

MVN-ANALYSIS: SOFTWARE TO CHARACTERIZE THE STRUCTURE AND FLUID
FLOW FORCES WITHIN ENGINEERED MICROVASCULAR NETWORKS

by

Ryan Armstrong

Senior Honors Thesis

Department of Physics

University of North Carolina at Chapel Hill

April 2020

This thesis has been reviewed by the research advisors and the departmental coordinator for the honors program and has been found to be satisfactory.

Dr. Boyce Griffith, Advisor

Dr. William Polacheck, Co-Advisor

Dr. Lu-Chang Qin, Departmental Coordinator for
the Honors Program

ACKNOWLEDGMENT

This thesis would not have been possible without the generosity of many wonderful people. I would like to thank my advisors, Dr. Boyce Griffith and Dr. Bill Polacheck, for providing me with the opportunity to learn through this project and their guidance along the way. Further, I owe a massive debt of gratitude to Dr. Simone Rossi, who not only provided the fluid solver used in this report, but also replied to a countless number of confused emails with unwavering patience and support.

Additionally, Crescentia Cho produced and imaged the microvascular networks used in the analysis with assistance from Joanna McDonald—without which, there would be no project. Similarly, Margaret Anne Smith donated her time to teach me the Trellis interface, which greatly increased my productivity.

Beyond the technical support, I have always been encouraged by my parents, Laura and Doug; my siblings, Cate, Holden and Harry; and my best friend, Andrea Brucculeri. I would also like to thank (or maybe blame) Samantha Pagan, Patrick Taylor, and Maggie Hildebrand for inspiring me to pursue a thesis. I miss the 2 a.m. walks from Phillips to Cosmic.

Abstract

Microvascular networks consist of interconnected blood vessels $\lesssim 150 \mu\text{m}$ in diameter and are responsible for transporting metabolites and nutrients to the body's cells. New microvascular blood vessels are formed from existing vessels through the process of angiogenesis. While many chemical gradients controlling angiogenesis are known, the role of hemodynamic forces—such as wall shear stress—are less understood. Controlling angiogenesis could be therapeutic to ischemic heart disease and can limit cancer metastasis, among others. Therefore, with the motivation of supporting the research of angiogenesis, we created a software suite capable characterizing the structure and fluid flow forces within imaged microvascular networks.

TABLE OF CONTENTS

	Page
ACKNOWLEDGMENT	ii
ABSTRACT	iii
LIST OF TABLES	vi
LIST OF FIGURES	vii
1 Introduction	1
1.1 Polacheck Microfluidic Devices	2
1.2 Image Segmentation	4
1.2.1 MVN Imaging	4
1.2.2 Segmentation Challenges	5
1.3 Necessary Fluid Dynamics	6
1.3.1 Reynolds Number	6
1.3.2 Stokes Flow	7
1.3.3 Wall Shear Stress	7
1.4 Open Source Image Processing	8
2 Methods	12
2.1 Summary of Major Outputs	13
2.2 Image Pre-Processing	15
2.3 <code>mvn-analysis</code> Computations	16
2.3.1 2D Segmentation	16
2.3.2 2D Weighted Graphs	17
2.3.3 2.5D Surface Mesh	24
2.3.4 3D Segmentation and Surface Mesh	29
2.3.5 Surface Meshes to Volume Meshes	37
2.3.6 Numerical Outputs	37

2.4	Performance Notes	37
2.4.1	Speed	37
2.4.2	Memory	40
3	Stokes Flow Simulations	41
3.1	Prepossessing	41
3.2	Visualizing Wall Shear Stress	43
3.3	Relevance	44
4	Preliminary Validation	45
4.1	Cross-Method Validation	45
4.2	T-Junction Artificial Data	46
5	Preliminary Data Exploration	48
5.1	Serum Depletion	48
5.2	Hypoxia	53
5.2.1	Summary	53
6	Current and Future Work	55
6.1	3D Graph Representation	55
6.2	Validation	55
6.3	Error Analysis	56
REFERENCES	.	59
APPENDIX		
A	Major Outputs of Additional Examples	61
B	Input Files for Sample Outputs	62
B.1	Sample A	62
B.2	Sample B	67

LIST OF TABLES

2.1	Graph summary values generated for Samples A and B.	22
2.2	Directionality scores for Samples A and B.	23
2.3	Volumetric data generated for Sample A.	38
2.4	Volumetric data generated for Sample B.	39
4.1	Sample B volume calculations.	46

LIST OF FIGURES

1.1	A microfluidic device similar to the once used in this study.	3
1.2	MVN _s created by the Polacheck lab in devices similar to the ones described. Horizontal parent channels are shown at the top of (a) and through the center of (b). HUVECs are labeled green.	4
1.3	Third party software 3D segmentations.	5
2.1	Flattened and contrast enhanced images of Sample A and Sample B. . . .	13
2.2	Major outputs from the 2D Segmentation step. Only the largest connected regions have been retained. <i>Left</i> : Enhanced contrast. <i>Middle</i> : Binary Mask. <i>Right</i> : Distance Transform and Skeleton.	18
2.3	Graph representation (nodes and edges) of Sample A and B overlayed on their respective flattened images. The blue lines show the skeleton used to generate the graph and the white lines show the graph edges. The original image can be found on GitHub if there is difficulty viewing the graph edges.	21
2.4	Local directionality. Blue ellipses show x, y distribution while red ellipses show y=x, y=-x distribution.	24
2.5	Comparison between enforced circular lumens (a, b) and enforced z-depth (c, d) for a highly planar network (Sample A).	27

2.6	Comparison between enforced circular lumens (a) and enforced z-depth (b, c) a highly three dimensional network.	28
2.7	Examples of the binary images at slices ($n/39$) through the Sample A 3D reconstruction	29
2.8	Performance of the lumen filling methods on Sample A. The blue represent the filling achieved by the ellipsoidal octant method and the red represents the filling achieved by ray tracing.	32
2.9	"Honest" scaling 3D mesh (a, c) and "rounded" scaling 3D mesh (b, d) for Sample A. For the rounded scaling, 0.5 prefactor was applied to the distance transforms.	35
2.10	Demonstration of vessels crossing in different z-planes. These the vessels shown are connected in the 2D and 2.5D representations of the network. . .	36
2.11	2.5D (a) and 3D (b) segmentations for Sample B.	36
3.1	Tetrahedral mesh of the lumen used in the Stokes flow example.	42
3.2	Visualizations of pressure and velocity distributions for the test flow network. .	43
3.3	Distribution of wall shear stress for the test flow network.	44
4.1	Review of artificial data segmentation.	47
5.1	Distribution of branch contraction factors for the serum depletion experiment.	49
5.2	Distribution of branch fractal dimensions for the serum depletion experiment.	49
5.3	Distribution of branch lengths factors for the serum depletion experiment. .	50
5.4	Distribution of branch radii for the serum depletion experiment.	50
5.5	Distribution of branch volumes for the serum depletion experiment.	51
5.6	Distribution of branch connectivity for the serum depletion experiment. . .	51

5.7	Distribution of branch surface area for the serum depletion experiment.	52
5.8	Positive trend between serum depletion and average segment radius.	52
5.9	Caption	53

Chapter One

Introduction

The cardiovascular system serves as the body's scale-linking mechanism. It connects the respiratory, digestive, endocrine, and excretory systems to individual cells by transporting gases (oxygen and carbon dioxide), nutrients, enzymes, hormones, and heat [1]. The actual exchange of these molecules and energy primarily occurs at the microvascular level where diameters of blood vessels are $\lesssim 150 \mu\text{m}$ [2]. Further, these exchanged metabolites and nutrients only diffuse a few hundred microns within tissue. Therefore, viable, metabolically active cells must have nearby access to microvascular blood vessels [3].

Creation of new microvascular vessels occurs through the process of angiogenesis. During this process, already formed vessels either sprout (sprouting angiogenesis) or split (intussusceptive angiogenesis) to create new vessels. Many biochemical factors and co-factors are known to stimulate or inhibit the process, especially for sprouting angiogenesis. For example, the fibroblast growth factor (FGF) and vascular endothelial growth factor (VEGF) proteins promote angiogenesis [4]. Therefore, hypoxic cells—which up-regulate FGF receptors—stimulate new vessel growth, while molecules that bind to VEGF—such as the drug Bevacizumab—inhibit angiogenesis [5, 6]. However, for both sprouting and intussusceptive angiogenesis, hemodynamic forces, such as wall shear stress (WSS) and pressure, also influence levels of vessel formation. Though, these factors are less understood than their chemical counterparts. Most studies report increases in angiogenesis with increases in shear stress [7–9], however a

This is subtle & beyond the scope, but the effect of flow on angiogenesis is quite complex. VEGFR, the cognate receptor for VEGF is activated in different manners & rates depends on flow

minority of studies report the opposite and claim decreases in angiogenesis with increases in shear stress [10].

Understanding the factors, including the hemodynamic factors, that control angiogenesis is clinically relevant. Stimulation of angiogenesis could be therapeutic to ischemic heart disease, peripheral arterial disease and wound healing. Alternatively, inhibiting angiogenesis could limit cancer metastasis, ophthalmic conditions, and rheumatoid arthritis [11].

Motivated by these applications, the Polacheck lab developed a novel microfluidic platform capable of supporting the growth of perfusible, three-dimensional microvascular networks (MVs) of human umbilical vein endothelial cells (HUVECs).¹ Compared to *in vivo* platforms, the Polacheck networks offer better control over the known chemical gradients that drive angiogenesis, and therefore can better isolate mechanical causes. However, the network vessels are too small to directly measure the fluid forces and too topologically complex to manually quantify structurally. Therefore, with the ultimate goal of investigating the relationship between fluid flow forces and angiogenesis, we developed a software suite (mvn-analysis) capable of characterizing the structure and fluid flow forces within the Polacheck microvascular networks.

1.1 Polacheck Microfluidic Devices

The work presented in this thesis is focused on the development and use of computational tools and not on the biology or development of the microvascular networks. However, a general understanding of the Polacheck microfluidic devices helps motivate the need for these computational tools.

Broadly, the Polacheck microfluidic devices are PDMS-on-glass chambers containing physiologic extracellular matrix (ESM) seeded with HUVECs and human lung fibroblasts. During

¹The methods paper for the Polacheck microvascular network platforms used in this study is not yet published. Though, see Polacheck *et al.* (2019) for similar [12].

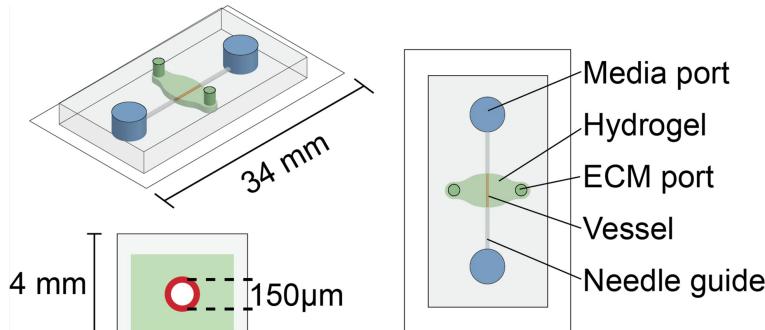


Figure 1.1 A microfluidic device similar to the once used in this study.

the construction of the devices, sterile needles are used to negatively cast two parallel channels in the ESM. These channels, referred to as parent channels, are seeded with HUVECs and irrigated with cell culture media by a rocker. Figure 1.1 shows a single channel variant of a similar device from the work published in Polacheck *et al.* (2019) [12].

Within successful devices, the ECM HUVECs create a network with open lumens. These networks undergo self directed growth and are expected to connect to the parent channels via anastomosis. Figure 1.2 shows examples of microvascular networks created in these types of devices.

The diameter, separation, and distance from the glass of the parent channels are all device variables currently being explored. However, the diameters are on the order of $\sim 150 - 300 \mu\text{m}$, the separations are on the order of $10^1 - 10^3 \text{ mm}$ and the distance from the glass is in order of $10^2 \mu\text{m}$. In addition, the composition of the ECM and the irrigation conditions (rocker settings) are also being investigated.

As can be seen from Figure 1.2, the MVNs are structurally complex. Therefore, changes in the device parameters are difficult to relate to changes in the networks. While the ultimate goal of this research is to relate fluid flow forces to angiogenesis, any computational methods able characterize the MVNs would be beneficial to standardizing their manufacturing parameters. This application is explored in Chapter 4, where `mvn-analysis` is used to compare the networks generated by devices varying in ECM composition and level of irrigation.

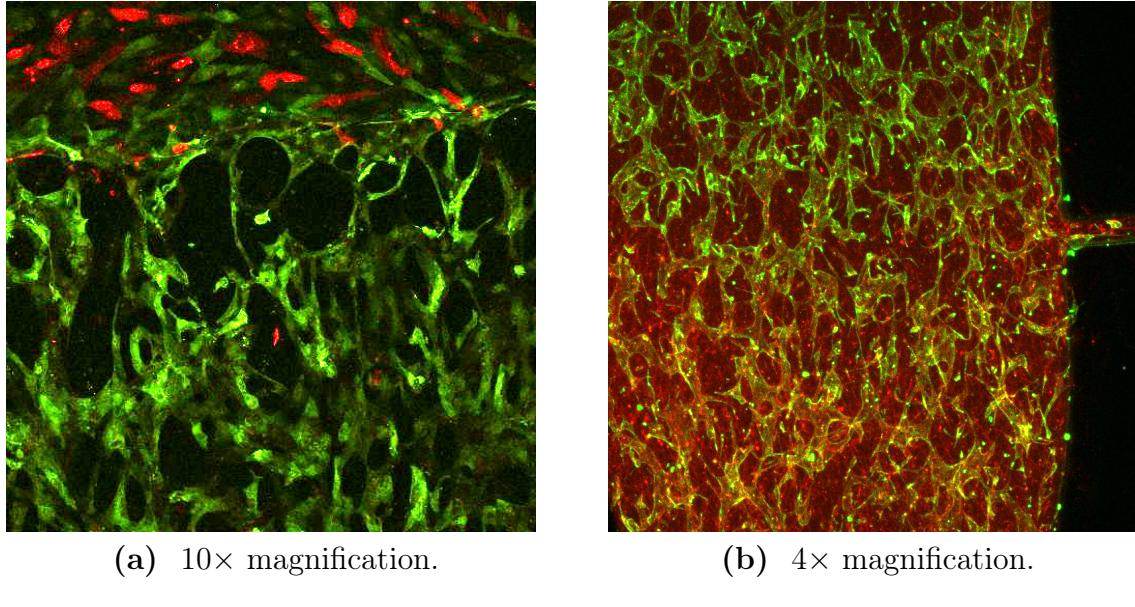


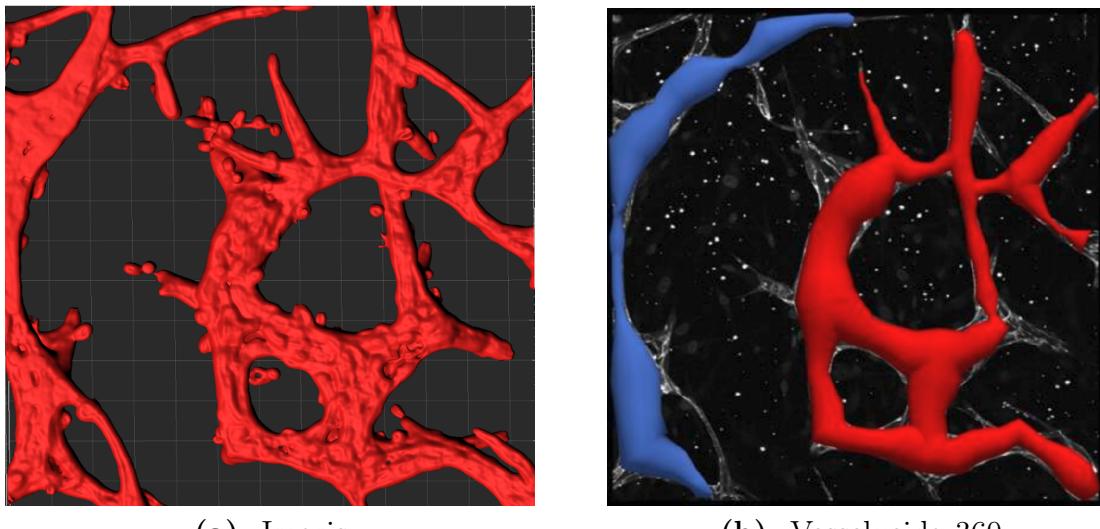
Figure 1.2 MVNs created by the Polacheck lab in devices similar to the ones described. Horizontal parent channels are shown at the top of (a) and through the center of (b). HUVECs are labeled green.

1.2 Image Segmentation

Within the context of this research, image segmentation relates to distinguishing between pixels that represent vessel lumens and pixels of non-lumenized space.

1.2.1 MVN Imaging

The HUVEC used in the Polacheck devices were stably transduced using lentiviral transduction particles to constitutively express GFP (green) and Ruby (red). The cells of the microvascular network express GFP and therefore are shown as green in Figure 1.2. Imaging of the MVNs was conducted using an Olympus FV3000 confocal microscope to capture discrete parallel planes along the z-axis.



