. Compression Characteristics

One of the most useful features of DT-Grids is the significant amount of compression provided by culling excess empty-space from a grid-based data set. This is particularly important for splatting algorithms since rendering time is directly proportional to the number of values in the data set. The overall compression also aids in allowing us to store larger data sets in memory. For comparison, we constrained our sample data sets to one byte per voxel. DT-Grids provided $\approx 10X$ compression for our data sets, although this is highly dependent on the structure of the data set (Figure 69). This becomes more pronounced as

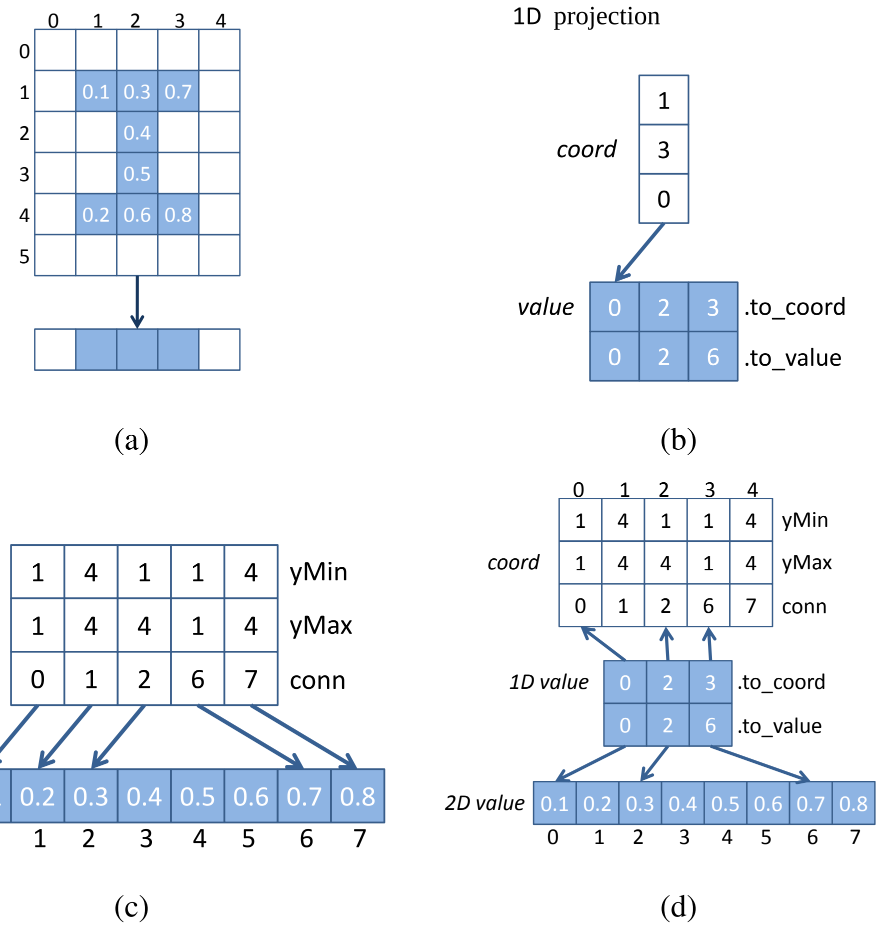


Fig. 68. Higher dimensional DT-Grids contain a lower dimensional component. (a) A two-dimensional structure is projected onto a one-dimensional array. (b) Instead of storing voxel data, the values for the 1D grid contain two indices into the 2D structure: `to_coord` and `to_value`. (c) The two-dimensional DT-Grid contains all volumetric data inside its `value` array. (d) The relationship between the 2D grid and the 1D projection. For any given point on the x-axis, the 1D grid contains a pointer to the first connected component along that axis and the first voxel value.

additional voxel data is added (colors, normals) since the overhead required for keeping track of the data structure remains constant. Nielson and Museth [65] have shown that DT-Grids provide greater compression than octrees by taking advantage of connected data. Our experiments show even greater improvement for space-spanning data sets such as biological networks (Figure 70). Although the data sets are sparse, the volumetric data tends to be spread throughout the entire data set. The average depth of octrees and BSP trees tends to increase for space-filling data in order to resolve each voxel.

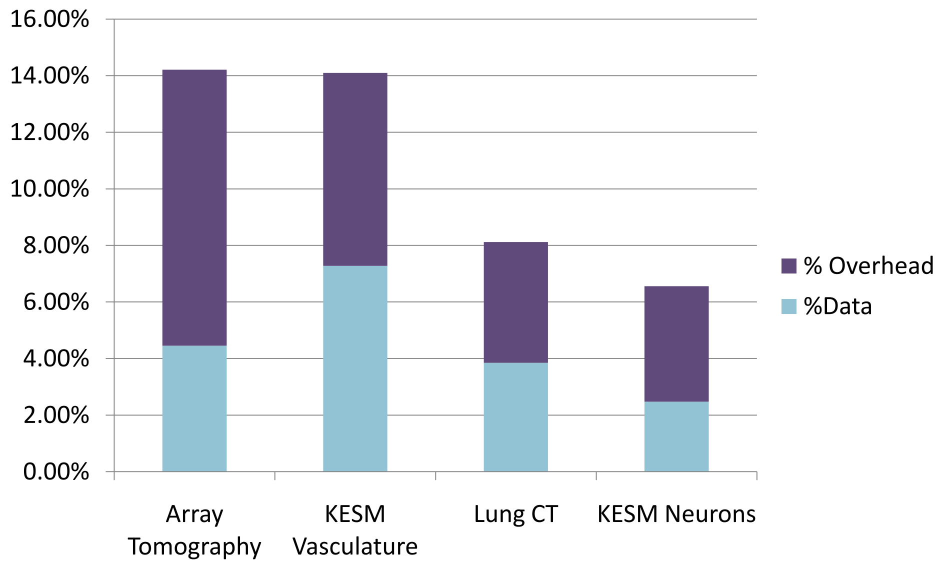


Fig. 69. Compression ratio computed from several different sparse data sets. The voxel data is stored as a single unsigned character value (one byte) for each data set to provide comparison. In general, thicker structures tend to provide a better data/overhead ratio while thin structures take less advantage of coherence between voxels.

6.2.4.2. Volume Splatting

In this section, we discuss how DT-Grids can be used to perform linear-time volume splatting in the optimal back-to-front order for orthographic and perspective projections. This is done by selectively choosing the iteration direction for each axis. We can then guarantee that individual voxels are visited in a back-to-front order.

In addition, it is important to note that a major constraint on iteration through a DT-Grid is that the nodes must be visited in x-y-z order. For example, all z_n points associated with a given value of y must be visited before moving on to the next value of y. In fact, the standard definition of a DT-Grid requires that all points be visited in lexicographic order.

