ANALYTICAL CENTERLINE EXTRACTION AND SURFACE FITTING USING CT SCANS FOR AORTIC ANEURYSM REPAIR

A Thesis

Presented to the Faculty of the Graduate School

of Cornell University

in Partial Fulfillment of the Requirements for the Degree of

Master of Science

by

Vikash Ravi Goel

May 2005

ABSTRACT

Endovascular aortic aneurysm repair promises to provide great advantages over traditional open surgery. Though this nascent form of treatment has already been demonstrated to be safer and easier for the patient, there is a strong need for technological advances which can improve precision and simultaneously reduce the amount of human effort involved in planning surgery. Computational analysis is also essential in predicting the future behavior of endovascular stent grafts, because at this stage there is very little empirical knowledge of long-term stent dynamics.

We present a geometric analysis procedure which aims to address this need. Our goal is to begin with CT scan data collected from aortic aneurysm patients and to produce a fully analytical model of the patient's arterial geometry with minimal user interaction. The product of this highly automated process can be a powerful tool for surgery planning and stent design. Furthermore, it provides a basis upon which computational fluid dynamics and mechanical behavior simulations can be built. The analytical nature of this representation allows for direct mathematical analysis or for arbitrarily precise discretization.

Our geometric analysis begins with a voxel segmentation. The segmented data set is then used as input into a unique robust centerline extraction procedure. This centerline then serves as the basis for an analytical fitting procedure which produces a lofted B-spline surface model. After describing this analysis process in detail, we present results from three input datasets.

BIOGRAPHICAL SKETCH

Vikash Ravi Goel grew up in the Washington, D.C. metropolitan area. He attended Cornell University as an undergraduate, and very much enjoyed his experience. Emboldened by the wonderful experience he had earning Bachelor of Science degree in Computer Science, he continued on to study at Cornell's Program of Computer Graphics, where his work culminated in this thesis.

ACKNOWLEDGEMENTS

Thank you, Don, for making this work possible. I am grateful for the opportunity you gave me. The educational experience and the chance to contribute to the Program of Computer Graphics were wonderful.

Thank you, Gün, for your support and advice. Your care and encouragement helped me tremendously.

Thank you, Hurf, both for always helping me get the tools I needed to do this work, for your very skilled storytelling, and for keeping me sane with our gearhead conversations.

Thank you, Peggy, for cheerfully dealing with everything complicated about my experience here.

Thank you, Linda, for always keeping track of me and helping to make sure I did what I needed to do.

Thank you, Martin, for keeping all the cool gizmos in the lab running.

Thank you, KB, for always volunteering your help.

Thank you, Steve, for offering me help and understanding. And thank you for the little tidbits of automotive knowledge you've given me.

Thank you, Adam, for everything. You've been my closest friend for nearly seven years now. Depending on what I needed most, you've been like a brother, a father, and a teacher to me. I could never have made it here without you.

Thank you, Bea, for giving me the support I needed and doing your best to help me whenever you could.

Thank you, Jacky, for helping me hash through these ideas and for being a good friend.

Thank you, Mike, for being one of the most well-rounded persons I know and helping me every chance you had.

Thank you, Spf, for allowing me to take advantage of your knowledge and experience.

Thank you, Henry, for bringing in the high-glucose treats that kept me going.

Thank you, Ryan, for the guidance you provided me and for all your hand-me-down workstations.

Thank you, Jeremiah, for providing the architect's view of things.

Thank you, Mary, for taking such good care of all of us in the PCG, and for putting up with me for so long.

Thank you, Chachi, for always calling me when I've neglected to keep in touch, and for all your empathic understanding.

Thank you, Mom, Dad, and Shiva for everything you've given me: your loving support, your gentle care, and your hopeful advice.

Thank you, Legacy Central, for providing more emotional support than I thought would have ever been possible from an online automotive forum.

This work was supported by the National Science Foundation Science and Technology Center for Computer Graphics and Scientific Visualization (grant number ASC-8920219) and by Doctor Roy K. Greenberg and his staff at the Cleveland Clinic.

TABLE OF CONTENTS

LIST OF FIGURES

Chapter 1

Introduction

Aortic aneurysm rupture ranks 13th among leading cause of death in the United States, killing 15,000 people every year. Traditional aortic aneurysm repair involves highly invasive and traumatic surgery. Recently, minimally-invasive repair techniques have emerged as a viable alternative to open surgical repair. These techniques can benefit greatly from computational analysis.