(a) Imaris.

(b) Vesselucida 360

Figure 1.3 Third party software 3D segmentations.

1.2.2 Segmentation Challenges

Segmenting the MVNs may initially appear trivial, however accurate, three dimensional reconstructions of large scale microvascular networks continues to be a challenge. As tubular objects at the micron scale, MVN segmentations are prone to distortions and artifacts [13]. Further, because the network is highly connected, structural modeling errors at one location may yield effects on distant parts of the network.

Additionally, to obtain the z-sliced images, we are only able to image the network fluoresce. This fluoresce comes from the GFP expressed in the HUVECs. Therefore, images only represent the cells (not the lumens). Further, GFP is not expressed uniformly across a cell's surface or between different cells. Together, this means image segmentation of the microvascular networks consists of determining lumen locations from images of the lumen surfaces that are discontinuous in all dimensions and of varying intensity.

There exist open source and enterprise software with functionalities designed to create these segmentations. Trials using Imaris and Vesselucida 360 are shown in Figure 1.3.²

Both the Imaris and Vesselucida segmentations have positive qualities. Imaris was able to

²The open source software, ImageJ produced results similar to Imaris.

keep vessels distinct that appear to cross but are on different z-planes. Vesselucida produced a completely smooth an output with no artefacts. However, the Imaris segmentation is not smooth and the lumen contains holes. Similarly, the Vesselucida segmentation neglects many smaller branches.

It is possible that with extra manual labor (pre-processing and post-processing), third party software could have provided the segmentations required for this project. However, the demonstrations with Imaris and Vesselucida show that the problem of segmenting microvascular networks is not yet completely solved.

1.3 Necessary Fluid Dynamics

1.3.1 Reynolds Number

The Reynolds number is a dimensionless quantity that can be used to predict fluid flow patterns. Specifically, it is the ratio between internal flow forces and viscous flow forces. Therefore, high Reynolds number flows are dominated by internal forces and are expected to be turbulent while low Reynolds number flows are dominated by viscous forces and are expected to be laminar. The Reynolds number is defined as,

$$Re = \frac{\rho u D}{\mu} \quad (1.1)$$

where ρ is the density of the fluid, u is the fluid velocity, D is the diameter of the tube, and μ is the fluid's viscosity. Thus, for fluids of constant density and viscosity, the Reynolds number is maximized by maximizing velocity and diameter.

Initial fluid experiments with the Polacheck MVNs plan to flow cell culture media. The cell culture media is primarily water and therefore is expected to be incompressible with an approximate density of $992.2 \frac{\text{kg}}{\text{m}^3}$ and viscosity near $0.653 \times 10^{-3} \frac{\text{kg}}{\text{ms}}$ at 37 C [14]. Further, *in vivo* experiments have measured blood velocity within the microcirculation to an upper range of $\sim 2 \text{ mm/s}$ [15]. Finally, the maximum diameters observed in the Polacheck MVNs

are $\lesssim 300 \mu\text{m}$. Therefore, within the Polacheck networks, the maximum Reynolds number should be $\sim 1 \times 10^{-4}$.

The recognized transition between laminar and turbulent flow occurs near $\text{Re} = 2300$. Therefore the flow within the network is clearly laminar. Further, as the flow satisfies the limit $\text{Re} \rightarrow 0$, the Stokes equations are a valid representation of the fluid dynamics [16].

1.3.2 Stokes Flow

As stated, Stokes flow occurs when $\text{Re} \rightarrow 0$. Under such conditions, the internal flow forces are negligible in comparison to the viscous flow forces. Therefore, for an incompressible, Newtonian fluid, such as cell culture media (water), the incompressible Naiver Stokes equations reduce to incompressible Stokes equations,

$$\mu \nabla^2 \mathbf{u} - \nabla p + \mathbf{f} = 0 \quad (1.2)$$

$$\nabla \cdot \mathbf{u} = 0 \quad (1.3)$$

where, p is pressure, \mathbf{u} , is the fluid velocity vector, and \mathbf{f} is an applied force. Equation 1.2 represents the momentum balance of the fluid and Equation 1.3 represents the incompressibility of the fluid. Unlike the full Naiver Stokes equations, the Stokes equations are linear in velocity and pressure. Therefore, they are more attractive for computational modeling [16].

1.3.3 Wall Shear Stress

Fluid flow in microvascular networks have been observed to obey a no-slip boundary on the vessel walls [8]. Therefore, the gradient of the fluid velocity ~~must~~ most be non-zero and the vessel wall must experience a shear stress. The shear stress at the vessel wall can be calculated by subtracting the normal component of the ~~viscous~~ stress tensor from the total ~~viscous~~ stress tensor [17],

$$\mathcal{T} - (\mathcal{T} \cdot \mathbf{n})\mathbf{n} \quad (1.4)$$

where \mathbf{n} is the unit normal vector to the vessel wall and \mathcal{T} is the viscous stress tensor,

$$\mathcal{T} = \mu(\nabla\mathbf{u} + (\nabla\mathbf{u})^T) \cdot \mathbf{n} \quad (1.5)$$

Therefore, fluid flow solutions that obtain velocity fields through the lumen can be used to calculate wall shear stress for a given viscosity. Further, for Newtonian fluids (constant viscosity), relative wall shear stress can be calculated from only the velocity profile, as WSS is linearly dependant on viscosity.³

1.4 Open Source Image Processing

Many of the steps in the developed image processing pipelines utilize third party software, such as `scikit-image` [20] and `SciPy` [21]. These packages are excellently documented, so the following descriptions focus on usage and effects rather than their implementations. The filters and computations described below will be referenced later in the discussion of the custom code elements. The euclidean distance transform and Gaussian blur use `SciPy`'s implementation. The rest use `scikit-image`.

Those familiar with common image processing filters and transformations can skip this section (1.4) without loss of continuity. ← very courteous

Contrast Enhancement

Each pixel in a single channel image has a finite range of possible intensity values. In general, we expect the vascularized regions to have higher pixel intensities than regions of empty space. Therefore, enhancing the contrast between background and vessel pixels is beneficial for segmentation.

³This paper focuses on the development of a process to determine wall shear stress within microvasculature. For more information on the biological pathway causing mechanically stimulated angiogenesis, see Polacheck *et al.* (2017), de la Paz *et al.* (2012), and Egginton *et al.* (2016) [9, 18, 19].

`scikit-image` provides three methods to alter the distribution of pixel intensities within an image. Contrast stretching scales the image pixel values to include all intensities which fall within a percentile range (often 2 to 98). Histogram equalization scales the image pixel such that the cumulative distribution of intensities is linear. Finally, adaptive histogram equalization behaves as a local version of histogram equalization (linearizing cumulative distributions within subsections of the image) [20].

Opening and Closing

Morphological opening is the process of eroding (shrinking bright regions and expanding dark regions) followed by dilation (expanding bright regions and shrinking dark regions). Opening has the effect of eliminating small bright areas and opening thin dark cracks. Morphological closing represents the opposite process (dilation followed by erosion). It has the effect of connecting nearby bright objects and closing thin dark cracks[20]. In particular, opening is beneficial for removing noise in the vessel segmentation while closing is useful for repairing broken vessel connections.

Reconstruction via Dilation

Reconstruction uses the dilation operation to isolate connected regions in an image [20]. This can be effective for segmenting images that have particularly sparse fluoresce, however aggressive use of the filter over connects the vessel segments.

Minimum Cross-Entropy Thresholding

Cross-entropy thresholding returns the intensity value that minimizes cross-entropy between the foreground and the foreground mean (lumens), and the background and the background mean (non-lumens). This value can then be used to convert the image to binary. The implementation of this method tests every possible threshold [22]. The benefit of cross-entropy thresholding is it requires no user input.

Li Thresholding

Li thresholding operates under the same theory as cross-entropy, however uses an iterative approach to finding the entropy minimum (rather than testing all possible values). Therefore, this implementation is quicker, but may return a local minimum [23].

Random Walker Segmentation

Random walker segmentation allows the user to label an image with known areas background (non-lumen) and foreground (lumen). The unknown regions are then assigned to the two groups by solving an anisotropic diffusion equation such that diffusion is difficult across high gradients [20, 24]. This is particularly useful as a backup segmentation method when cross-entropy produces poor results. The disadvantage is the requirement of user inputs (although `mvn-analysis` helps provide these inputs).

Sauvola Local Thresholding

Sauvola local thresholding determines thresholding values for images within a user defined window. The smaller the window, the more local the threshold and the more detail (and artefacts) will be extracted. This is especially useful in 3D segmentation as information from adjacent z-plane can be utilized [25].

Skeletonization

Skeletonization reduces a binary objects to a 1 pixel wide representation. It is useful for extracting the center line of segmented vessels [26].

Euclidean Distance Transform

The euclidean distance transform sets the intensity of pixels to their minimum euclidean distance (in pixels) to the background. Used on lumen segmentations, this transformation

is useful for tracking cell widths. Used on the inverse of a lumen segmentation (lumens are dark, non-lumens are light), the euclidean distance transform can be used to determine distance from a vessel (i.e. distance from an nutrient source).

Gaussian Blur

This blurs the image. It has the effect of smoothing and therefore can limit artefacts when generating a skeleton. To a certain degree, it can also be used to fill holes prior to segmentation.

Marching Cubes

The marching cubes algorithm extracts a 2D surface from a 3D volume. This algorithm is used to generate the surface mesh OBJs of the lumens [20].

Chapter Two

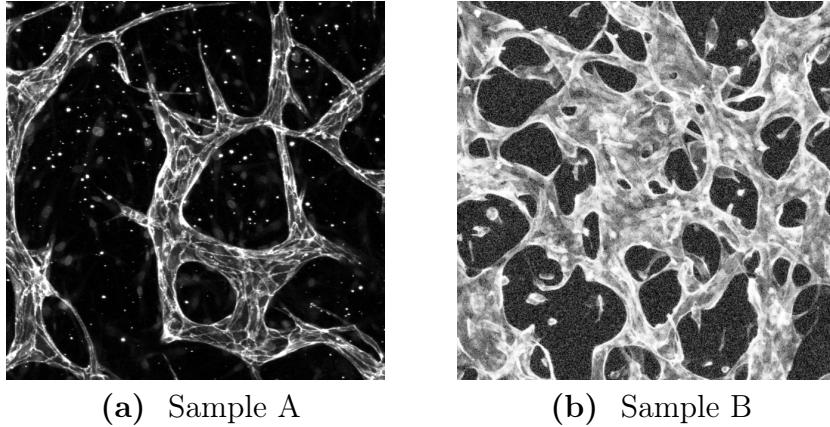
Methods

The following processes were developed specifically for images produced by the Polacheck lab. However, the software is freely available (github.com/Ryan-A-Armstrong/mvn-analysis) and accepts any single-channel, multi-page, .tif image.

Further, the described computations were run locally on a laptop with 8 GB of RAM and an Intel i7 2.8 GHz Quad Core processor. When possible, the software parallelizes computations across all available processing units. Additionally, the software automatically limits the computation resolution in an attempt to keep memory usage under ~ 7 GB.¹ For instructions on how to use the software, see the documentation on GitHub.

Throughout this chapter, two example data sets will be routinely used to demonstrate the software's major functionalities and outputs. Figure 2.1 shows these data sets as flattened, contrast enhanced, two dimensional images. Sample A is section of a network that is relatively simple and planar. Sample B is a section of a network that is relatively more complex and three dimensional. These samples were chosen due to their differences, however they alone do not represent all networks. Therefore, Appendix A contains major outputs of additional samples. I recommend exploring these outputs to better understand the performance of the software on a more diverse range of inputs.

¹The 7 GB maximum usage is only for `mvn-analysis` memory usage and does not protect against *total* memory usage. i.e. running other programs along with `mvn-analysis` may cause memory overages.



(a) Sample A

(b) Sample B

Figure 2.1 Flattened and contrast enhanced images of Sample A and Sample B.

Finally, the input files used to generate the outputs of Samples A and B are included in Appendix B.

2.1 Summary of Major Outputs

Code written for `mvn-analysis` has two major goals. Goal 1 is to extract a volume mesh representing the lumens that can be used in fluid dynamics simulations. Goal 2 is to opportunistically collect as much data along the way that could be used to characterize or compare MVNs. The following serves as a reference for the major data types a user has access to.

2D Segmentation

- **2D Binary Mask:** A two dimensional binary image where vessel lumens are stored as high values and the background is stored as low values.
- **2D Skeleton:** A binary image representing the centerlines of the vessel lumens.
- **2D Distance Transform:** A grayscale image where the intensity of a pixel is the euclidean distance of the pixel from the nearest background pixel.

2D Weighted Graphs

- **NetworkX Graph:** A graph representation (nodes and weighted edges) of the two dimensional microvascular network segmentation. Nodes represent vessel branches and termini. Edges represent the connections between the nodes and carry weights related to the physical structure of the network.
- **Graph Characteristics:** Quantification and distributions of vessel segment lengths, volumes, surface area, average radii, contraction factor, fractal dimensions, and node connectivity.
- **Graph Comparisons:** Histogram and normal fit comparisons of characteristics between two or more separate networks (different networks, the same network over time, or the same network generated with graphs of different parameters).
- **Directionality Analysis:** Empirical measures and qualitative visualizations for the directionality (vessel alignment) within a network.

25D Surface Meshes

- **Surface Mesh:** Watertight OBJ mesh representing the lumen surface. This mesh is based on the two dimensional segmentation and assumes ellipsoidal lumens.

3D Segmentation and Surface Meshes

- **3D Binary Mask:** A three dimensional binary image where vessel lumens are stored as high values and the background is stored as low values.
- **Inverted 3D Distance Transform:** A three dimensional image where pixel values represent the distance to the nearest vessel wall (from outside the vessel).
- **3D Skeleton:** A three dimensional binary image representing the centerlines of the vessel lumens.
- **Honest-Scaling 3D Surface Mesh:** OBJ mesh representing the lumen surface. This

mesh is based on analyzing each slice of the input image and is z-scaled to the physical image acquisition parameters.

- **Rounded-Scaling 3D Surface Mesh:** OBJ mesh representing the lumen surface. This mesh is based on enforcing rounded lumens to the honest scaling representation.

Volume Meshes

- **Tetrahedral Meshes:** Each described OBJ can be converted to a tetrahedral mesh representing the lumen volume. This mesh is guaranteed to have no internal holes and can also be converted to ExodusII format for flow simulations.

Numerical Outputs

- **Volume Analysis:** Each three dimensional segmentation can also calculate volume based data. This data includes total lumen volume, vessel density, and average distance to the nearest vessel wall (from empty space).

2.2 Image Pre-Processing

The Polacheck lab imaged the microvascular networks using fluorescent confocal microscopy. The output images captured vessel walls at parallel planes along the z -axis and were saved in Olympus' proprietary multi-page, multi-channel, `.oir` file format.

ImageJ was used to open the `.oir` files in grayscale and convert them to `.tif` format. Additionally, ImageJ was used to read the metadata and record the scaling information for the sample (microns / pixel in the x , y , and z directions).

Then, if the original image was multi-channeled, the `Stack To RBG` command was used to merge the channels. Finally, the `Image Type` command was used to convert from RBG to 8-Bit grayscale. The resulting single-channel, multi-page, `.tif` image was saved to be used as the data input to `mvn-analysis`.

2.3 mvn-analysis Computations

The following processes all occur within the `mvn-analysis` software. For more information on the effects of specific sub-processes and setting parameters, see the documentation on Github.

2.3.1 2D Segmentation

Reading .tif Stacks and Setting the Global Unit

Each page of the input `.tif` is converted to a two dimensional NumPy array. The array size is then assessed to determine if any memory saving actions are required. Specifically, the maximum length of an image axis is set to 512 pixels and 750 microns. Violations of the pixel length maximum will lead to down-scaling the image axis to 512 pixels. Violation of the physical length maximum will yield an increase in global unit size (the micron / pixel scale of `mvn-analysis` outputs²). These limits are set to keep memory usage under 7 GB throughout the entire analysis pipeline, not just the 2D segmentation. The downscale factor and the unit conversion are stored as parameters for later calculations in order to maintain the proper physical representation. Finally, the stack of image arrays are flattened to a single 2D representation by summing the pixels along the z-axis.

Generating and Scaling the Outputs

The contrast of the flattened image is increased via one of the three methods described in section 1.4 (user chosen). Then, small holes in the vessel ares are identified by taking the difference between the morphological closing and the original contrast enhanced image.

²`mvn-analysis` defaults to producing outputs of 1 micron/pixel, which I refer to as the global unit. For large inputs, this unit is increased in integer amounts until the memory requirements are met. i.e. a sample that spanned 500 microns would produce outputs at 1 micron/pixel while a sample that spanned 1200 microns would produce outputs at 2 microns/pixel.

These dark spots (now represented as light spots) are added back to the enhanced image. Non-vessel light spots are filtered by taking the morphological opening of the working image. The background and vessels are then segmented using dilation reconstruction. From there, the image is thresholded by minimum cross-entropy or random-walker segmentation (user defined).