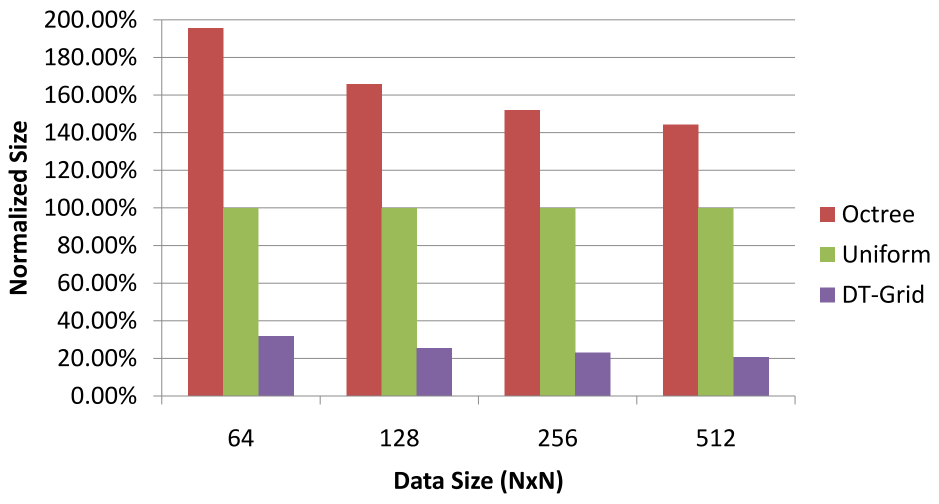


Fig. 70. The size of an octree, uniform 3D grid, and DT-Grid at increasing resolutions (normalized to the size of the uniform grid). Since the complexity of the volumetric structures is spread evenly throughout the volume, using an octree to store the data actually adds additional overhead to a uniform grid.

By allowing iteration in reverse, we can loosen this constraint enough to allow back-to-front splatting from any viewpoing.

6.2.4.3. Independent-Axis Iteration

In order to loosen the current constraints on iteration, we note that we can control the *direction* of iteration along each axis. For example, we can iterate through the 1D sub-grid from x_{min} to x_{max} while incrementing the pointer in the two-dimensional grid from y_{max} to y_{min}. This allows us to break the constraint on iteration in lexicographical order for the entire data set, provided that we iterate continuously along each axis. Being able to select the iteration direction for each axis is key to performing splatting in the proper order for both orthographic and perspective projections.

We wish to choose an ordering for each axis that will guarantee that any visited voxel will lie on top of all previously visited voxels. Consider the two-dimensional case of an orthographic projection of a square onto a plane P (Figure 71). We prove that this can be done recursively by showing that, no matter the position or orientation of P, we can pick an ordering of x such that any line drawn from (x_{tx}, y_{min}) to (x_{tx}, y_{max}) will lie on top of the previous line (x_{tx-1}, y_{min}) to (x_{tx-1}, y_{max}), where tx is the time at which the line is rendered.

Now consider any line from (x_{tx}, y_{min}) to (x_{tx}, y_{max}) with point p_{ty} =(x_{tx}, y_{ty}), where y_{ty} is the y-position of the point when it is visited at time ty (where $ty \geq tx$). We can also choose an iteration order for the y-axis in which p_{ty} will always be projected on top of p_{ty-1} (Figure 71b).

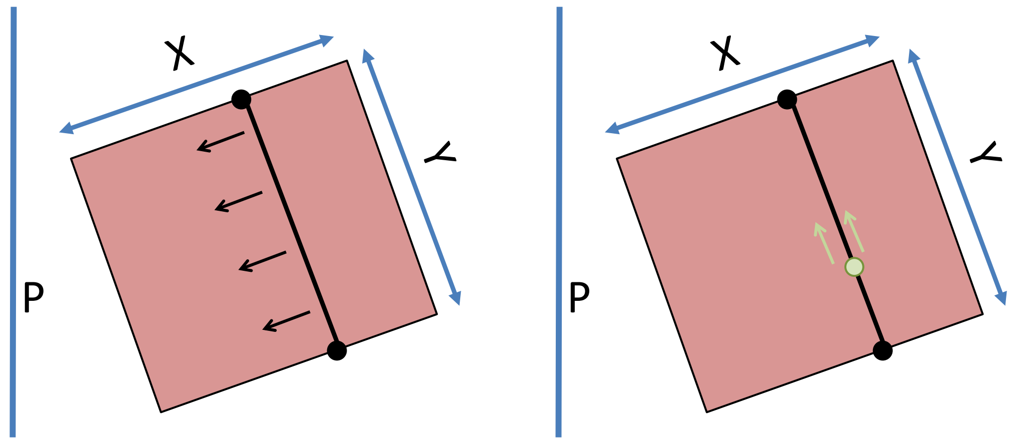


Fig. 71. By selecting the iteration direction for each axis, each voxel in the data set can be visited in the proper order to ensure a back-to-front orthographic projection onto a 2D plane P. (a) Selecting the iteration direction for x and (b) for y depends on the Cartesian quadrant in which the normal for p resides.

6.2.4.4. Perspective Projection

Note that this proof assumes that we are projecting the data set onto P using an orthographic projection. This method works for a vast majority of viewpoints in a perspective projection as well, however there will be some visible popping when looking down the y and z axes (Figure 72). This popping occurs when a front face of the data set occludes all other faces.

The extent of this popping is therefore dependent on the viewing angle and the distance between the view point and the closest point in the data set.

Removing these rendering artifacts requires a slightly more complex iteration scheme. One way of thinking of our current iteration constraints (using axis-independent iteration) is that we must always start at one corner of the data set and iterate inwards. At each point in the iteration process, we can retrieve the current (x, y, z) position in constant time, since it is updated every iteration. We can therefore stop iteration at any time and continue on to the next column by incrementing the `projection`-iterator (Figure 73-left).

We can therefore split the data set into parts and iterate across each part separately, provided that it is bounded by edges (and at least one corner) of the original data set. This requires very little additional overhead since each voxel is still only visited once. We can therefore eliminate artifacts in the perspective projection by breaking the volume into four regions. We then iterate through each region independently and in the appropriate order. A side view of two of these regions are shown in Figure 73.