1.1 Aortic aneurysms

An aneurysm is a weakening of the wall of a patient's artery. The arterial lumen, the space within the artery through which blood flows, becomes enlarged as blood pressure deforms the arterial tissue. Left unchecked, an aneurysm can rupture, a lethal event.

The aorta is the main artery leaving the heart. An aneurysm of the aorta is of exceptional concern; due to the volume and pressure of blood flowing through the aorta, a rupture is very likely to result in death.

With modern medical imaging techniques, an aortic aneurysm can be detected before it becomes life-threatening, and the unreliable portion of the blood vessel can be repaired.

1.1.1 Traditional repair

The conventional method of repairing an aortic aneurysm is a highly invasive one. An incision is made running the length of the patient's torso in order to access the aorta. The aneurysm is cut out and replaced with a graft. A patient typically

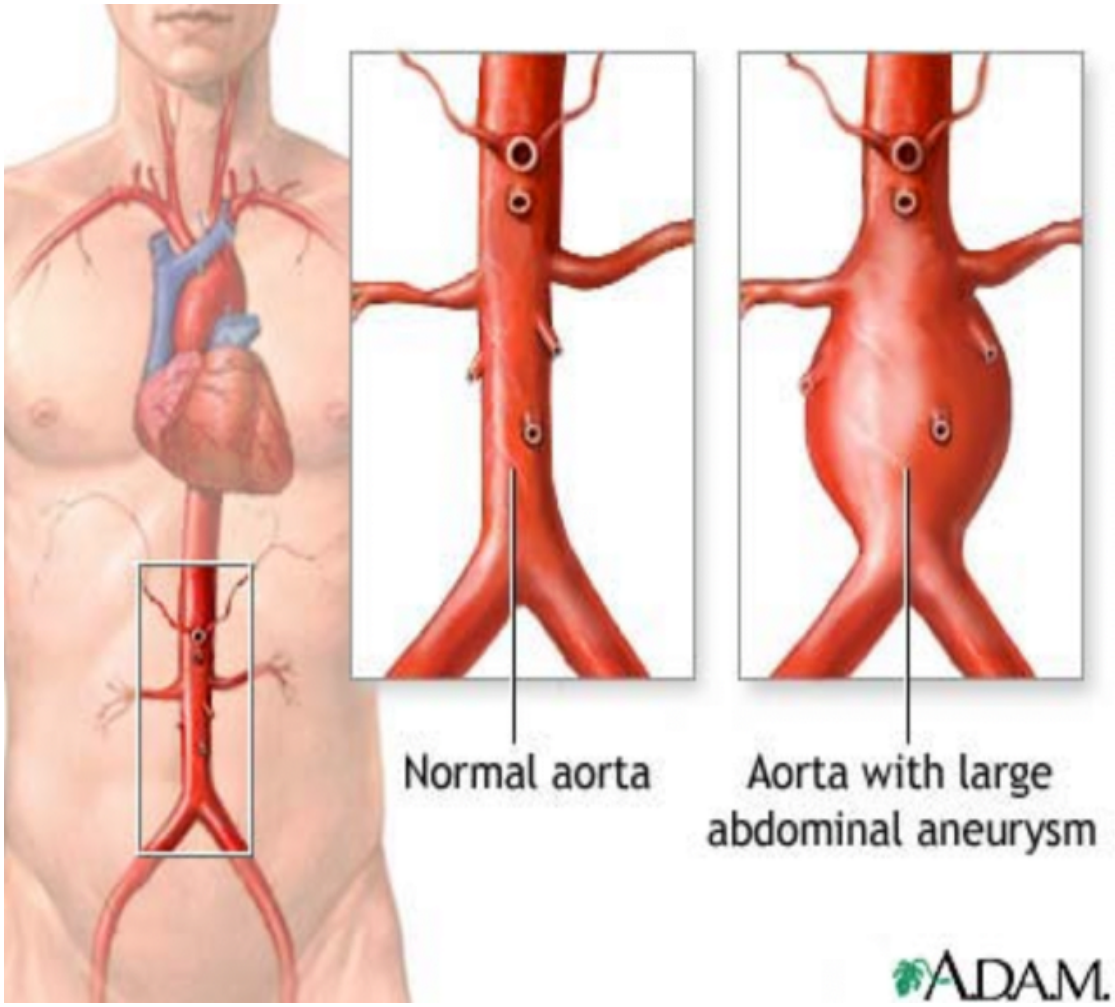


Figure 1.1: *An aneurysm is a weakening of the arterial wall, causing a bulge in the artery. Aneurysms of the aorta are likely to be fatal if allowed to grow unchecked. Image courtesy of Doctor Roy K. Greenberg, Cleveland Clinic.*

spends weeks or months in recovery, and can be at high risk of infection or other complications.