The resulting binary image has opening and closing performed (to remove artefacts and connect broken vessel segments). If specified by the user, the image is separated into connected regions and retains only the largest segment. The binary image is then up-scaled such that each pixel physically represents the global unit discussed earlier. The resulting mask is smoothed by a Gaussian blur to reduce surface roughness that can contribute to skeletonization artefacts.

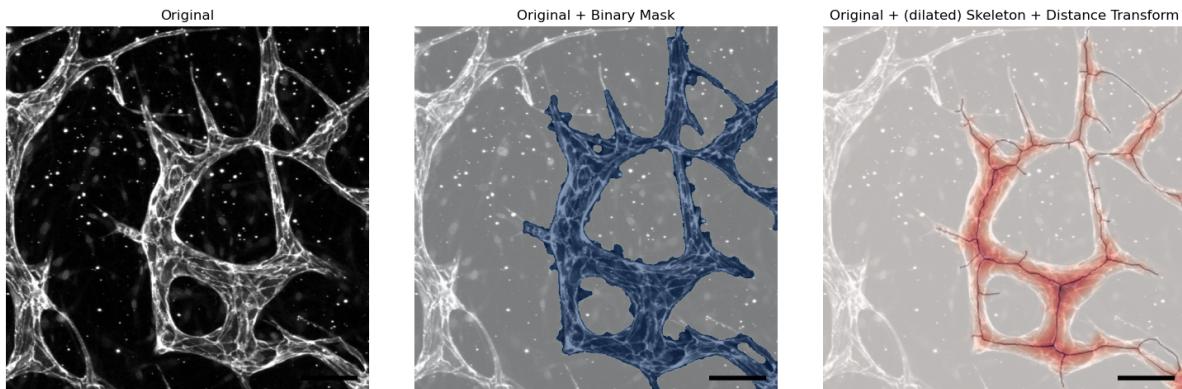
Finally, the binary mask is converted to a skeleton and a euclidean distance transform. The binary mask, skeleton, and distance transform are all stored as inputs to later processes. Figure 2.2 shows a summary of the major outputs of the 2D segmentation when performed on Sample A and Sample B.

2.3.2 2D Weighted Graphs

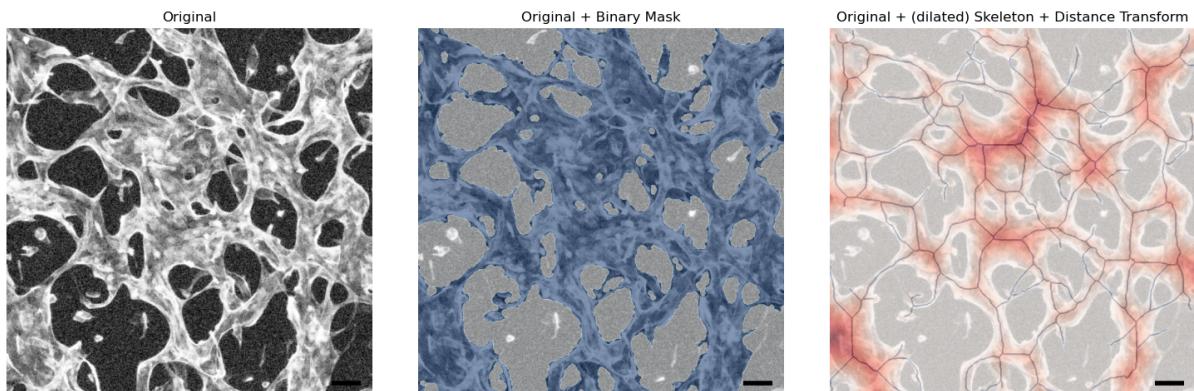
Graphs are a data structure composed of nodes and edges. Within the Polacheck networks, nodes are the locations of vessel branches and termini (ends). Edges represent the connections between the nodes and are weighted by parameters such as segment length, radius, volume, surface area, contraction factor, and fractal dimension. Representing the Polacheck networks as graphs is motivated by the need for empirical measures which can be used to compare separate networks or a single network over time.

The graph representations in `mvn-analysis` utilizes the NetworkX package. Graphs in the NetworkX data format have immediate compatibility with a large library of graph analysis algorithms. Therefore, generating NetworkX graphs allows for simple extendability

I really like
the graphical
output here



(a) Sample A: Scale 100 μm



(b) Sample B: Scale 100 μm

Figure 2.2 Major outputs from the 2D Segmentation step. Only the largest connected regions have been retained. *Left:* Enhanced contrast. *Middle:* Binary Mask. *Right:* Distance Transform and Skeleton.

of data analysis even after the graphs are constructed [27].

Identifying Ends and Branches

Branch and end points are determined for the network graph by analyzing the skeleton produced in the 2D segmentation step. Specifically, the skeletonization is represented as a binary array of 0s and 1s. For every pixel of value 1, the sum of the surrounding 3×3 pixel area is taken (excluding the center pixel). If the sum equals 1, then the center pixel must be an end point. If the sum is 2, the pixel is a point along a vessel path and is ignored. Else, the sum is greater than 3 and the pixel must be a branch point.

Generating the Weighted Graph

With the ends and branch locations known, it is possible to generate the graph nodes and edges. First, every end and branch location is represented as a NetworkX node with its euclidean coordinate as an accessible attribute. Then, a copy of the skeleton is made with the node locations removed (set to zero). Starting at each branch location, a recursive algorithm steps along all paths in the skeleton. For each step, the SkeletonWalker algorithm adds the distance traveled (the global unit times 1 or $\sqrt{2}$ for a straight or diagonal step), notes the vessel radius (the global unit time the value of the distance transform at the current location), and removes the pixel from the copied skeleton.

SkeletonWalkers which reach pixels that have no neighbors are adjacent to, or at, the location of a node. The identity of this node can be determined by checking the ends and branches sets for existence of the coordinates in the immediate 3×3 neighborhood of the the SkeletonWalker. Once determined, a NetworkX edge can be created between the origin node of the SkeletonWalker and the node it reached. The edge is weighed with the parameters the SkeletonWalker accumulated throughout its journey, such as total segment length and the estimated segment volume and surface area. The process described can be seen as pseudo code in Procedure 1. Remember, Procedure 1 would be instantiated for every branch node

in the network.

Procedure 1 SkeletonWalker

Input: skeleton, distTransform, originNode, totLength, totVolume, totSurfaceArea

Input: The first call of SkeletonWalker from any origin node sets all parameter totals to 0

erodedSkeleton = skeleton – current pixel location

neighbors = non zero elements in surrounding 3x3 area

for all neighbors **do**

move to neighbor

d = distance to neighbor

r = value of distTransform at current location

totLength = totalLength + d

totVolume = $d * \pi r^2$

totSurfaceArea = $d * 2\pi r$

Call SkeltonWalker with updated location and weights

end for

if there are no neighbors **then**

Determine what destination node has been reached by searching ends and branches

Create NetworkX edge between originNode and currently reached node with
the current SkeletonWalker weights

end if

Cleaning the Graph

The generated NetworkX graphs are cleaned based on user parameters. For example, nodes are given a minimum euclidean separation and edges have a minimum length. For nodes within the minimum separation, a new node is created at their midpoint. The new node is passed the edges of the old nodes and adjusts the weights to account for the new position. Edges that do not meet the minimum length are simply removed. Additionally, edges with

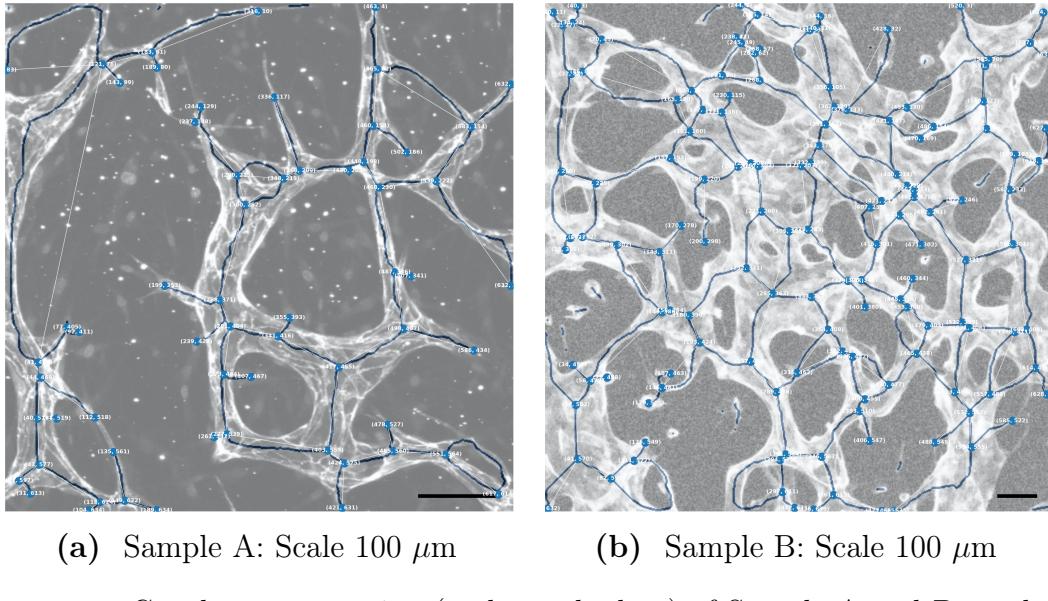


Figure 2.3 Graph representation (nodes and edges) of Sample A and B overlaid on their respective flattened images. The blue lines show the skeleton used to generate the graph and the white lines show the graph edges. The original image can be found on GitHub if there is difficulty viewing the graph edges.

positive length but zero displacement (self loops) are also removed. Like the image masks, the user can decide to retain only the largest connected network. Additionally, the user can allow disconnected graphs, but specify a minimum number of nodes for a viable network, thereby removing artifacts.

The nodes and edges of the graphs created for Samples A and B are shown in 2.3. For visual validation, the nodes are overlayed on the flattened sample image at their stored euclidean coordinates. If the graph generation process was successful, the node locations should correspond with the branches and ends observed in the image.

Numerical Analyses

In addition to setting the nodes and edges, constructing the network graphs generates a set of numerical characteristics for each sample. Specifically, each vessel segment has a length, volume, surface area, average radius, contraction factor, and fractal (Hausdorff) dimension,

Table 2.1 Graph summary values generated for Samples A and B.

Graph Characteristic	Sample A	Sample B
Number of branch points	34	103
Number of end points	27	33
Total length μm	4783.2	18804.1
Total surface area μm^2 (assumes circular vessels)	477455.3	4450488.0
Total volume μm^3 (assumes circular vessels)	5057376.9	122078184.2
Average branch length μm	75.9	108.7
Average branch surface area μm^2	7578.7	25725.4
Average branch volume μm^3	80275.8	705654.2
Average branch radius μm	17.3	45.0
Average fractal dimension	1.023	1.0213
Average contraction factor	0.9106	0.9144
Average node connectivity	1.1949	1.6339

where

$$\text{contraction factor} = \frac{\text{displacement}}{\text{length}}, \quad (2.1)$$

and [28],

$$\text{fractal dimension} = \frac{\ln(\text{length})}{\ln(\text{displacement})} \quad (2.2)$$

These segment values are combined (summed or averaged) to provide network-wide characteristics. The characteristics for Samples A and B are summarized in Table 2.1. Note, "node connectivity" is defined later in **Graph Analyses**.

In addition to the network-wide values shown, the distribution of individual segment values can be compared between networks. This process is automated within `mvn-analysis`

Table 2.2 Directionality scores for Samples A and B.

Direction	Sample A	Sample A	Sample B	Sample B
	Unweighted	Weighted	Unweighted	Weighted
Percent y-axis	0.3188	0.3334	0.2716	0.2701
Percent x-axis	0.2514	0.2421	0.2067	0.2025
Percent $y=-x$ axis	0.234	0.2052	0.269	0.2873
Percent $y=x$ axis	0.1958	0.2192	0.2527	0.2400
Z-Score of max:	1.5452	1.6708	0.8296	1.1587

and explored in Chapter 4.

Directionality

Frequently, members of the Polacheck lab have postulated that certain MVNs—or certain sections of MVNs—have their vessels oriented in a similar direction. However, this "directionality" characteristic is too qualitative for robust comparisons between MVNs.

Therefore, the SkeletonWalker code is leveraged to create an empirical measure of directionality. At each location, the SkeletonWalker can only move in one of four directions: along the x axis, the y axis, the $y = x$ axis, and the $y = -x$ axis. During the edge creation process, the number of moves along each axis is recorded (moves along the x and y axes increase their counts by 1, moves along the diagonals increase their counts by $\sqrt{2}$). Thus, at the end of the edge formation, the graph outputs the directionality percentages for each axis. A second version of this process weights each move based on the the current radius of the vessel. Outputs of these calculations are shown in Table 2.2. Additionally, local directionality can be displayed as a series of ellipses that have their major and minor axes scaled to the ratios between the directionality axes. See Figure 2.4 for an example.

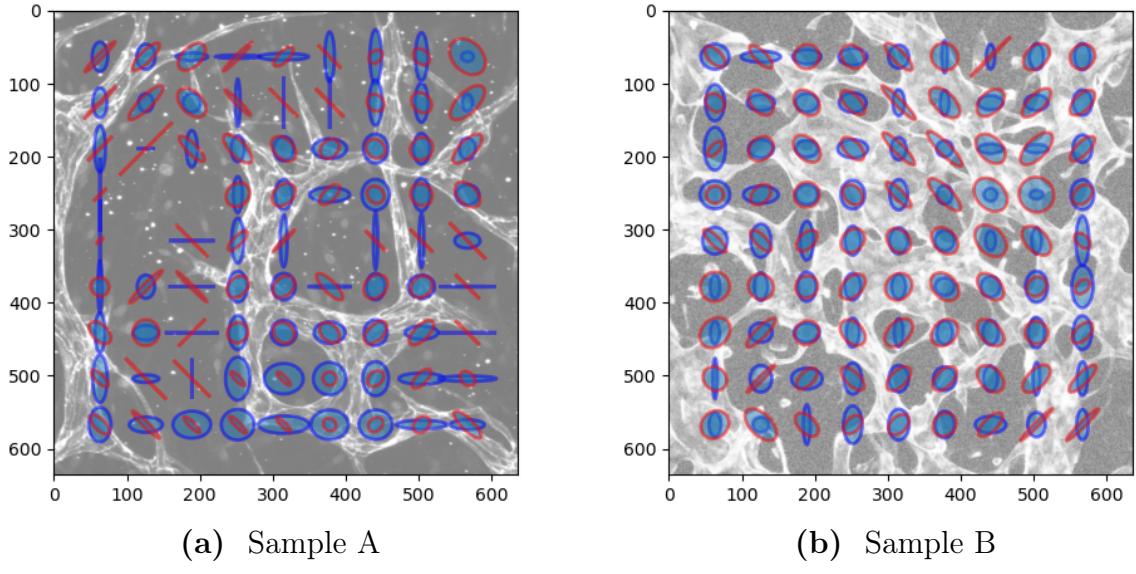


Figure 2.4 Local directionality. Blue ellipses show x , y distribution while red ellipses show $y=x$, $y=-x$ distribution.

Graph Analyses

Generating the graph using NetworkX provides access to a large library of graph analysis algorithms. Currently the only algorithm used by `mvn-analysis` is the `allPairsNodeConnectivity` algorithm. For every combination of nodes in the graph, the `allPairsNodeConnectivity` algorithm determines the fewest number of edges that would need to be cut to completely disconnect the two nodes. Similar to the characteristics described in the Numerical Analyses section, the graph connectivity is another way to compare MVNs. However, once flow tests begin, connectivity may serve as a predictor for network pruning. For end-user data exploration, `mvn-analysis` save the network object such that it can be opened in a separate IPython environment.

2.3.3 2.5D Surface Mesh

For this report, a 2.5D object is a three dimension object that is generated only from a two dimensional representation. Specifically, the euclidean distance transform generated in the

2D segmentation step can be used to create a 2.5D surface and volume mesh representation of the lumens.

Imposing the Assumption of Round, Symmetric Vessels

The euclidean distance transform is a representation of the the lumen binary mask where pixel intensities equal the shortest euclidean distance to a non-lumen area. Further, the skeletonizaiton of the lumen binary mask represents the vessel center-lines. Therefore, the product of the distance transform and skeleton yields a skeletonization with non-zero elements equal to the value of the distance transform at the same location.

If vessel cross sections are circular and the MVNs are completely planar (all vessel center-lines exist on the xy plane), then the weighted skeleton would describe the local radius of the vessels. Thus, an accurate, three dimensional representation of the network could be created by projecting the radii into three dimensions and creating circular tube surfaces.

Obviously, the MVNs do not have perfectly circular cross sections nor are they completely planar. However, enforcing these conditions allows us to follow similar logic and robustly generate surface and volume meshes that share characteristics with the real, three dimensional geometries.

Projecting Into 3-Space

To produce a 2.5D volume, we fill voxels above and below the plane of the distance transform according their intensity. For example, if a pixel has intensity 5, then the 5 voxels above and 5 voxels below the xy plane are filled. However, doing this procedure directly on the euclidean distance transform would create diamond lumen cross sections (the values the distance transform are linear with respect to the perpendicular distance from the vessel centerline). Therefore, prior to the voxel filling procedure, the distance transform is scaled to have a quadratic relationship with respect to perpendicular distance from the vessel centerline.