6.2.4.5. Fast Random Access

The degree of compression realized by using DT-Grids for sparse volume data is particularly interesting for hardware accelerated rendering techniques because limited memory and texture constraints on graphics cards set a practical limit to the size of data sets that can be rendered. Sparse grid representations, such as DT-Grids and octrees, pose a particular problem for hardware accelerated rendering because ray-casting and texture mapping algorithms require random access. This has been addressed through the use of GPU-based octree implementations [43], although these methods are primarily used for storing textures for mapping onto a 2D surface. For space-filling structures such as volumetric networks, octree branches tend to approach their maximum length in order to address all of the important voxels in the data set. In this section, we will compare the average number of iterations

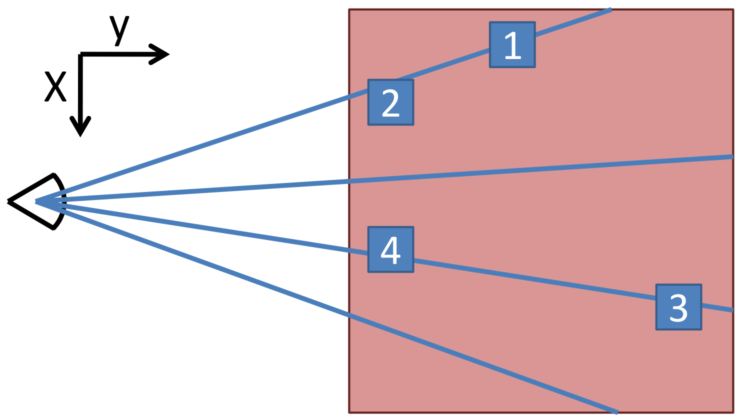


Fig. 72. At certain viewing angles in a perspective projection, it is impossible to select an axis direction that will satisfy the constraints of front-to-back ordering. In this image, the iteration methods described in Section 6.2.4.3 would render a plane sweeping along the *x*-axis. In order to satisfy back-to-front rendering constraints, region 1 must be visited before region 2 and region 3 must be visited before region 4. There is no single combination of iteration directions that would meet these constraints.

required to evaluate a random access query into a DT-Grid and an octree for our sample data sets.

As discussed in Section 6.2.3.5, random access is implemented by performing a binary search through a list of connected components stored in the `coord` array. Random access into higher-dimensional grids involves a single binary search for each dimension. Random access into a two-dimensional DT-Grid, for example, is performed by first locating the relative *x* column in the one-dimensional `projection` grid using a search through its `coord` array. If the *x*-coordinate is found, another binary search is performed in the `coord` array of the two-dimensional grid. This second search is bounded by the result of the 1D search, so only a subset of the 2D `coord` array is used.

Although the computational overhead is higher, the unique structure of sparse volumetric data allows random access operations to be evaluated in significantly fewer iterations

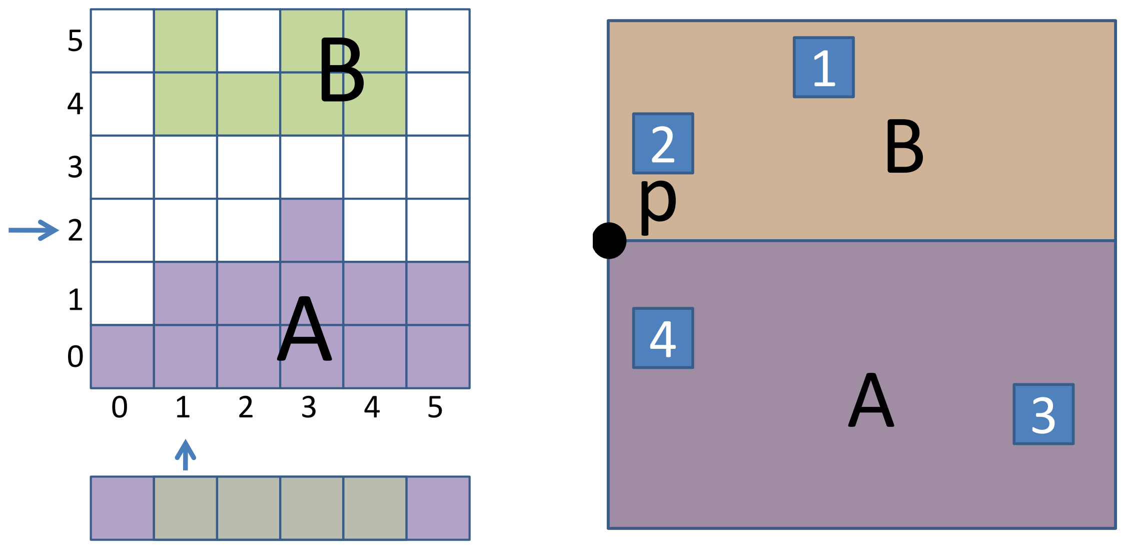


Fig. 73. (left) A two-dimensional example of iterating until a specified coordinate. By moving along the y axis until $y \geq 3$ and then incrementing the 1D iterator, we can visit all nodes in region A while excluding region B. (right) Using this technique, we can eliminate artifacts in the perspective projection by iterating until point p (see Figure 72) to visit all of region A, and in the opposite direction, visiting all voxels in region B.

than an octree representing the same volume. This is due to the fact that large volumes of evenly packed sparse structures tend to fill the corresponding 1D and 2D projections. As the size of the data set increases, the number of separate connected components in the 1D and 2D sub-grids decreases. Therefore, as the size of the data set increases, the average time to access the appropriate x and y components of the three-dimensional DT-Grid approaches constant time. The majority of the search operation is spent locating the correct connected component along the z-axis. Since the data set is sparse in three dimensions, the number of connected components is often very small, even for large data sets. This is illustrated in Figure 75 for the 2D case.

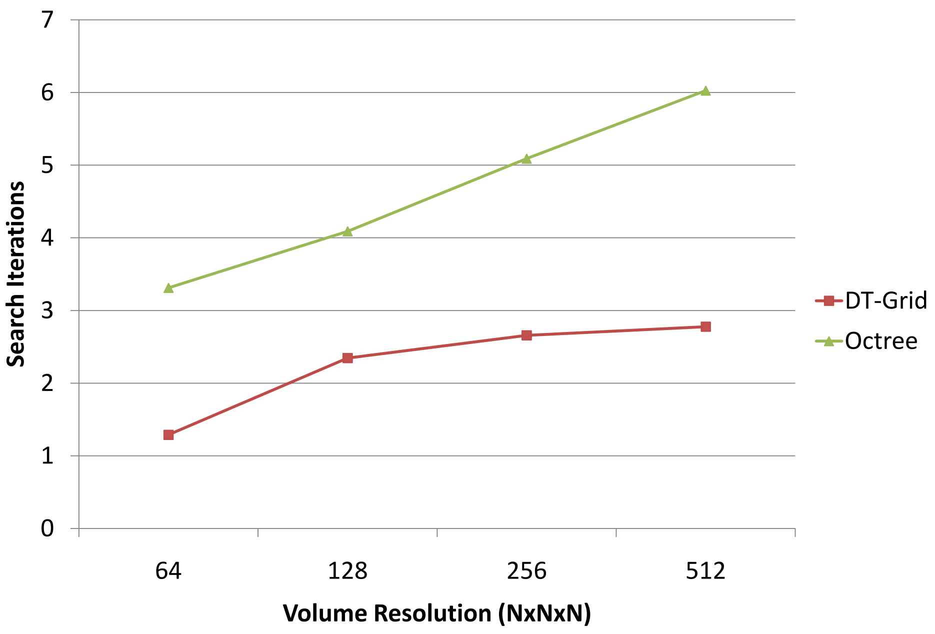


Fig. 74. Average number of search iterations required for random access into a DT-Grid and an octree. Random access was tested using the KESM Vascular data set. The average was computed from the number of search iterations required to find every voxel in the data set. This is similar to what would be required for ray-casting or texture mapping.