1.1.2 Endovascular repair

As an alternative to traditional invasive aortic aneurysm repair, engineers and surgeons have developed endovascular repair techniques. In endovascular aneurysm repair, a bracing stent is implanted into the patient's aorta by tools inserted through a small incision in an artery in the groin. The stent begins in a compressed form. It is positioned carefully and then unfolded. Once the procedure is complete, the stent takes on the full force of blood flow, relieving the arterial wall of the pressure.

Endovascular repair has marked advantages over open surgical repair. The greatly reduced amount of damage to healthy tissue means healing can happen much more quickly and safely. Patients typically can resume their normal activity after one or two weeks, and are less likely to develop complications.

Though it is enjoying greater and greater success, endovascular repair is still an experimental treatment. The technological sophistication of the procedure, though quite impressive, is still in great need of augmentation.

In planning the endovascular repair surgery, the stent must be designed to specifically fit the patient's anatomy. At present, anatomical information is collected via 3-D medical imaging techniques. Scans of the patient are then analyzed by the surgeon and engineers, who determine the appropriate specifications for a stent. 3-D printing techniques are sometimes employed to aid in this analysis. The stent is modeled with computer aided design, and then manufactured according to these precise parameters.

This planning procedure has two notable weaknesses: geometric complexity and

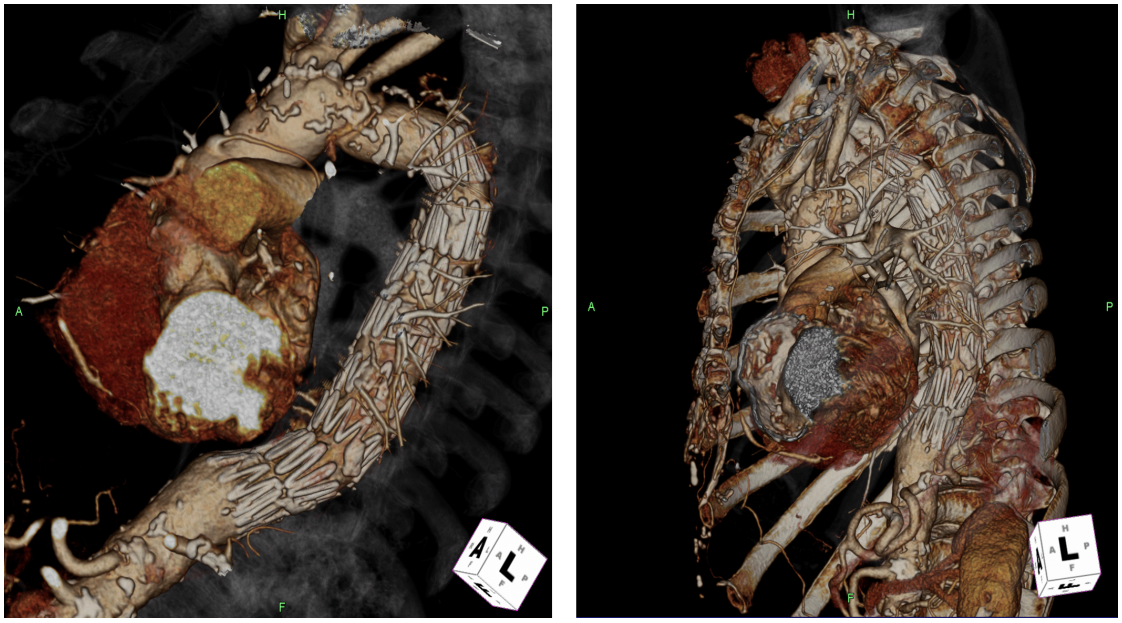


Figure 1.2: *Renderings of medical scans of an aortic aneurysm patient with a stent implanted. The stent bears the stresses of blood flow, effecting a repair of the aneurysm. Images courtesy of Dr. Roy K. Greenberg, Cleveland Clinic.*