At each location along the skeleton, the $\text{distVal} \times \text{distVal}$ neighborhood of the distance transform is sampled (where distVal is the distance transform value at the current location). Each pixel within the neighborhood is then assigned a scaled value according to,

$$\text{scaled} = \sqrt{(\max - \min)^2 - (\max - \text{original})^2} + \min \quad (2.3)$$

where "original" represents the original pixel value and "max" and "min" are the maximum and minimum values in the neighborhood. Once every skeleton location has been sampled, the final transformed image is represented by the average of each pixel's scaled values. The procedure has the effect of converting the euclidean distance transform to a quadratic transform and smoothing the result. From there, voxel filling can proceed.

Extracting Surface Representation

The marching cubes algorithm is used to extract the 2D surface from the 3D binary image. PyMesh is used clean and convert the extracted triangle representation to an OBJ file.

Z-Scaling and Examples

The described procedure will create circular lumen cross sections. However, general ellipsoidal cross sections can be obtained by pre-scaling the distance transform prior to smoothing. For example, by multiplying the distance transform by 0.5, the resulting lumen cross sections will have xy axes twice the magnitude of their z-axes. When selected by a user, `mvn-analysis` will automatically choose this scaling parameter to force the 2.5D volume to have z-depth equal to the z-depth of the sample (based on image metadata). Figure 2.5 shows the 2.5D representation of Sample A using both circular lumens and automated ellipsoidal lumens.

As can be seen, enforcing proper z-depth decreases the height of the lumen cross sections. We assume for sample A, this re-scaling more correctly fits the physical network as the circular lumens have a z-depth greater than the total image acquisition depth. However,

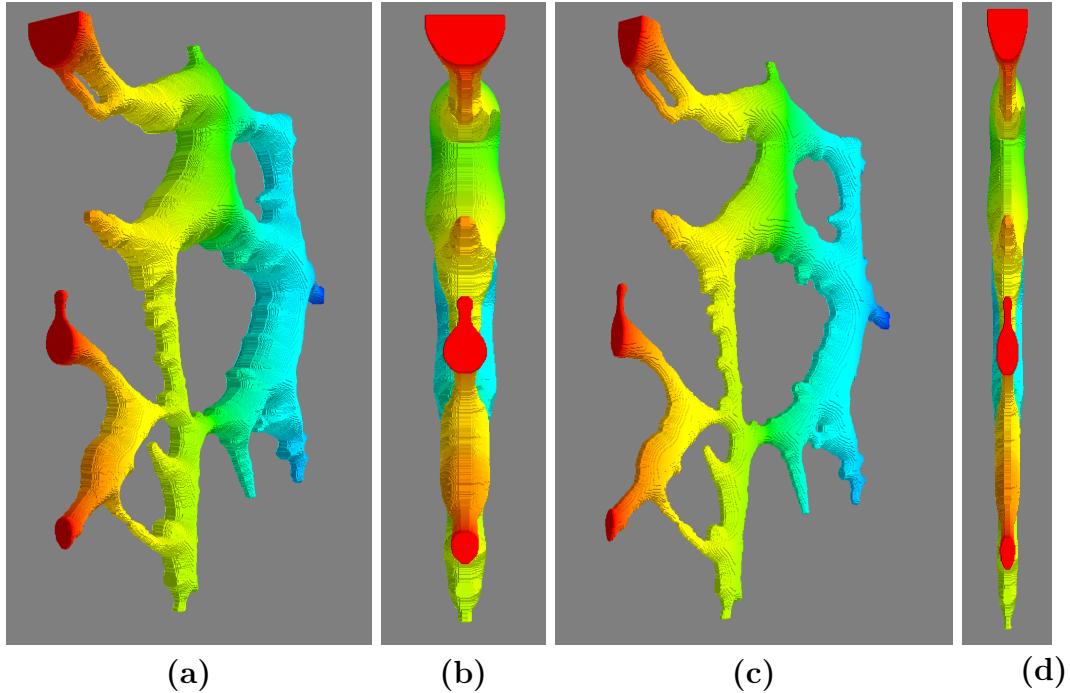


Figure 2.5 Comparison between enforced circular lumens (a, b) and enforced z-depth (c, d) for a highly planar network (Sample A).

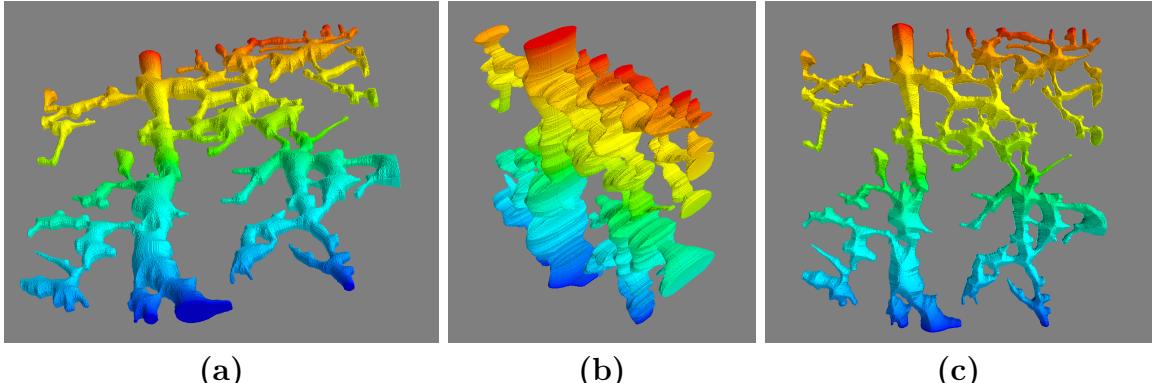


Figure 2.6 Comparison between enforced circular lumens (a) and enforced z-depth (b, c) a highly three dimensional network.

using the automated z-scaling is not always beneficial. For example, Figure 2.6 shows 2.5D representations of a more three dimensional MVN.

As can be seen, the automated z-scaling generates unrealistic lumen cross sections. This occurs because the sample is highly three dimensional, and therefore image depth is a poor proxy for vessel thickness. Finally, Figure 2.6 (c) shows the z-fitted 2.5D mesh looking directly along the z-axis. From this view, the mesh looks identical to the circular lumen mesh, demonstrating that only the z-components are altered during these procedures.

Uses

As shown in the comparisons of Figure 2.5 and 2.6, the degree to which the 2.5D mesh represents the MVN is dependent on individual MVN characteristics. Beyond losing information about vessel z-location, the 2.5D mesh representations will have connections between vessels that appear to cross but exist on different z planes. Therefore, the procedure is not high fidelity. Nevertheless, the generation of 2.5D meshes is robust and lightweight. The meshes generated are guaranteed to be water tight, smooth, and provide flat surfaces at inflow-outflow locations. Therefore, they are easily adapted to fluid flow models.

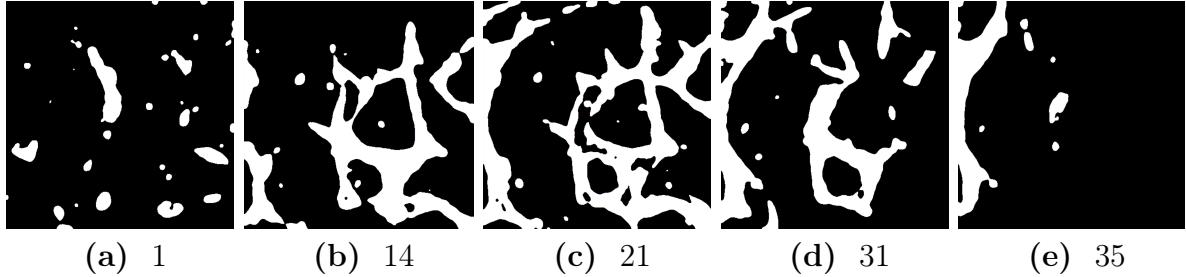


Figure 2.7 Examples of the binary images at slices ($n/39$) through the Sample A 3D reconstruction

2.3.4 3D Segmentation and Surface Mesh

Because the MVNs grow in three dimensions, a three dimensional segmentation is required to fully capture the structural data. The following sections discuss methods developed in an attempt to overcome the challenges of 3D segmentation described in Section 1.2.2.

Thresholding Image Planes

The 3D segmentation process takes in the original .tif image represented as a stack of NumPy arrays. Each array has its contrast enhanced and noise removed. Then the entire stack (represented as a single 3D image) has threshold values calculated using Sauvola local thresholding. Somewhat uniquely, the calculated thresholds are not used to convert the image to binary, but are rather used as the input images to Li threshold (which ultimately ends in conversion to binary). Morphological opening and closing are performed, however unlike the 2D segmentation, disconnected regions are always retained.

At this point, the MVN is represented as a stack of binary images. Figure 2.7 shows examples of the binary slices produced in this step. As predicted from Section 1.2.2, the areas of lumen are not full and are often not connected. Therefore, the following sections are focused on filling these errors.

Lumen Filling via the Ellipsoid Octant Method

The first procedure we developed to fill the lumen holes is referred to as the ellipsoid octant method. The motivation behind the method is the assumption that dark voxels that have a high ratio of light voxels in a given neighborhood are more likely to be part of the lumen. Therefore, the ellipsoid octant method creates a three dimational binary mask that represents an ellipsoid of a given xy dimension and z height. The ellipsoid is then broken into 8 equal sections (hence the octant) and centered on dark voxels within the 3D binary array that represent the current MVN segmentation.

Each octant then sums the product of itself and the segmentation, and divides by its own sum. This yields the percentage of the voxels spanned by the octant that are bright. A thershould percentage is provided to the method, and an octant is considered "on" if it reaches the threshold.

Additionally, the method is provided an acceptance rule. If the *total ellipsoid* acceptance rule is used, the dark voxel being investigated is set to bright (included in the lumen) if the total percentage of bright neighbors for the whole ellipsoid is greater than the threshold. If instead, the *total octant count* acceptance rule is used, the dark voxel is set to bright if the number of "on" octants is greater than a given threshold. Finally, if the *opposite pairs count* condition is used, the voxel is set to bright if the total number of pairs of octants that are spatially furthest from each other and both "on" is greater than a threshold. The different acceptance methods—along with their respective thresholds—are designed to provided various levels of rigger to the filter.

Finally, the ellipsoid octanct method uses the 2D segmentation mask as a prior. Specially, the method only tests voxels that are dark in the 3D segmentation, but bright in the 2D mask. As the 2D segmentation captures the greatest extent of all vessels, any area not included in the 2D mask also should not be included in the 3D segmentation. This qualification both increases the speed of the algorithm (by reducing the search space) and prevents the method

from filling areas that are outside the lumen densely vascularized.

The method starts with an ellipsoid of a given size and iterates over the search space. Then the ellipsoid size is decreased and the processes is repeated until the ellipsoid is only searching adjacent voxels. The main goal of the ellipsoid octant method is to close gaps in the vessels walls.

Lumen Filling via the Ray Tracing Method

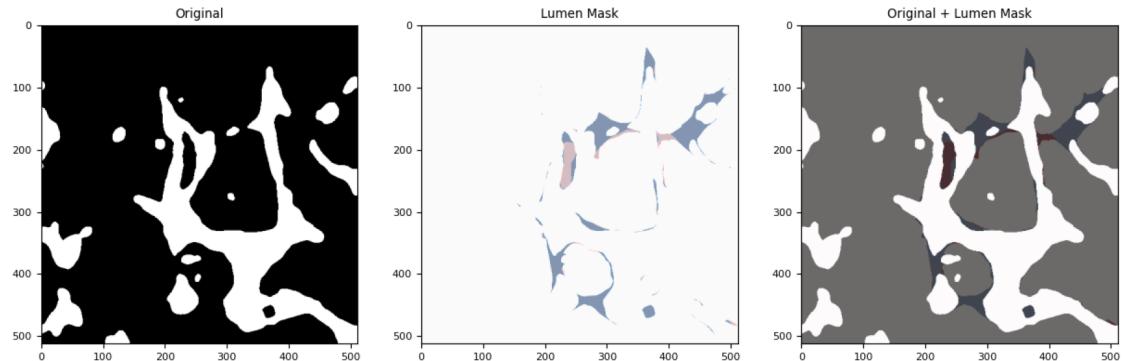
The ellipsoid octant method returns a new binary array representation of the MVN segmentation. Ideally, the gaps in the vessel walls will have been sealed. However, large holes in the lumen usually persist. Therefore, a ray tracing method was developed to fill remaining lumen holes.

Once again starting from dark voxels that are bright in the 2D segmentation, the ray tracing method generates three dimensional trajectories (the traditional ϕ and θ angles of spherical coordinates). The method then propagates the rays thorough the binary image and notes the values of the voxels encountered. Rays that originate in the lumen should never reach the array edges without passing through a bright pixel.

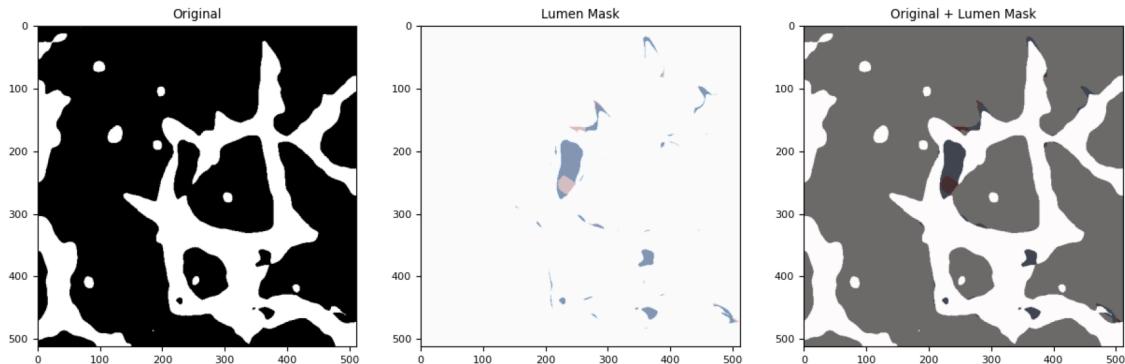
Therefore, the ray tracing method calculates the percentage of rays which reach the array edge. This percentage can then be used as an acceptance criteria for the voxel. The total number of rays as well as their distribution (uniform, random, or xy planar) can also be controlled. The ray tracing method has the advantage of being distant tolerant. It does not matter how large the lumen hole is, only that it adequately sealed. Figure 2.8 demonstrates the effects of the ellipsoid octant method and ray tracing method.

Lumen Filling via Iterative Meshing

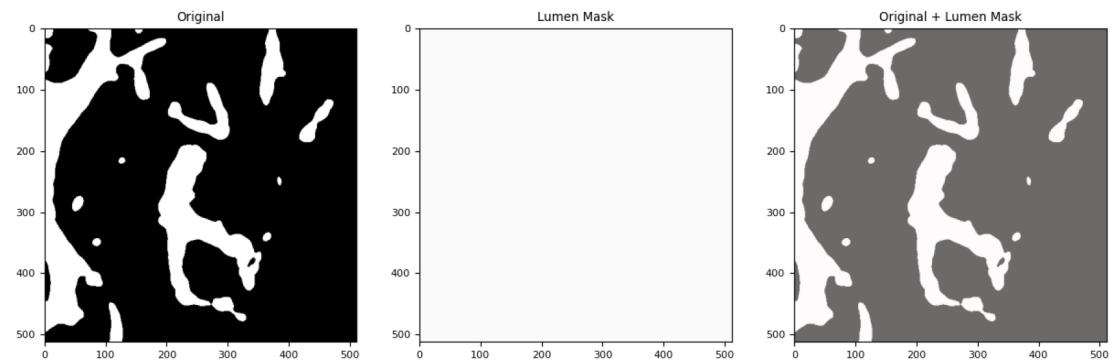
Examining Figure 2.8 shows (a, b) to successfully fill the holes in the lumen. However, a closer examination of (c) shows holes can still persist (look near $x=350$, $y=375$). Therefore, one final method is used to fill the lumens.



(a)



(b)



(c)

Figure 2.8 Performance of the lumen filling methods on Sample A. The blue represent the filling achieved by the ellipsoidal octant method and the red represents the filling achieved by ray tracing.

The marching cubes algorithm was mentioned previously in reference to extracting a 2D surface from a 3D volume image. Intuitively, these surfaces occur where regions of white meet regions of black. Therefore, holes in the lumen can be represented as surfaces within the lumen.

To take advantage of this representation, the 3D binary image is first separated into connected regions and the largest section is retained. Then, this connected image is converted to surface meshes using the marching cubes algorithm. The largest output mesh (measured by number vertices) is assumed to represent the exterior lumen surface. Therefore, all other meshes represent holes within the lumen.

Bounding boxes are generated for the sub-surfaces. Voxels of the 3D binary array that are within the bounding box coordinates are identified as holes and have their dark values set to light values. The update image is then re-meshed and re-examined for holes. The process continues until meshing the 3D binary image returns a single surface. The resulting binary array represents the largest connected region of lumen.

Scaling and Interpolating

The final major step in segmenting the lumen involves interpolating between z-planes. The MVN images are taken in discrete z-steps that are often up to $6\times$ the size of the distance between pixels.