6.2.5. Results

In this chapter, we have demonstrated that DT-Grids are a highly efficient structure for storing and visualizing sparse volumetric data. These types of biological structures are becoming more prevalent in volumetric data sets as medical imaging technology improves. The compression provided by this structure is very useful since biological networks tend to span large regions that can be difficult to manage using uniform grids. The ability to perform volume splatting in linear time dependent on the volume of the structure, rather than the embedding volume is also an important advantage. Although this can be done using either octrees or point sets, they are limited to rendering techniques, such as a maximum intensity projection, that are independent of rendering order. As seen in our sample

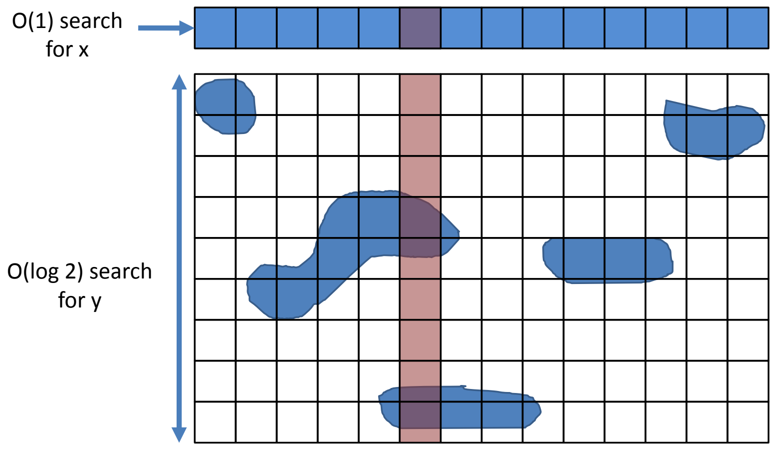


Fig. 75. The DT-Grid structure can provide highly-efficient random access for sparse data sets when the data is spread evenly throughout the volume. This two-dimensional example shows how the *x*-component is resolved in constant time, limiting the search to only two connected components along the *y*-axis.

data sets (Figure 76), shading provides important visual queues that allow a user to better understand the structure of these complex data sets.

6.2.6. Discussion

Fast random access is an important feature, particularly for software-based ray-casting where octrees are the standard data structure for representing sparse data sets. Although octrees are effective at representing data sets with regions of high local detail (such as small local regions and surfaces), they are significantly less efficient when the overall complexity of the volume is high. DT-Grids are able to take advantage of this overall complexity in both speed and efficient memory usage.

There are several important properties of DT-Grids that we did not explore in this chapter. In particular, iterators can be used to provide constant-time access to neighboring values in a template. This is important for computing gradients for shading, and can be used for blurring and resampling of the data set in linear (O(n)) time.

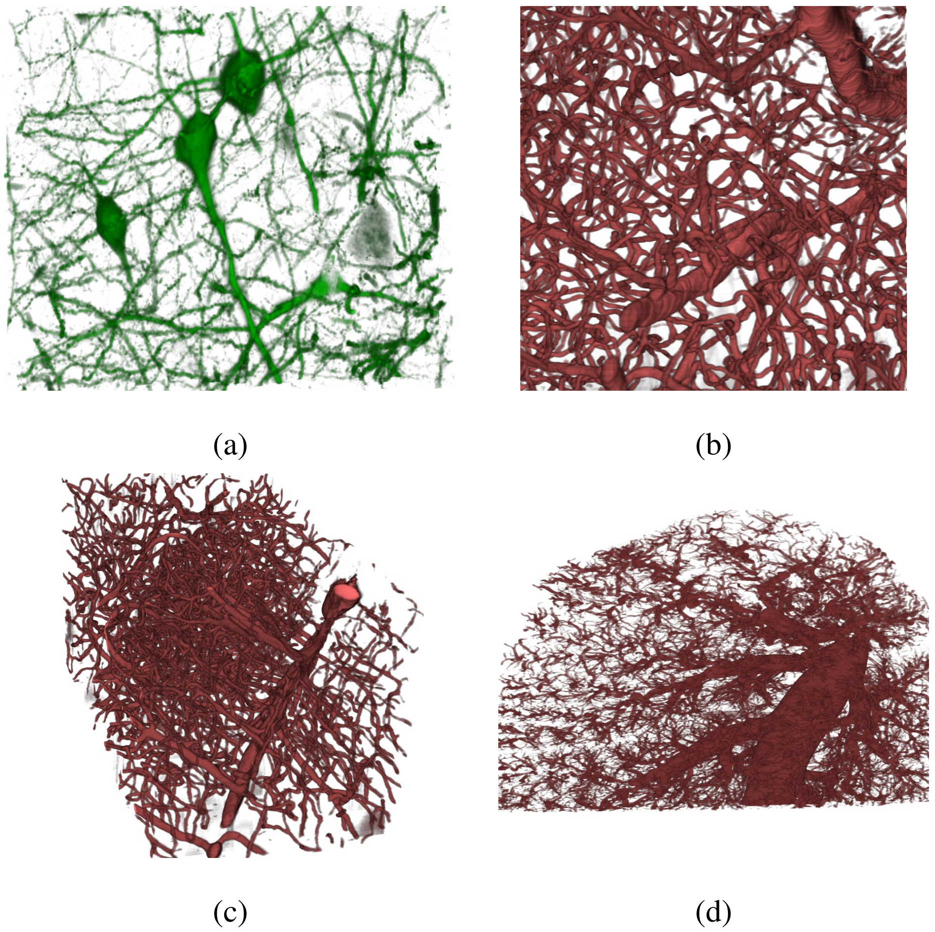


Fig. 76. Volume visualizations of several sample data sets used in this chapter. (a) Neural networks imaged using Array-Tomography [61]. microvascular networks imaged using KESM [52] in both an (a) orthographic and (b) perspective view. (c) Lung vasculature imaged using Computed Tomography.