Within `mvn-analysis` two binary image planes are interpolated by first representing them as signed euclidean distance transforms (the euclidean distance transform of the image minus the euclidean distance transform of the inverted image). Then, the weighted average of the two signed transforms is taken (weighted on how close the interpolation should be to each plane). Finally, the interpolated plane is taken to be the values greater than 0 in the weighted average.

Interpolated planes are added to the 3D binary MVN segmentation such that each z-step represents the global unit. Then, each plane in the image is scaled in the x and y axes such

that their respective steps also equal the global unit. The final result is a physiologically binary representation of the largest connected lumen.

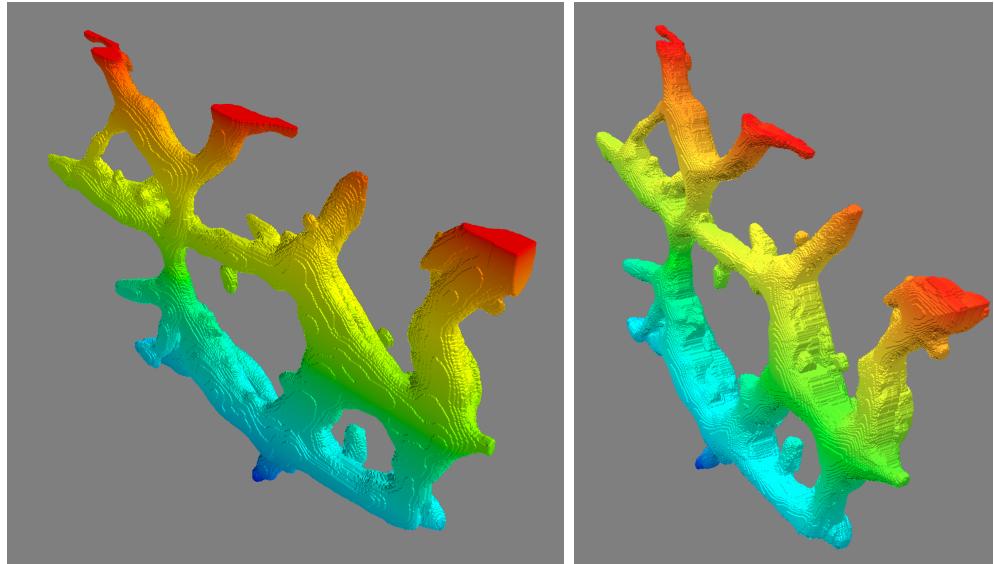
3D Surface Meshes

Using the marching cubes algorithm on this binary array yields the "honestly scaled" mesh mentioned in Section 2.1.

However, while the procedures used to fill lumen holes are relatively robust, the 3D segmentation still struggles to accurately capture the entire lumen. In particular, the upper and lower sections of the vessels are often missing (leading to flat surfaces on the top and bottom of the honest mesh). This occurs for two reasons. First, the upper and lower sections of the lumens are represented by thin lines when observed as z-slice. Therefore, they are often excluded in the threshold (as their inclusion would also bring in artefacts). Second, as mentioned, large z-stepping while imaging can simply skip these areas.

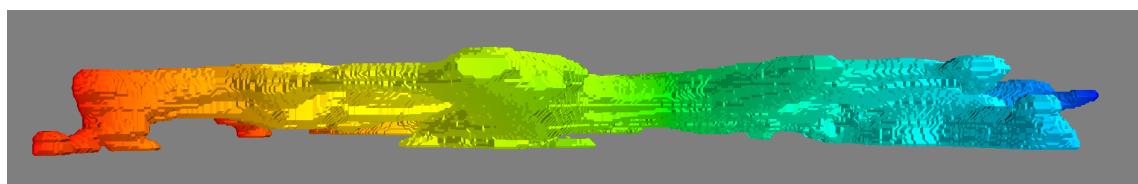
To attempt to reconstruct these missing areas, the process used to generate the 2.5D meshes is applied to each plane in the binary array, thus enforcing curved vessels where xy planes existed. Pre-scaling the distance transforms—as discussed in Section 2.33—is used to mitigate the effect of the smoothing.

Figure 2.9 demonstrates the effect of the filter. As can be seen, the rounding filter reestablishes curvature in the top and bottom planes of the network. Further, Figure 2.10 shows the 3D segmentation is able to have vessels cross in separate z-planes without being joined. Finally, Figure 2.11 shows the 3D mesh of Sample B compared to its 2.5D representation. Taken together, these figures show how segmenting in three dimensions has large impact on vessel connectivity.

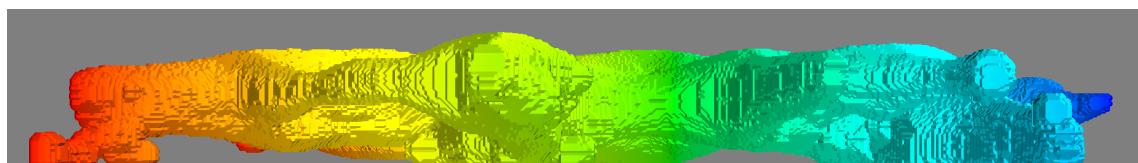


(a)

(b)



(c)



(d)

Figure 2.9 "Honest" scaling 3D mesh (a, c) and "rounded" scaling 3D mesh (b, d) for Sample A. For the rounded scaling, 0.5 prefactor was applied to the distance transforms.

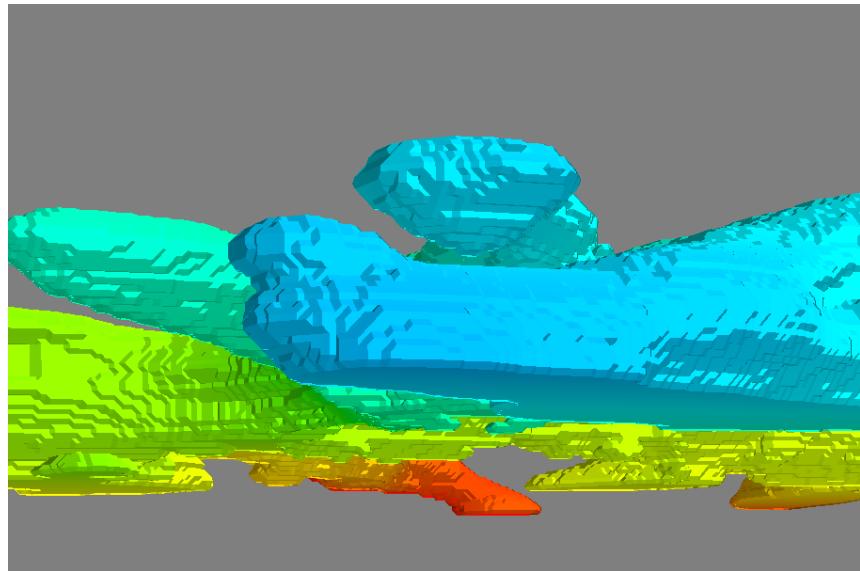


Figure 2.10 Demonstration of vessels crossing in different z-planes. These are the vessels shown are connected in the 2D and 2.5D representations of the network.

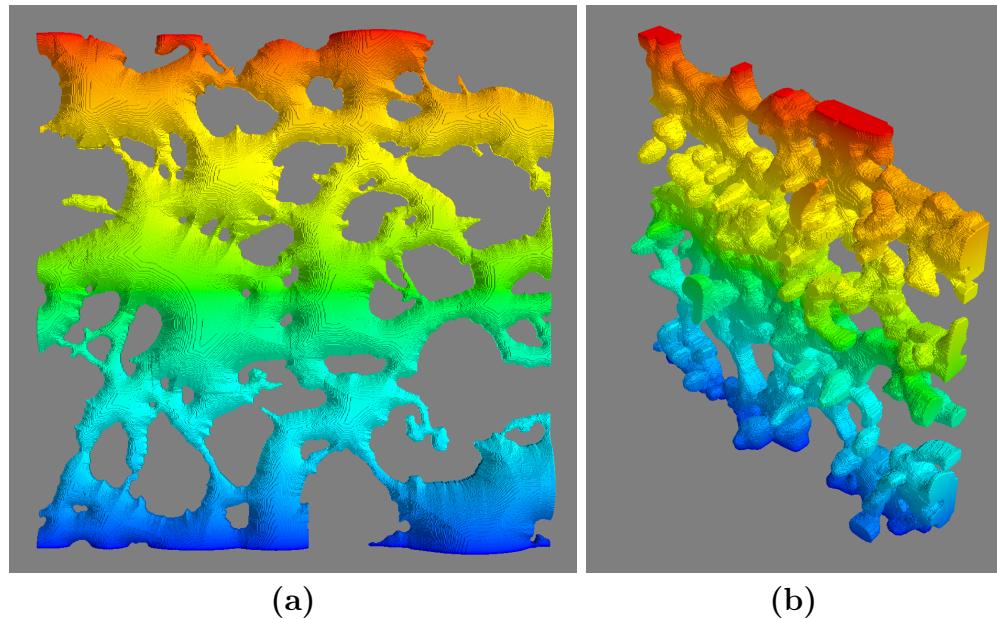


Figure 2.11 2.5D (a) and 3D (b) segmentations for Sample B.

2.3.5 Surface Meshes to Volume Meshes

The surfaces generated in the following sections are save as Wavefront (OBJ) triangle mesh. However, the Stokes flow solver used later in the analysis requires volume meshes. For this, the TetWild software package is used to convert the OBJ to a tetrahedral volume MSH file [29].

2.3.6 Numerical Outputs

Within `mvn-analysis`, three dimensional binary arrays of the lumens are constructed in three separate ways: 2.5D transformation, honest scaling 3D segmentation, and rounded 3D segmentation. During each of these processes, the 3D binary array is used to calculate volumetric characteristics of the network. The outputs of this data is shown in Tables 2.3 and 2.4

2.4 Performance Notes

2.4.1 Speed

Many operations within `mvn-analysis` require iterating over large, three dimensional arrays. Therefore, whenever possible, these processes utilize the parallel Numba just-in-time compiler to pre-compile Python loops to machine code and execute the process in parallel. Additionally, information gathered from the 2D segmentation process (fast) is used to optimize the 3D segmentation process. For example, the algorithms that fill internal lumen holes do not search all space, but rather only search the space that is spanned by the 2D image mask (as the 2D lumen will represent the extreme values of the lumen). Nevertheless, an obvious speed increase could come from transferring the project from Python to C++.

Table 2.3 Volumetric data generated for Sample A.

Source	25D	Honest 3D	Rounded 3D prefactor-0.5
Connected	True	True	True
Bounding Box (z, x, y) (μm)	(43, 635, 446)	(40, 635, 469)	(69, 635, 469)
Total volume μm^3	2003356	2688945	3391810
Vascular density $(\frac{VesselVolume}{TotalVolume})$	0.1645	0.2257	0.1650
Avg dist from vessel wall μm	26.7	27.2	29.9
Max dist from vessel wall μm	132.6	143.2	145.3
Avg max dist from vessel wall per z-plane μm	131.6	140.8	141.6

Table 2.4 Volumetric data generated for Sample B.

Source	25D	Honest 3D	Rounded 3D prefactor-0.5
Connected	True	True	True
Bounding box (z, x, y) (μm)	(238, 1270, 1270)	(232, 1270, 1270)	(302, 1270, 1270)
Total volume μm^3	91363744	101779440	109411016
Vascular density $\left(\frac{\text{VesselVolume}}{\text{TotalVolume}} \right)$	0.2380	0.2651	0.2246
Avg dist from vessel wall μm	45.7	46.4	48.2
Max dist from vessel wall μm	183.0	216.8	230.5
Avg max dist from vessel wall per z-plane μu	162.0	166.3	172.5

2.4.2 Memory

The major memory saving parameters have already been discussed (increasing the global unit and thereby decreasing the image and mesh resolutions). However, the software could be further optimized to limit memory usage by transferring from a NumPy image representation to a SciPy sparse array representation. Even the most vessel dense image samples were majority empty space. Further, many of the operations are embarrassingly parallel. Therefore, large image sets could be broken into smaller sub samples, processed, and recombined.

Chapter Three

Stokes Flow Simulations

As discussed, the ultimate research goal is the relation of fluid flow forces to angiogenesis. Therefore, this chapter demonstrates how the outputs of `mvn-analysis` can be used to visualize the pressure, velocity, and wall shear stress distributions for a MVN segment. This demonstration uses the 2.5D lumen segmentation of Figure 2.6.

3.1 Prepossessing

At the end of the `mvn-analysis` pipeline, the tetrahedral lumen mesh is stored as an ExodusII file. In order to run fluid computations on the mesh, boundary conditions must be assigned to the surfaces of the mesh. Therefore, Trelis was used to label groups of surfaces that shared boundary conditions.

`sectionBeatIt` `BeatIt` is Dr. Simone Rossi's suite of "c++ code for heart biomechanics and more," and is available on GitHub (github.com/rossisimone/beatit). In particular, `BeatIt` proves a finite-element solution to the Stokes flow equations. Within `BeatIt`, surfaces can be prescribed pressure boundary conditions, no-slip boundary conditions, and zero-traction boundary conditions.

Figure 3.1 shows the geometry on which we will run the Stokes solver. For the example, the inflow and outflows of the parent channel were prescribed an arbitrary pressure difference.

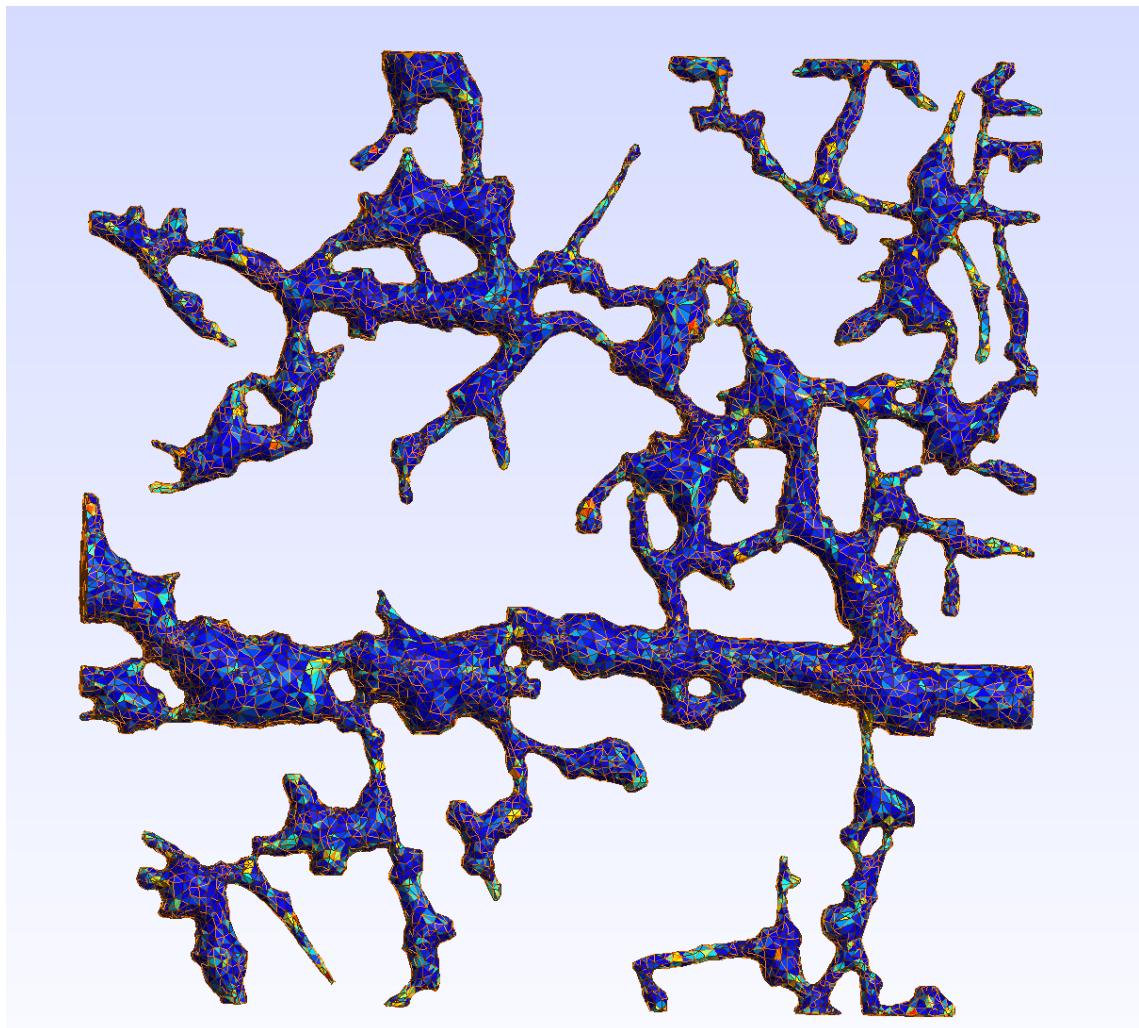
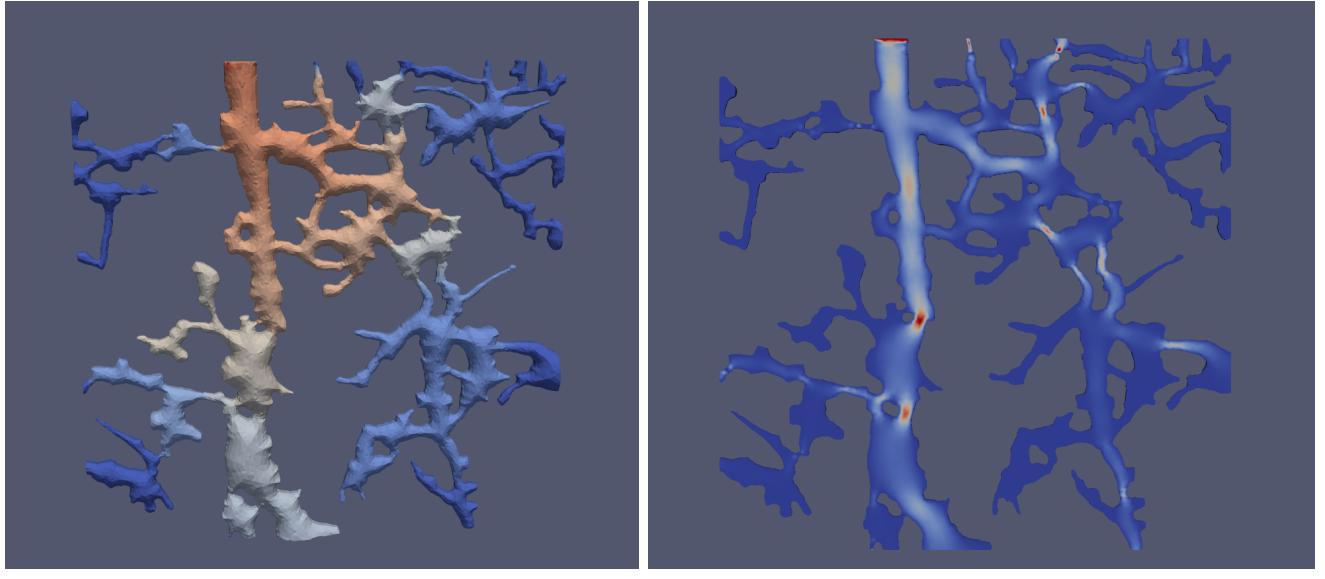


Figure 3.1 Tetrahedral mesh of the lumen used in the Stokes flow example.