Finally, there are several important limitations to DT-Grids as a rendering structure. Although accessing neighboring values using iterators is straightforward, accessing a neighbor of a random voxel requires an additional random access. Therefore, linear interpolation for ray-casting and texture-mapping is more expensive than with uniform grids. DT-Grids are also difficult to manipulate once constructed, making them cumbersome for representing time-varying data sets.

CHAPTER VII

SUMMARY AND FUTURE WORK

In this dissertation I have described the development of the prototype Knife-Edge Scanning Microscope (KESM). KESM allows tissue samples in excess of $1cm^3$ to be imaged at a resolution comparable to standard optical microscopes. In addition, KESM provides a data rate that is far faster than current imaging techniques and is capable of creating high-resolution large-scale data sets consisting of over three terabytes of data in a matter of days. This introduces the concept of *high-throughput microscopy*, where high-resolution data sets can be created from large tissue volumes far faster than any standard methods currently available. One of the major problems when dealing with this data is that the amount of data from high-throughput imaging greatly exceeds our ability to analyze it.

I then introduce methods to aid in the analysis of KESM data. These include fast and highly parallel image processing algorithms that allow artifacts to be removed in parallel with KESM imaging. I discuss algorithms that I have developed for performing fast segmentation of filament networks, which are a common feature in microscopy data sets. These tracking algorithms take advantage of modern graphics hardware in order to quickly track fibers in volumetric data sets, creating connected three-dimensional graphs representing filament networks.

Based on these graphs, I develop a framework for constructing high-resolution microvascular models. I show how these models can be used to perform measurement and analysis of the three-dimensional structure of microvascular networks. Although microvascular morphometry has been performed using standard techniques such as confocal microscopy, KESM is the first imaging method that can extract the three-dimensional structure of large-scale microvascular networks.

Finally, I discuss methods for visualizing complex filament data. Standard volume visualization methods using a uniform grid do not easily convey the underlying structure of filament data. The methods I describe use selective visualization, allowing the user to specify the data that is shown using graph queries. The user can then exclude the surrounding structures, visualizing only data of interest. This method of selective visualization is then extended to provide ways for the user to understand the relationships between these complex filament networks and the surrounding tissue.

The techniques described in this dissertation form a broad base of tools with which we can begin to explore tissue microstructure in three dimensions. The imaging techniques that I describe allow us to scan three-dimensional tissue volumes far larger than those possible using the state of the art in optical microscopy. The analysis and visualization algorithms that I provide are also faster than standard methods and allow us to more easily explore complex structures unique to microscopy data sets.

7.1. Future Work

In this section I will discuss future directions for KESM and high-throughput microscopy in general. In addition, I will discuss how the imaging and analysis tools presented in this dissertation could be used to solve several problems posed in my introduction (Chapter I).

7.1.1. Knife-Edge Scanning Microscopy

The most obvious extension to KESM is to modify the technology to allow fluorescence microscopy. Fluorescent stains have become the *de facto* standard in biomedical imaging of cellular and sub-cellular data, mostly due to their enhanced selectivity. In addition, fluorescence staining can be performed using transgenic animals, which allows complete staining of a tissue sample without resorting to the *en bloc* staining techniques that we are currently using (Chapter II).

7.1.2. High-Throughput Microscopy

Several recent methods are capable of performing high-throughput microscopy using different imaging modalities. While KESM is the only method capable of imaging large tissue volumes, techniques such as Array Tomography [61], Serial Block Face Scanning Electron Microscopy (SBF-SEM) [14], and the Automated Tape-Collecting Lathe Ultramicrotome (ATLUM) [**?**] are capable of higher resolution and selectivity for much smaller volumes. Ideally, these methods could be combined with KESM to provide a comprehensive understanding of tissue samples at many scales. The most difficult problem to overcome seems to be the destructive nature of all of these imaging modalities. This limits us to a statistical comparison across multiple specimens rather than a multi-scale analysis of a single specimen.

7.1.3. Analysis

Finally, the computational methods described have several future applications in biological research. In particular, I mention three computational problems in Chapter I:

- Creating databases of tissue features
- Constructing high-resolution three-dimensional models
- Understanding connectivity in the brain

Large-scale databases of tissue features have immediate research applications for comparing animal models to accepted controls. These controls can be imaged using KESM and reconstructed with the computational methods described in this dissertation. This would allow researchers to compare smaller samples from their models, obtained using standard imaging techniques, with a freely available database of tissue imaged with KESM.

Anatomical information obtained using KESM can also serve as input for complex physically-based models for understanding tissue function. One particular application is the simulation of blood flow through microvascular networks. We have used KESM to capture complete and fully-resolved microvascular information in high contrast using India-ink staining. By using the computational methods discussed in Chapter V, physically-based simulation of blood flow may be possible.

Finally, understanding the structure and function of the brain is a long-term goal, and has been acknowledged as a Grand Challenge Problem [79]. Since neuron connectivity is an important part of brain function, KESM can perform an important function in this research by imaging far-reaching connectivity within the brain. However, completely understanding brain function would also require significantly more information about local connections and physiology. This would require additional high-resolution imaging methods, like those mentioned above, as well as a complete understanding of brain physiology over time.

7.2. Conclusion

The KESM is the first device to allow high-resolution microscope imaging of tissue at this large scale. As such, KESM opens the door to a far more comprehensive three-dimensional study of anatomy. The particular focus of my work has been on brain tissue. This is due both to its complexity, as compared to other organs, and our relative lack of information about its three-dimensional structure. Although the cellular structure of the brain is extremely complex and detailed at the microscopic level, connections can span large regions. This requires large volumes of tissue to be studied at the microscopic level in order to gain a complete context of the structure and function of these networks.

In addition to advances in imaging methods, segmentation and visualization algorithms must also be improved in order to deal with the large size of high-throughput data sets. In particular, this requires making use of simple processing algorithms on highly parallel systems, such as GPU architectures and cluster computers. Finally, new algorithms are necessary for exploring structures, such as filament networks, that are far more common in microscopy data than in standard biomedical data sets. The algorithms that I have presented in this dissertation provide an important basis for the exploration and modeling of these unique and important structures.

REFERENCES

[1] L. C. Abbott and C. Sotelo. Ultrastructural analysis of catecholaminergic innervation in weaver and normal mouse cerebellar cortices. *The Journal of Comparative Neurology*, 426:316–329, 2000.

[2] D. A. Agard. Optical sectioning microscopy: Cellular architecture in three dimensions. *Annual Reviews in Biophysics and Bioengineering*, 13:191–219, 1984.

[3] K. Al-Kofahi, S. Lasek, D. Szarowski, C. Pace, G. Nagy, J. Turner, and B. Roysam. Rapid automated Three-Dimensional tracing of neurons from confocal image stacks. *IEEE Transactions on Information Technology in Biomedicine*, 6:171–186, 2002.

[4] M. Alexa, J. Behr, D. Cohen-Or, S. Fleishman, D. Levin, and C. T. Silva. Point set surfaces. *Proceedings of the conference on Visualization'01*, pages 21–28, 2001.

[5] F. Aurenhammer. Voronoi diagrams: A survey of a fundamental geometric data structure. *ACM Computing Surveys (CSUR)*, 23(3):345–405, 1991.

[6] P. J. Basser, S. Pajevic, C. Pierpaoli, J. Duda, and A. Aldroubi. In vivo fiber tractography using DT-MRI data. *Magnetic Resonance in Medicine*, 44(4):625–632, 2000.

[7] G. D. Battista, P. Eades, R. Tamassia, and I. G. Tollis. *Graph Drawing: Algorithms for the Visualization of Graphs*. Prentice Hall, Upper Saddle River, NJ, 1998.

[8] J. Bloomenthal and K. Shoemake. Convolution surfaces. *Proceedings of the 18th Annual Conference on Computer Graphics and Interactive Techniques*, pages 251–256, 1991.

[9] J. A. Bondy and U. S. R. Murty. *Graph Theory with Applications*. Elsevier Science Ltd., 1976.

[10] M. Born and E. Wolf. *Principles of Optics: Electromagnetic Theory of Propagation, Interference and Di?raction of Light*. Cambridge University Press, London, 7th edition, 1999.

[11] J. Bresenham. Algorithm for computer control of a digital plotter. *IBM Systems Journal*, 4(1):25–30, 1965.

[12] A. Can, H. Shen, J. N. Turner, H. L. Tanenbaum, and B. Roysam. Rapid automated tracing and feature extraction from retinal fundus images using direct exploratory algorithms. *IEEE Transactions on Information Technology in Biomedicine*, 3:125–138, 1999.

[13] D. Cohen and Z. Sheffer. Proximity cloudsan acceleration technique for 3D grid traversal. *The Visual Computer*, 11(1):27–38, 1994.

[14] W. Denk and H. Horstmann. Serial Block-Face scanning electron microscopy to reconstruct Three-Dimensional tissue nanostructure. *PLoS Biology*, 2:e329, 2004.

[15] W. Denk, J. H. Strickler, and W. W. Webb. Two-photon laser scanning fluorescence microscopy. *Science*, 248:73–76, 1990.

[16] A. Dima, M. Scholz, and K. Obermayer. Automatic segmentation and skeletonization of neurons from confocal microscopy images based on the 3-[D] wavelet transform. *IEEE Transactions on Image Processing*, 11(7):790–801, 2002.

[17] P. Doddapaneni. *Segmentation Strategies for Polymerized Volume Data Sets*. PhD thesis, Department of Computer Science, Texas A&M University, 2004.

[18] R. A. Drebin, L. Carpenter, and P. Hanrahan. Volume rendering. In *Proceedings of the 15th Annual Conference on Computer Graphics and Interactive Techniques*, volume 22, pages 65–74, 1988.

[19] A. D'Souza. *Automated Counting of Cell Bodies Using Nissl Stained Cross-Sectional Images*. Master's thesis, Texas A&M University, 2007.

[20] K. Engel, M. Kraus, and T. Ertl. High-quality pre-integrated volume rendering using hardware-accelerated pixel shading. *Proceedings of the ACM SIGGRAPH/EUROGRAPHICS Workshop on Graphics Hardware*, pages 9–16, 2001.

[21] G. Gerig, T. Roller, G. Szekely, C. Brechbuhler, and O. Kubler. *Symbolic description of 3-D structures applied to cerebral vessel tree obtained from MR angiography volume data*, volume 687 of *Lecture Notes in Computer Science*, pages 94–111. Springer Berlin, 1993.

[22] C. Golgi. Sulla struttura della sostanza grigia del cervello. *Gazzetta Medica Italiana*, 33:244–246, 1873.

[23] C. Golgi. Sulla ?na anatomia del cervelletto umano. *Rendiconti del R. Istituto Lombardo di Scienze e Lettere*, 7:69–72, 1874.

[24] C. Golgi. Sulla ?na struttura dei bulbi olfattori. *Rivista Sperimentale di Freniatria e Medicina Legale*, 1:403–425, 1875.

[25] R. C. Gonzalez and R. E. Woods. *Digital Image Processing*. Prentice Hall, 2nd edition, 2002.

[26] H. K. Hahn, B. Preim, D. Selle, and H. O. Peitgen. Visualization and interaction techniques for the exploration of vascular structures. *Proceedings of the Conference on Visualization'01*, pages 395–402, 2001.

[27] M. Harris. *GPU Gems 2: Mapping Computational Concepts to GPUs*. Addison Wesley, Mar, 2005.

[28] B. T. Hawkins and T. P. Davis. The Blood-Brain Barrier/Neurovascular unit in health and disease. *Pharmacological Reviews*, 57:173, 2005.

[29] X. He, E. Kischell, M. Rioult, and T. J. Holmes. Three-Dimensional thinning algorithm that peels the outmost layer with application to neuron tracing. *Journal of Computer-Assisted Microscopy*, 10(3):123–135, 1998.

[30] S. Heinzer, T. Krucker, M. Stampanoni, R. Abela, E. P. Meyer, A. Schuler, P. Schneider, and R. Muller. Hierarchical microimaging for multiscale analysis of large vascular networks. *NeuroImage*, 32(2):626–636, Aug. 2006.

[31] B. Herman and K. Jacobson. *Optical Microscopy for Biology*. Wiley-Liss, New York, 1990.

[32] K. E. Hoff III, T. Culver, J. Keyser, M. Lin, and D. Manocha. Fast computation of generalized voronoi diagrams using graphics hardware. pages 277–286, 1999.

[33] T. Ju, J. Warren, G. Eichele, C. Thaller, W. Chiu, and J. Carson. A geometric database for gene expression data. In *Proceedings of the 2003 Eurographics/ACM SIGGRAPH symposium on Geometry processing*, pages 166–176, 2003.

[34] R. N. Kalaria. Vascular factors in alzheimer's disease. *International Psychogeriatrics*, 15:47–52, 2005.

[35] C. Kirbas and F. Quek. A review of vessel extraction techniques and algorithms. *ACM Computing Surveys*, 36:81–121, 2004.

[36] J. Kniss, G. Kindlmann, and C. Hansen. Interactive volume rendering using multi-dimensional transfer functions and direct manipulation widgets. *Proceedings of Visualization'01*, pages 255–562, 2001.