(a) Pressure

(b) Velocity

Figure 3.2 Visualizations of pressure and velocity distributions for the test flow network.

All other surfaces on the xy edges of the computational domain were assigned zero-traction boundary conditions. Finally, the remaining surfaces (non inlet-outlets) were assigned non-slip boundary conditions.

Solutions to the pressure at the surface of the network and the velocity at the midpoint of the network are shown in Figure 3.2. The absolute magnitude of these outputs are not relevant as the inputs were arbitrary, however dark blue represents low magnitude and deep red represents high magnitude.

3.2 Visualizing Wall Shear Stress

To generate wall shear stress, ParaView filters were used to generate weighted average velocity vectors (i.e. velocity vectors at points weighted on the volume of their original solution element). Then, the gradient of the velocity was taken (via the Gradient of Unstructured Data ParaView filter) and applied to equation 1.4 along with the normal vectors of the ves-

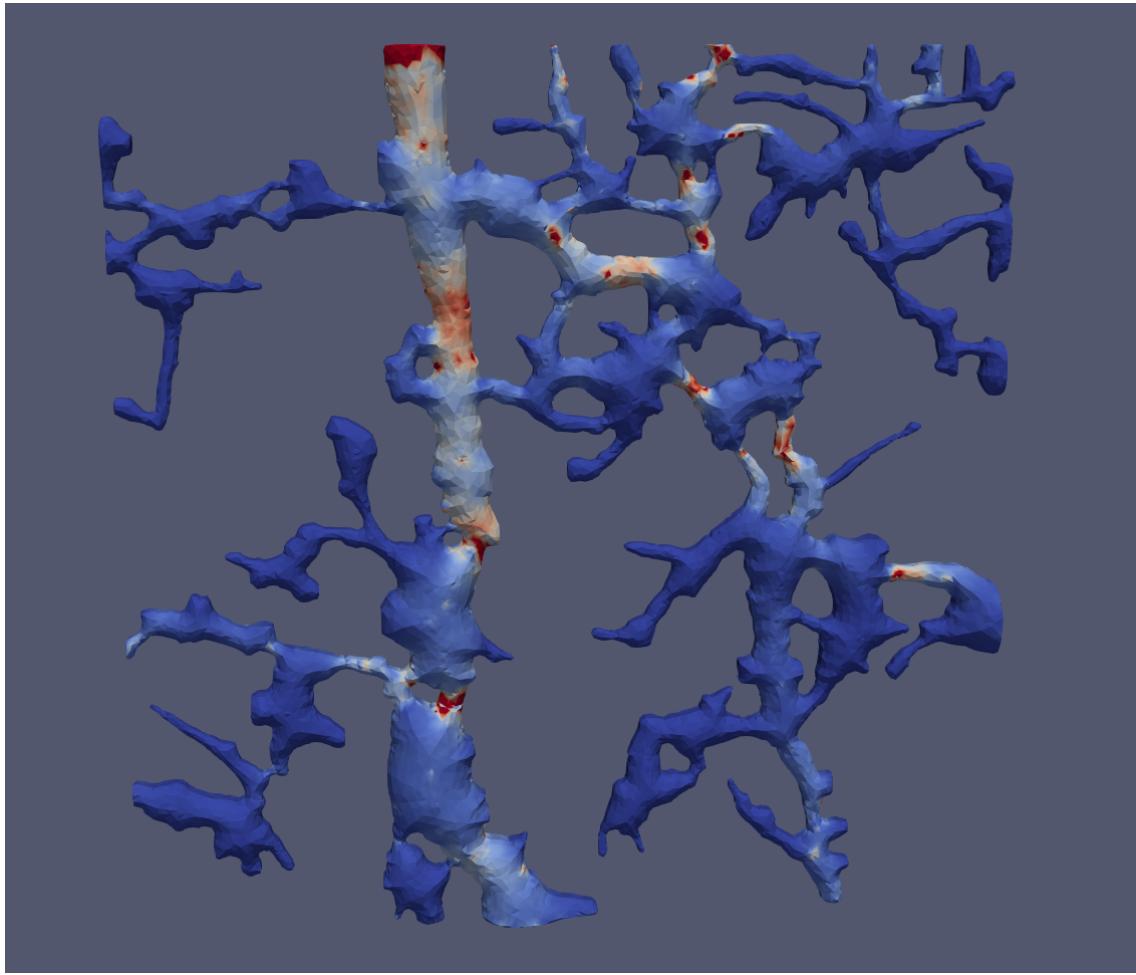


Figure 3.3 Distribution of wall shear stress for the test flow network.

sel. For the flow shown in Figure 3.2, the corresponding distribution of wall shear stress is shown in Figure 3.3

3.3 Relevance

While the wall shear stress calculations shown in Figure 3.3 are not physiologically relevant, the procedure acts as initial proof of feasibility to determine WSS within the MVNs. To truly utilize the model, pressure boundary conditions and viscosity of the experimental flows would need to be accurately determined.

Chapter Four

Preliminary Validation

The segmentations and calculations produced by `mvn-analysis` are understood to be imperfect. However, quantification of this error and proof of algorithmic validity is critical if the software is to be used in a research setting.

4.1 Cross-Method Validation

Within the segmentation and analysis, total vessel volume was calculated in four separate ways: 2.5D transform, Honest 3D segmentation, Rounded 3D segmentation, and graph generation. Comparing these values gives a high-level sanity check for consistency across the software (but not of absolute accuracy).

The graph based volumes in Table 2.1 retained smaller, disconnected networks, while the volumes in 2.3 and 2.4 were generated only from the largest connected network. Therefore, the comparison is only carried out for Sample B, as there is near negligible difference between its connected and unconnected representations. The volume data is reproduced in Table 4.1, along with the standard deviations of the set.

The standard deviation reported in Table is ~ 10 percent the value of the mean.

Table 4.1 Sample B volume calculations.

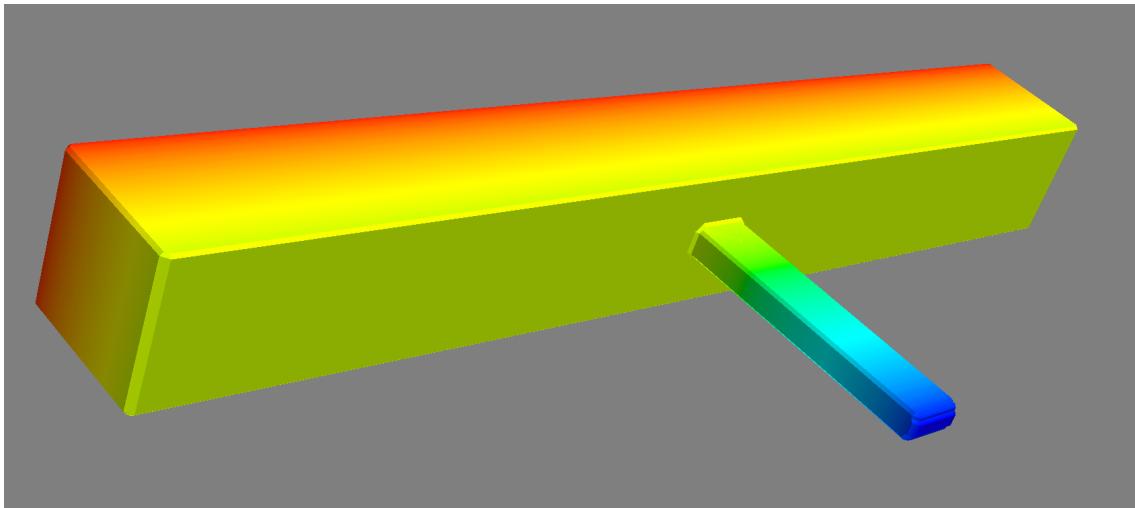
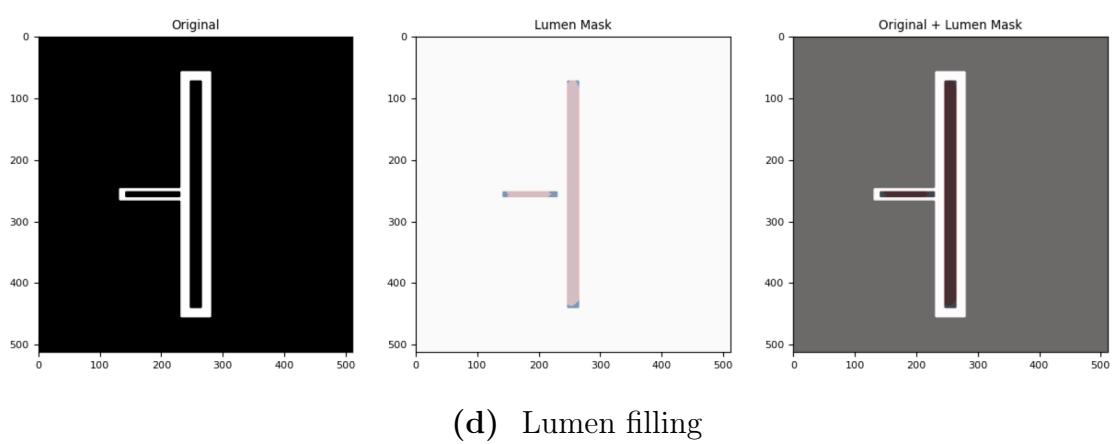
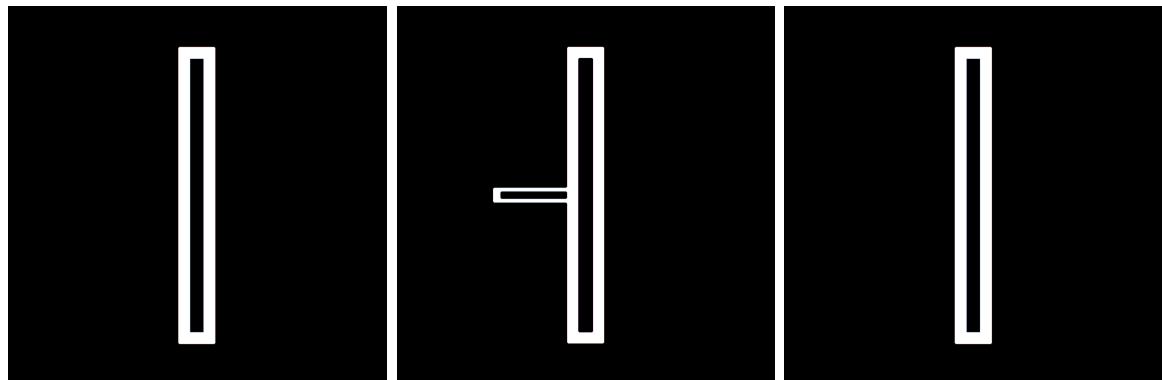
Source	2.5D	Honest 3D	Rounded 3D	Graph
Volume	91363744	101779440	109411016	122078184
Mean	106158096		Std.	11203528

4.2 T-Junction Artificial Data

In order to determine the accuracy of the segmentation process, a simple artificial data set was created. In the data set, a $20 \times 100 \times 10$ hollow box makes a T-Junction with a $400 \times 50 \times 50$ hollow box. The artificial data is provided as a `.tif` stack with z-slices at every other pixel. Therefore, to accurately return volume parameters, `mvn-analysis` must accurately fill the lumen holes and interpolate in the z-direction. The major results of the test are shown in Figure 4.1

The volume analysis returned a value of 977980 pixels³, underestimating the known value of 1020000 pixels³ by approximately 4 percent. When the analysis was re-run using the entire sample resolution (i.e., not skipping z-planes), the error was reduced to approximately 2 percent, indicating that some—but not all—of the discrepancy comes from data lost in the discrete z-steps.

While this test far from validates the software as a whole, it does support the validity of the core segmentation process.



(e) Resulting 3D Segmentation

Figure 4.1 Review of artificial data segmentation.

Chapter Five

Preliminary Data Exploration

In this chapter, 2D graphs are generated for MVNs grown under varying conditions.

5.1 Serum Depletion

The Polacheck MVNs are cultured with serum that contains co-factors required for network remodeling and growth. In order to test the effect of serum level, networks were made with 12, 25, 25, 50, and 50 percent serum depletion. These networks were then imaged and converted to 2D graphs through the process described in Section 2.3.2. To investigate the networks, the graph edge weights were compared via the histograms and distributions shown in Figures 5.1-5.7. Most of the distributions do not have strong enough trends to yield conclusions about the the serum depletion's relationship to the MVN characteristics. However, all histograms are displayed as example of the type of data produced by the graphs.

From inspection, percent serum depletion and branch radii appear to be the most strongly correlated. Plotting the average segment radius versus percentage serum depletion support this assumption and shows a positive relationship (Figure 5.8).

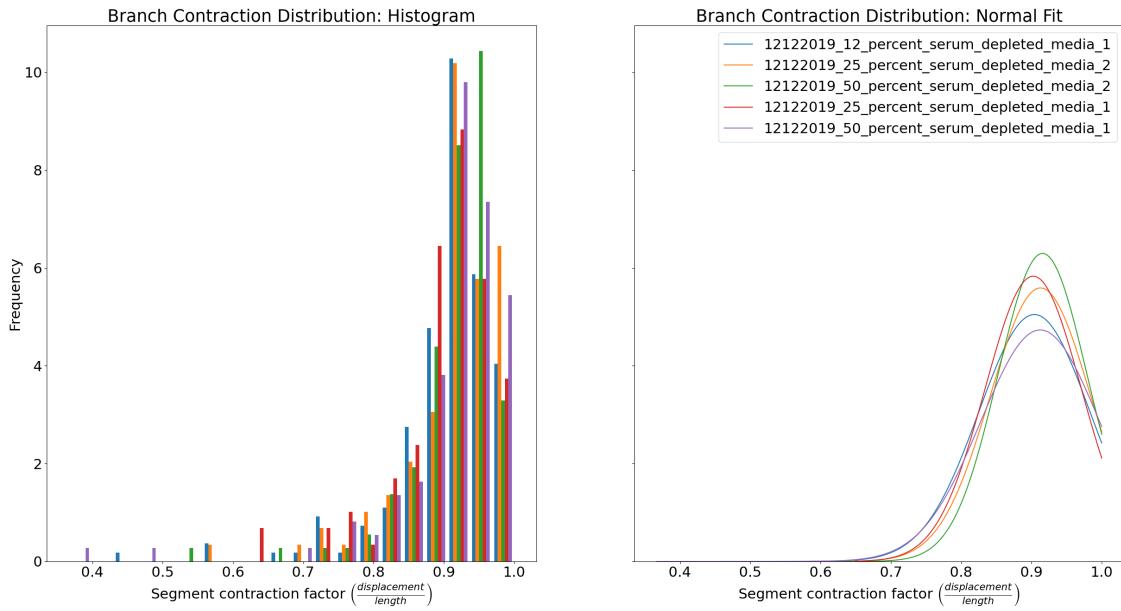


Figure 5.1 Distribution of branch contraction factors for the serum depletion experiment.