[37] J. Kniss, A. Lefohn, R. Strzodka, S. Sengupta, and J. D. Owens. Octree textures on graphics hardware. *International Conference on Computer Graphics and Interactive Techniques*, 2005.

[38] K. Krissian, G. Malandain, N. Ayache, R. Vaillant, and Y. Trousset. Model-based detection of tubular structures in 3D images. *Computer Vision and Image Understanding*, 80(2):130–171, 2000.

[39] A. J. Lacey. *Light Microscopy in Biology: A Practical Approach.* IRL Press, Oxford, 2nd edition, 1989.

[40] D. Laur and P. Hanrahan. Hierarchical splatting: a progressive refinement algorithm for volume rendering. *ACM SIGGRAPH Computer Graphics*, 25(4):285–288, 1991.

[41] F. Lauwers, F. Cassot, V. Lauwers-Cances, P. Puwanarajah, and H. Duvernoy. Morphometry of the human cerebral cortex microcirculation: General characteristics and space-related profiles. *NeuroImage*, 39:936–948, 2008.

[42] J. Lee, P. Beighley, E. Ritman, and N. Smith. Automatic segmentation of 3D micro-CT coronary vascular images. *Medical image analysis*, 11(6):630–647, Dec. 2007.

[43] S. Lefebvre, S. Hornus, and F. Neyret. Octree textures on the GPU. *GPU Gems II*, pages 595–613, 2005.

[44] M. Levoy. Efficient ray tracing of volume data. *ACM Transactions on Graphics (TOG)*, 9(3):245–261, 1990.

[45] M. Levoy and T. Whitted. The use of points as a display primitive. *Tech. Report 85-022, University of North Carolina at Chapel Hill*, 1985.

[46] G. Li, T. Liu, J. Nie, L. Guo, J. Chen, J. Zhu, W. Xia, A. Mara, S. Holley, and S. Wong. Segmentation of touching cell nuclei using gradient flow tracking. *Journal of Microscopy*, 231(1):47–58, 2008.

[47] W. Li, K. Mueller, and A. Kaufman. Empty space skipping and occlusion clipping for texture-based volume rendering. *Proceedings of the 14th IEEE Visualization'03*, pages 317–324, 2003.

[48] W. E. Lorensen and H. E. Cline. Marching cubes: A high resolution 3D surface construction algorithm. pages 163–169, 1987.

[49] L. Lorigo, O. Faugeras, W. Grimson, R. Keriven, R. Kikinis, A. Nabavi, and C. Westin. CURVES: curve evolution for vessel segmentation. *Medical Image Analysis*, 5:195–206, 2001.

[50] Y. Masutani, T. Schiemann, and K. H. Hohne. Vascular shape segmentation and structure extraction using a shape-based region-growing model. *Proceedings of the 1st International Conference on Medical Image Computing and Computer-Assisted Intervention*, pages 1242–1249, 1998.

[51] I. Mateo, J. Llorca, J. Infante, E. Rodrguez-Rodrguez, C. Snchez-Quintana, P. Snchez-Juan, J. Berciano, and O. Combarros. Case-control study of vascular endothelial growth factor (VEGF) genetic variability in alzheimer's disease. *Neuroscience Letters*, 401:171–173, 2006.

[52] D. Mayerich, L. C. Abbott, and B. H. McCormick. Knife-Edge scanning microscopy for imaging and reconstruction of Three-Dimensional anatomical structures of the mouse brain. *Journal of Microscopy*, 231(1):134–143, July 2008.

[53] D. Mayerich, J. Kwon, Y. Choe, L. Abbott, and J. Keyser. Constructing high resolution microvascular models. In *3rd Microscopic Image Analysis with Applications in Biology Workshop*, 2008.

[54] D. Mayerich, B. H. McCormick, and J. Keyser. Noise and artifact removal in Knife-Edge scanning microscopy. *Proceedings of the 4th IEEE International Symposium on Biomedical Imaging: From Nano to Macro*, pages 556–559, 2007.

[55] D. M. Mayerich and J. Keyser. Filament tracking and encoding for complex biological networks. In *Proceedings of the 2008 ACM symposium on Solid and physical modeling*, pages 353–358, Stony Brook, New York, 2008. ACM.

[56] D. M. Mayerich, Z. Melek, and J. Keyser. *Fast Filament Tracking Using Graphics Hardware*. Technical report (submitted for review), Texas A&M University, 2007.

[57] A. Mazziotti, A. Cavallari, and J. Belghiti. *Techniques in Liver Surgery*. Greenwich Medical Media, 1997.

[58] B. McCormick, B. Busse, P. Doddapaneni, Z. Melek, and J. Keyser. Compression, segmentation, and modeling of filamentary volumetric data. pages 333–338, 2004.

[59] D. P. Mccullough, P. R. Gudla, K. Meaburn, A. Kumar, M. Kuehn, and S. J. Lockett. 3D segmentation of whole cells and cell nuclei in tissue using dynamic programming. *Proceedings of the 4th IEEE International Symposium on Biomedical Imaging: From Nano to Macro*, pages 276–279, 2007.

[60] Z. Melek, D. M. Mayerich, C. Yuksel, and J. Keyser. Visualization of fibrous and thread-like data. *IEEE Transactions on Visualization and Computer Graphics*, 12:1165–1172, 2006.

[61] K. D. Micheva and S. J. Smith. Array tomography: A new tool for imaging the molecular architecture and ultrastructure of neural circuits. *Neuron*, 55:25–36, 2007.

[62] C. I. Moore and R. Cao. The Hemo-Neural hypothesis: On the role of blood flow in information processing. *Journal of Neurophysiology*, 99(5):2035–2047, 2007.

[63] D. C. Morris, K. Davies, Z. Zhang, and M. Chopp. Measurement of cerebral microvessel diameters after embolic stroke in rat using quantitative laser scanning confocal microscopy. *Brain Research*, 876(1-2):31–36, Sept. 2000.

[64] D. Nain, A. Yezzi, and G. Turk. Vessel segmentation using a shape driven flow. *Medical Imaging Computing and Computer-Assisted Intervention*, pages 51–59, 2004.

[65] Nielsen and Museth. Dynamic tubular grid: An efficient data structure and algorithms for high resolution level sets. *Journal of Scientific Computing*, 26:261–299, 2006.

[66] J. F. O'Brien and N. F. Ezquerra. Automated segmentation of coronary vessels in angiographic image sequences utilizing temporal, spatial and structural constraints. pages 25–37, 1994.

[67] A. Okabe, B. Boots, and K. Sugihara. *Spatial Tessellations: Concepts and Applications of Voronoi Diagrams*. John Wiley & Sons, Inc., New York, NY, 1992.

[68] J. Orchard and T. Moller. Accelerated splatting using a 3D adjacency data structure. *Proceedings of Graphics Interface 2001*, pages 191–200, 2001.

[69] S. J. Osher and R. P. Fedkiw. *Level Set Methods and Dynamic Implicit Surfaces*. Springer, 2002.

[70] J. B. Pauley, editor. *Handbook of Biological Confocal Microscopy*. Plenum Press, New York, 1995.

[71] V. Petrovic, J. Fallon, and F. Kuester. Visualizing Whole-Brain DTI tractography with GPU-based tuboids and LoD management. *Transactions on Visualization and Computer Graphics*, 13:1488–1495, 2007.

[72] H. Pfister, M. Zwicker, J. van Baar, and M. Gross. Surfels: surface elements as rendering primitives. *Proceedings of the 27th Annual Conference on Computer Graphics and Interactive Techniques*, pages 335–342, 2000.

[73] F. H. Post, B. Vrolijk, H. Hauser, R. S. Laramee, and H. Doleisch. The state of the art in flow visualisation: Feature extraction and tracking. *Computer Graphics Forum*, 22(4):775–792, 2003.

[74] A. R. Pries, T. W. Secomb, P. Gaehtgens, and J. F. Gross. Blood flow in microvascular networks. experiments and simulation. *Circulation Research*, 67(4):826–834, 1990.

[75] A. Puig, D. Tost, and I. Navazo. An interactive cerebral blood vessel exploration system. *Proceedings of the 8th Conference on Visualization '97*, pages 433–436, 1997.

[76] L. Ren, H. Pfister, and M. Zwicker. Object space EWA surface splatting: A hardware accelerated approach to high quality point rendering. *Computer Graphics Forum*, 21(3):461–470, 2002.

[77] A. Reymond, V. Marigo, M. B. Yaylaoglu, A. Leoni, C. Ucla, N. Scamu?a, C. Caccioppoli, E. T. Dermitzakis, R. Lyle, S. Ban?, G. Eichele, S. E. Antonarakis, and A. Ballabio. Human chromosome 21 gene expression atlas in the mouse. *Nature*, 420:582–586, 2002.

[78] C. Rezk-Salama, K. Engel, M. Bauer, G. Greiner, and T. Ertl. Interactive volume on standard PC graphics hardware using multi-textures and multi-stage rasteriza-

tion. *Proceedings of the ACM SIGGRAPH/EUROGRAPHICS Workshop on Graphics Hardware*, pages 109–118, 2000.

[79] B. Roysam, W. Shain, and G. A. Ascoli. The central role of neuroinformatics in the national academy of engineering's grandest challenge: reverse engineer the brain. *Neuroinformatics*, 7(1):1–5, 2009. PMID: 19140032.

[80] Y. Sato, S. Nakajima, N. Shiraga, H. Atsumi, S. Yoshida, T. Koller, G. Gerig, and R. Kikinis. Three-dimensional multi-scale line filter for segmentation and visualization of curvilinear structures in medical images. *Medical Image Analysis*, 2:143–168, 1998.

[81] H. Schmitt, M. Grass, V. Rasche, O. Schramm, S. Haehnel, and K. Sartor. An x-ray-based method for the determination of the contrast agentpropagation in 3-D vessel structures. *IEEE Transactions on Medical Imaging*, 21:251–262, 2002.

[82] P. Schroder and G. Stoll. Data parallel volume rendering as line drawing. pages 25–32, 1992.

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

[84] D. Spector and R. Goldman. *Basic Methods in Microscopy*. Cold Spring Harbor Laboratory Press, Cold Spring Harbor, NY, 2006.

[85] C. Stoll, S. Gumhold, and H. Seidel. Visualization with stylized line primitives. In *Proceedings of IEEE Visualization*, pages 695–702, 2005.

[86] T. Tozaki, Y. Kawata, N. Niki, H. Ohmatsu, K. Eguchi, and N. Moriyama. Three-dimensional analysis of lung areas using thin slice [CT] images. volume 2709, pages

1–11, 1996.

[87] P. S. Tsai, B. Friedman, A. Ifarraguerri, B. D. Thompson, V. Lev-Ram, C. B. Scha?er, Q. Xiong, R. Y. Tsien, J. A. Squier, and D. Kleinfeld. All-Optical histology using ultrashort laser pulses. *Neuron*, 39:27–41, 2003.

[88] M. Wiercigroch and E. Budak. Sources of nonlinearities, chatter generation and suppression in metal cutting. *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 359:663–693, 2001.

[89] M. Woo, J. Neider, T. Davis, and D. Shreiner. *OpenGL Programming Guide*. Addison-Wesley Professional, 3rd edition, 1999.

[90] S. Worz and K. Rohr. Segmentation and quantification of human vessels using a 3-D cylindrical intensity model. *IEEE Transactions on Image Processing*, 16(8):1994–2004, 2007.

[91] S. R. y Cajal. *Histologie du systeme nerveux de l'homme et des vertbrs*, volume 2. Consejo Superior de Investigaciones Cienti?cas, Madrid, 1911.

[92] S. P. Yang, D. G. Bae, H. J. Kang, B. J. Gwag, Y. S. Gho, and C. B. Chae. Co-accumulation of vascular endothelial growth factor with beta-amyloid in the brain of patients with alzheimer's disease. *Neurobiology of Aging*, 25:283–290, 2004.

[93] Y. Zhang, Y. Bazilevs, S. Goswami, C. L. Bajaj, and T. J. R. Hughes. Patient-specific vascular NURBS modeling for isogeometric analysis of blood flow. *Computer Methods in Applied Mechanics and Engineering*, 196(29-30):2943–2959, 2007.

[94] H. Zhao. A fast sweeping method for eikonal equations. *Mathematics of Computation*, 74:603–627, 2004.

[95] B. Zlokovic. The Blood-Brain barrier in health and chronic neurodegenerative disorders. *Neuron*, 57:178–201, 2008.

[96] M. Zwicker, H. Pfister, J. van Baar, and M. Gross. Surface splatting. *Proceedings of the 28th Annual Conference on Computer Graphics and Interactive Techniques*, pages 371–378, 2001.

[97] M. Zwicker, H. Pfister, J. van Baar, and M. Gross. EWA splatting. *IEEE Transactions on Visualization and Computer Graphics*, 8(3):223–238, 2002.

VITA

Name:	David Matthew Mayerich
Address:	Department of Computer Science and Engineering Texas A&M University TAMU 3112 College Station, TX 77843-3112
Email Address:	mayerichd@neo.tamu.edu
Education:	B.S., Computer Science, Southwestern Oklahoma State University, 2000 M.S., Computer Science, Texas A&M University, 2003 Ph.D., Computer Science, Texas A&M University, 2009