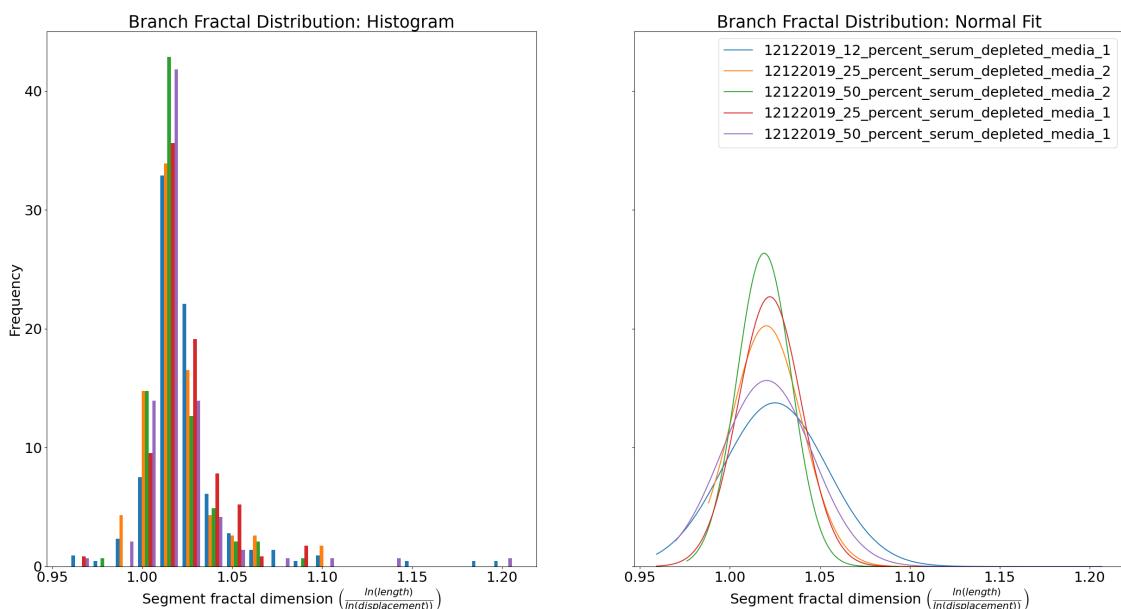


Figure 5.2 Distribution of branch fractal dimensions for the serum depletion experiment.

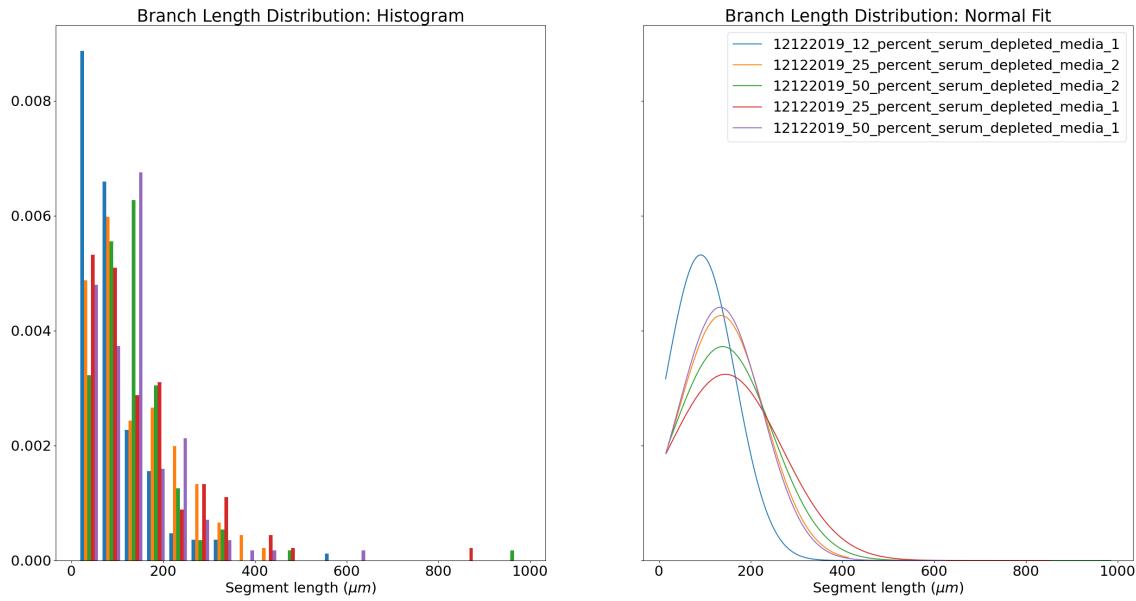


Figure 5.3 Distribution of branch lengths factors for the serum depletion experiment.

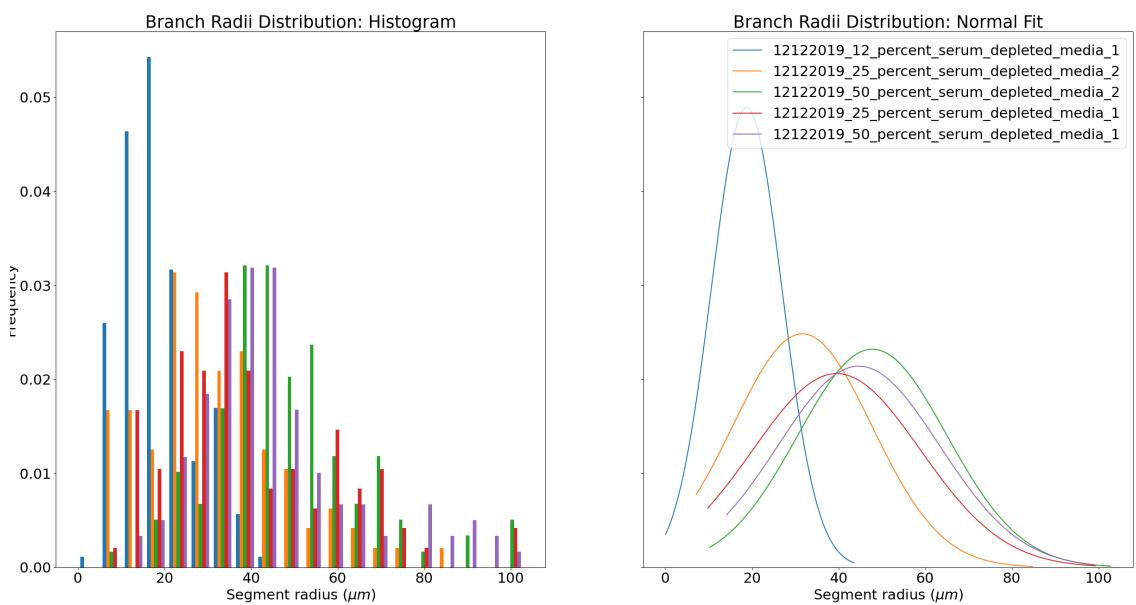


Figure 5.4 Distribution of branch radii for the serum depletion experiment.

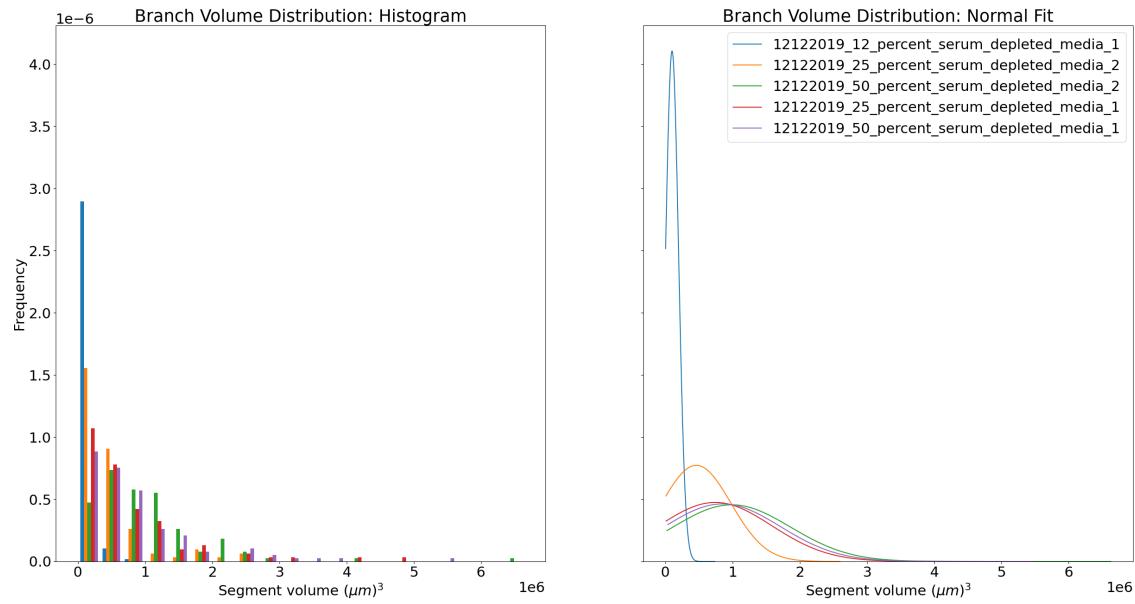


Figure 5.5 Distribution of branch volumes for the serum depletion experiment.

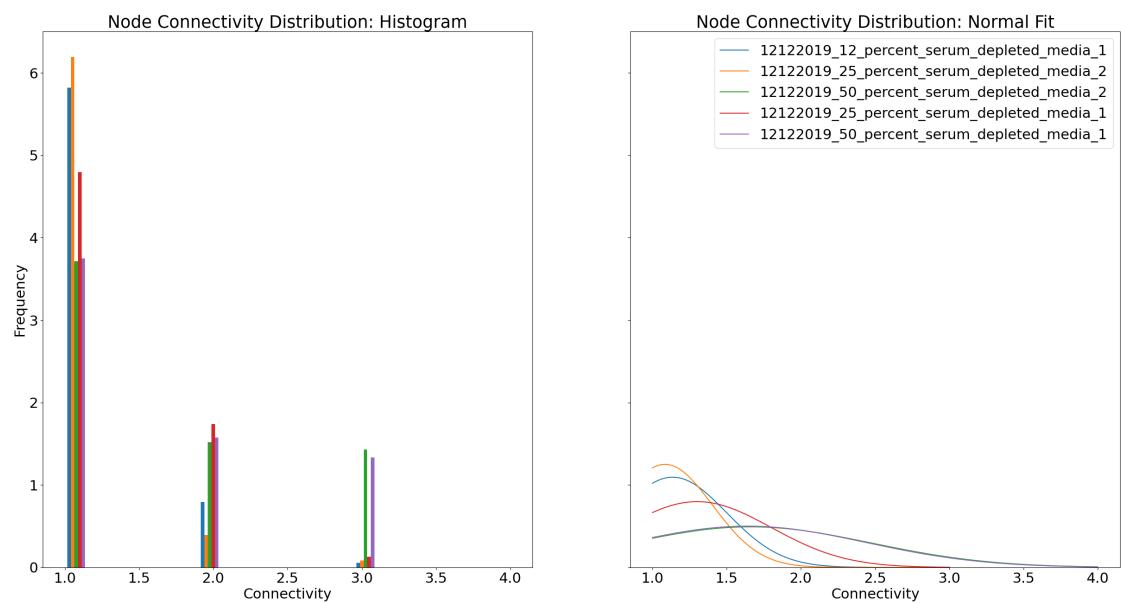


Figure 5.6 Distribution of branch connectivity for the serum depletion experiment.

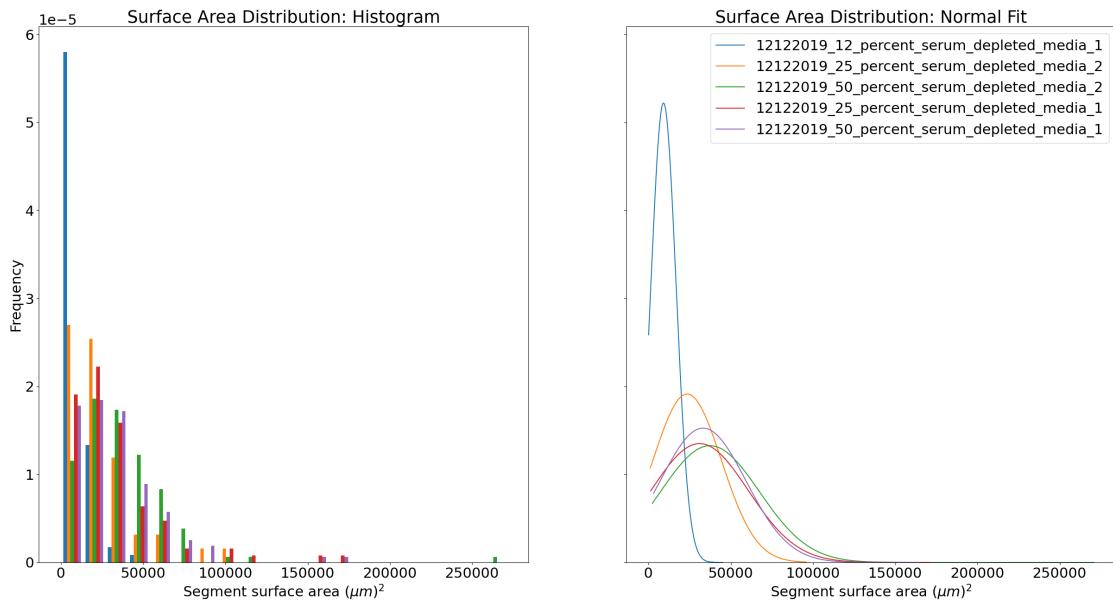


Figure 5.7 Distribution of branch surface area for the serum depletion experiment.

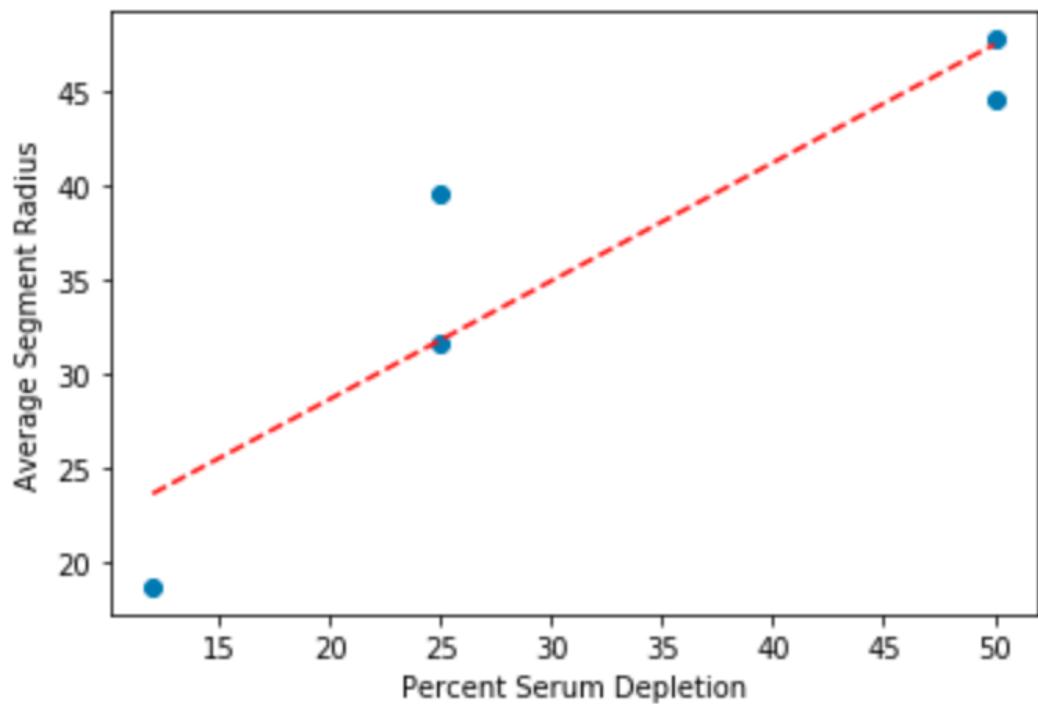


Figure 5.8 Positive trend between serum depletion and average segment radius.

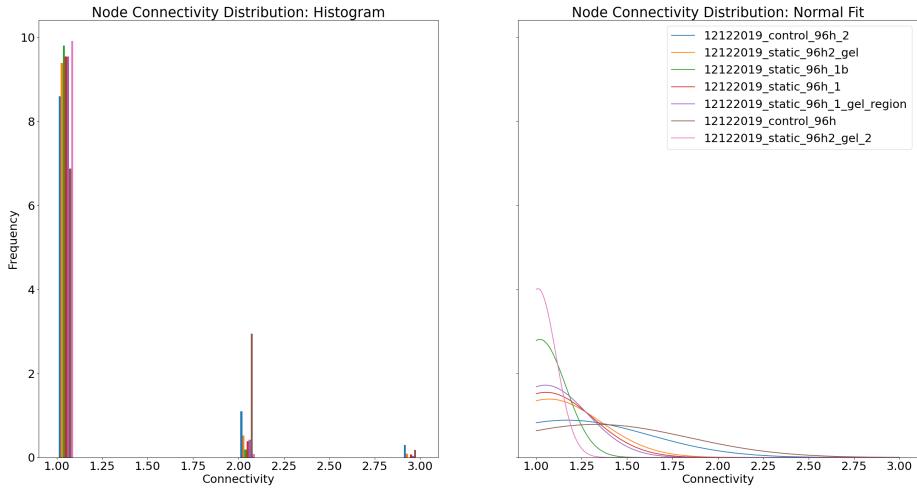


Figure 5.9 Caption

5.2 Hypoxia

The hypoxia experiment tested the change in the structure of the MVN when they were kept in static conditions (and therefore, not being as well oxygenated). Similarly to the serum depletion experiment, the graph edge weights were viewed as distributions. For this data set, the node connectivity distribution appeared the most correlated to the experimental conditions. The distribution of contraction factors is shown in Figure 5.9.

As can be seen in the distribution, the control (non-static) distribution have larger positive tails. However, this relationship does not appear as conclusive as the relationship between serum depletion and radii.

5.2.1 Summary

The data explored in this chapter provides the hypothesis that depleting serum increases average segment radii and decreasing oxygenation decreases node connectivity. Generally, higher radii and connectivity are seen in the vessel plexus rather than in fully formed networks, so this may indicate hypoxia and serum are beneficial to network development. More

data is required to strengthen these initial results. Additionally, further investigation into how the 2D graph responds to changes in dimensional (i.e., if a network becomes more three dimensional) is also needed. Alternatively, network graphs could be represented as 3D segmentations.

Chapter Six

Current and Future Work

6.1 3D Graph Representation

Representing the MVNs as NetworkX graphs is not restricted to two dimensions. In fact, a three dimensional representation is preferred as it more accurately reflects the true connectivity of the network. The 2D graph generation process can be converted to a 3D graph generation process simply by having the SkeletonWalker code search the immediate $3 \times 3 \times 3$ area. The only delay on this implementation is the creation of clean 3D skeletons to provide as inputs to the SkeletonWalker. The skeletonization process is sensitive to surface noise, which is far greater in the 3D segmentations than the 2D segmentations. I am currently working on 3D skeleton filtering procedures and should release the 3D graph generation to GitHub shortly.

6.2 Validation

All processes described in Chapter 2 require validation with artificially generated data.

6.3 Error Analysis

Beyond pure validation (obtaining results fitting expectations) I am especially interested in understanding how non-biologic factors impact the final results of calculations. For example, I would like to investigate if the difference in graph outputs when the same MVN is imaged at different magnifications and when different user parameters are used to generate the segmentations.

REFERENCES

- [1] C Alberto Figueroa, Charles A Taylor, and Alison L Marsden. “Blood Flow”. In: *Encyclopedia of Computational Mechanics Second Edition* (2017), pp. 1–31.
- [2] Alfons JHM Houben, Remy JH Martens, and Coen DA Stehouwer. “Assessing microvascular function in humans from a chronic disease perspective”. In: *Journal of the American Society of Nephrology* 28.12 (2017), pp. 3461–3472.
- [3] Mark A Skylar-Scott et al. “Biomanufacturing of organ-specific tissues with high cellular density and embedded vascular channels”. In: *Science advances* 5.9 (2019), eaaw2459.
- [4] Thomas H Adair and Jean-Pierre Montani. “Angiogenesis”. In: *Colloquium series on integrated systems physiology: from molecule to function*. Vol. 2. 1. Morgan & Claypool Life Sciences. 2010, pp. 1–84.
- [5] Y Ganat et al. “Chronic hypoxia up-regulates fibroblast growth factor ligands in the perinatal brain and induces fibroblast growth factor-responsive radial glial cells in the sub-ependymal zone”. In: *Neuroscience* 112.4 (2002), pp. 977–991.
- [6] Brian I Rini. “Vascular endothelial growth factor-targeted therapy in renal cell carcinoma: current status and future directions”. In: *Clinical Cancer Research* 13.4 (2007), pp. 1098–1106.
- [7] Peter A Galie et al. “Fluid shear stress threshold regulates angiogenic sprouting”. In: *Proceedings of the National Academy of Sciences* 111.22 (2014), pp. 7968–7973.
- [8] Siavash Ghaffari, Richard L Leask, and Elizabeth AV Jones. “Flow dynamics control the location of sprouting and direct elongation during developmental angiogenesis”. In: *Development* 142.23 (2015), pp. 4151–4157.
- [9] William J Polacheck et al. “A non-canonical Notch complex regulates adherens junctions and vascular barrier function”. In: *Nature* 552.7684 (2017), p. 258.
- [10] Guillaume Chouinard-Pelletier, Espen D Jahnsen, and Elizabeth AV Jones. “Increased shear stress inhibits angiogenesis in veins and not arteries during vascular development”. In: *Angiogenesis* 16.1 (2013), pp. 71–83.

- [11] József Timár et al. “Angiogenesis-dependent diseases and angiogenesis therapy”. In: *Pathology Oncology Research* 7.2 (2001), pp. 85–94.
- [12] William J Polacheck et al. “Microfabricated blood vessels for modeling the vascular transport barrier”. In: *Nature Protocols* 14.5 (2019), p. 1425.
- [13] Pol Kennel et al. “Toward quantitative three-dimensional microvascular networks segmentation with multiview light-sheet fluorescence microscopy”. In: *Journal of biomedical optics* 23.8 (2018), p. 086002.
- [14] Kerry J Howe et al. *Principles of water treatment*. John Wiley & Sons, 2012.
- [15] KP Ivanov, MK Kalinina, and Yu I Levkovich. “Microcirculation velocity changes under hypoxia in brain, muscles, liver, and their physiological significance”. In: *Microvascular research* 30.1 (1985), pp. 10–18.
- [16] Brian J Kirby. *Stokes Flow*. URL: <http://www.kirbyresearch.com/index.cfm/wrap/textbook/microfluidicsnanofluidicsch8.html#x53-1690008>.
- [17] Chet Miller. *Wall Shear Stress*. Jan. 2016. URL: <https://physics.stackexchange.com/questions/250227/wall-shear-stress>.
- [18] Nathaniel G dela Paz et al. “Role of shear-stress-induced VEGF expression in endothelial cell survival”. In: *J Cell Sci* 125.4 (2012), pp. 831–843.
- [19] Stuart Egginton et al. “Shear stress-induced angiogenesis in mouse muscle is independent of the vasodilator mechanism and quickly reversible”. In: *Acta Physiologica* 218.3 (2016), pp. 153–166.
- [20] Stéfan van der Walt et al. “scikit-image: image processing in Python”. In: *PeerJ* 2 (June 2014), e453. ISSN: 2167-8359. DOI: [10.7717/peerj.453](https://doi.org/10.7717/peerj.453). URL: <https://doi.org/10.7717/peerj.453>.
- [21] Pauli Virtanen et al. “SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python”. In: *Nature Methods* 17 (2020), pp. 261–272. DOI: <https://doi.org/10.1038/s41592-019-0686-2>.
- [22] Chun Hung Li and CK Lee. “Minimum cross entropy thresholding”. In: *Pattern recognition* 26.4 (1993), pp. 617–625.
- [23] CH Li and Peter Kwong-Shun Tam. “An iterative algorithm for minimum cross entropy thresholding”. In: *Pattern recognition letters* 19.8 (1998), pp. 771–776.
- [24] Leo Grady. “Random walks for image segmentation”. In: *IEEE transactions on pattern analysis and machine intelligence* 28.11 (2006), pp. 1768–1783.
- [25] Jaakko Sauvola and Matti Pietikäinen. “Adaptive document image binarization”. In: *Pattern recognition* 33.2 (2000), pp. 225–236.

- [26] TY Zhang and Ching Y. Suen. “A fast parallel algorithm for thinning digital patterns”. In: *Communications of the ACM* 27.3 (1984), pp. 236–239.
- [27] Aric Hagberg, Pieter Swart, and Daniel S Chult. *Exploring network structure, dynamics, and function using NetworkX*. Tech. rep. Los Alamos National Lab.(LANL), Los Alamos, NM (United States), 2008.
- [28] Tilman Gneiting, Hana Ševčíková, and Donald B Percival. “Estimators of fractal dimension: Assessing the roughness of time series and spatial data”. In: *Statistical Science* (2012), pp. 247–277.
- [29] Yixin Hu et al. “Tetrahedral meshing in the wild.” In: *ACM Trans. Graph.* 37.4 (2018), pp. 60–1.

APPENDIX

Appendix A

Major Outputs of Additional Examples

The input files for each example are available on GitHub. Simply match the name of the sample to the name of the text file in the `inputs/` directory.

<https://github.com/Ryan-A-Armstrong/mvn-analysis/tree/master/inputs>

Appendix B

Input Files for Sample Outputs

B.1 Sample A

`input.txt`

```
# MVN Image Analysis Input File Sample A
=====
Required Fields
=====
# Input image and scale values um / pixel
TIF_FILE=/home/ryan/Desktop/mvn-analysis/data/original_sample.tif
SCALE_X=1.2420
SCALE_Y=1.2420
SCALE_Z=1.1400
#### Pipelines to Run ####
SEGMENT_2D=1
MESH_25D=1
SEGMENT_3D=1
MESH_3D=1
SKEL_3D=0
VOLUME_ANALYSIS=1
NETWORK_2D_GEN=1
NETWORK_2D_COMPARE=0
```

```
NETWORK_3D_GEN=0  
NETWORK_3D_COMPARE=0  
##### Save Parameters #####  
OUTPUT_DIR=/home/ryan/Desktop/mvn-analysis/outputs/  
SAVE_2D_MASK=1  
SAVE_2D_SKEL=1  
SAVE_2D_DIST=1  
SAVE_2D_DISPLAY=1  
SAVE_2D REVIEW=1  
SAVE_25D_MESH=1  
SAVE_25D_MASK=1  
GENERATE_25D_VOLUME=1  
SAVE_3D_MASK=1  
SAVE_3D_SKEL=1  
SAVE_3D_MESH=1  
GENERATE_3D_VOLUME=1  
SAVE_3D_MESH_ROUND=1  
GENERATE_3D_ROUND_VOLUME=1  
SAVE_2D_NETWORK=1  
SAVE_3D_NETWORK=0  
##### Display Parameters #####  
PLOT_ALL_2D=0  
REVIEW_PLOT_2D=1  
PLOT_25D_MESH=1  
PLOT_3D_THRESH_SLICES=1  
PLOT_LUMEN_FILL=1  
PLOT_3D_MESHES=1  
PLOT_3D_SKELS=0
```

```

PLOT_NETWORK_GEN=1

PLOT_NETWORK_DATA=0
=====
Adjustable Parameters
=====

##### 2D Analysis #####
# original, rescale, equalize, adaptive
CONTRAST_METHOD=rescale
# entropy, random-walk
THRESH_METHOD=entropy
RWALK_THRESH_LOW=0.08
RWALK_THRESH_HIGH=0.18
BTH_K_2D=3
WTH_K_2D=3
DILA_GAUSS=0.333
OPEN_FIRST=0
OPEN_K_2D=0
CLOSE_K_2D=5
CONNECTED_2D=1
SMOOTH=1

##### 25D Analysis #####
H_PCT_25D=-1
CONNECTED_25D_MESH=1
CONNECTED_25D_VOLUME=0

##### GRAPH Analysis #####
CONNECTED_NETWORK=0
MIN_NODE_COUNT=3
NEAR_NODE_TOL=15

```

```

LENGTH_TOL=3

##### 3D Analysis #####
# Output parameters

CONNECTED_3D_MASK=1

CONNECTED_3D_MESH=1

CONNECTED_3D_SKEL=1

### Thresholding parameters

# original, rescale, equalize, adaptive

SLICE_CONTRAST=original

# sauvola, none

PRE_THRESH=sauvola

WINDOW_SIZE_X=15

WINDOW_SIZE_Y=15

WINDOW_SIZE_Z=7

BTH_K_3D=3

WTH_K_3D=3

CLOSE_K_3D=1

### Lumen filling parameters

# octants, pairs, ball

ELLIPSOID_METHOD=octants

MAXR=15

MINR=1

H_PCT_R=0.5

THRESH_PCT=0.15

THRESH_NUM_OCT=5

THRESH_NUM_OCT_OP=3

MAX_ITERS=1

# uniform, random

```

```
RAY_TRACE_MODE=uniform  
# sweep, xy, exclude_z_pole  
THETA=exclude_z_pole  
N_THETA=6  
N_PHI=6  
MAX_ESCAPE=3  
PATH_L=1  
FILL_LUMEN_MESHING=1  
FILL_LUMEN_MESHING_MAX_ITS=3  
ENFORCE_ELLIPSOID_LUMEN=1  
H_PCT_ELLIPSOID=0.5  
### Skeletonization Parameters  
SQUEEZE_SKEL_BLOBS=1  
REMOVE_SKEL_SURF=1  
SKEL_SURF_TOL=5  
SKEL_CLOSING=1
```

B.2 Sample B

input.txt

```
# MVN Image Analysis Input File Sample B
=====
Required Fields
=====
# Input image and scale values um / pixel
TIF_FILE=/home/ryan/Desktop/mvn-analysis/data/Shared/09092019_gel_region_10x.tif
SCALE_X=2.4859
SCALE_Y=2.4859
SCALE_Z=6.00
#### Pipelines to Run ####
SEGMENT_2D=1
MESH_25D=1
SEGMENT_3D=1
MESH_3D=1
SKEL_3D=0
VOLUME_ANALYSIS=1
NETWORK_2D_GEN=1
NETWORK_2D_COMPARE=0
NETWORK_3D_GEN=0
NETWORK_3D_COMPARE=0
#### Save Parameters #####
OUTPUT_DIR=/home/ryan/Desktop/mvn-analysis/outputs/
SAVE_2D_MASK=1
SAVE_2D_SKEL=1
SAVE_2D_DIST=1
```

```
SAVE_2D_DISPLAY=1
SAVE_2D REVIEW=1
SAVE_25D_MESH=1
SAVE_25D_MASK=1
GENERATE_25D_VOLUME=1
SAVE_3D_MASK=1
SAVE_3D_SKEL=0
SAVE_3D_MESH=1
GENERATE_3D_VOLUME=1
SAVE_3D_MESH_ROUND=1
GENERATE_3D_ROUND_VOLUME=1
SAVE_2D_NETWORK=1
SAVE_3D_NETWORK=0
##### Display Parameters #####
PLOT_ALL_2D=0
REVIEW_PLOT_2D=1
PLOT_25D_MESH=1
PLOT_3D_THRESH_SLICES=1
PLOT_LUMEN_FILL=1
PLOT_3D_MESHES=1
PLOT_3D_SKELS=0
PLOT_NETWORK_GEN=1
PLOT_NETWORK_DATA=0
=====
Adjustable Parameters
=====
#####
##### 2D Analysis #####
# original, rescale, equalize, adaptive
```

```

CONTRAST_METHOD=equalize

# entropy, random-walk

THRESH_METHOD=random-walk

RWALK_THRESH_LOW=0.45

RWALK_THRESH_HIGH=0.48

BTH_K_2D=3

WTH_K_2D=1

DILA_GAUSS=0.0

OPEN_FIRST=0

OPEN_K_2D=0

CLOSE_K_2D=0

CONNECTED_2D=1

SMOOTH=2

##### 25D Analysis #####
H_PCT_25D=-1

CONNECTED_25D_MESH=1

CONNECTED_25D_VOLUME=0

##### GRAPH Analysis #####
CONNECTED_NETWORK=0

MIN_NODE_COUNT=3

NEAR_NODE_TOL=15

LENGTH_TOL=3

##### 3D Analysis #####
# Output parameters

CONNECTED_3D_MASK=1

CONNECTED_3D_MESH=1

CONNECTED_3D_SKEL=1

### Thresholding parameters

```

```

# original, rescale, equalize, adaptive
SLICE_CONTRAST=original

# sauvola, none
PRE_THRESH=sauvola

WINDOW_SIZE_X=13

WINDOW_SIZE_Y=13

WINDOW_SIZE_Z=9

BTH_K_3D=1

WTH_K_3D=1

CLOSE_K_3D=1

### Lumen filling parameters

# octants, pairs, ball
ELLIPSOID_METHOD=octants

MAXR=15

MINR=1

H_PCT_R=0.5

THRESH_PCT=0.15

THRESH_NUM_OCT=5

THRESH_NUM_OCT_OP=3

MAX_ITERS=1

# uniform, random

RAY_TRACE_MODE=uniform

# sweep, xy, exclude_z_pole

THETA=exclude_z_pole

N_THETA=6

N_PHI=6

MAX_ESCAPE=1

PATH_L=1

```

```
FILL_LUMEN_MESHING=1  
FILL_LUMEN_MESHING_MAX_ITS=3  
ENFORCE_ELLIPSOID_LUMEN=1  
H_PCT_ELLIPSOID=0.5  
### Skeletonization Parameters  
SQUEEZE_SKEL_BLOBS=1  
REMOVE_SKEL_SURF=1  
SKEL_SURF_TOL=5  
SKEL_CLOSING=1
```