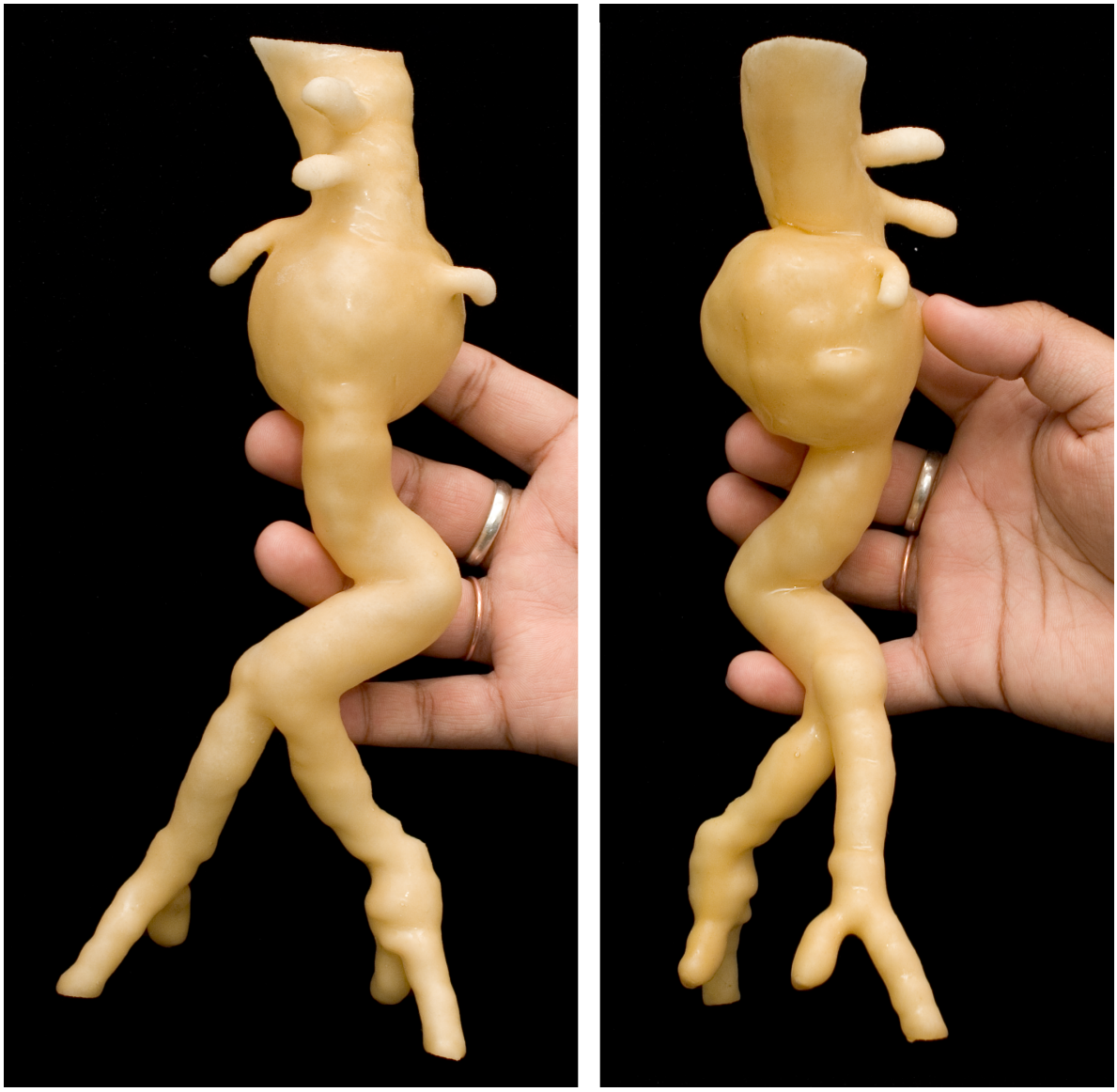


Figure 1.3: *A rapid prototyping device is sometimes employed to produce a real-life 3-D model of the patient's anatomy. This model is used in the design of a stent tailored to the patient's arterial geometry. Courtesy of Dr. Roy K. Greenberg, Cleveland Clinic.*

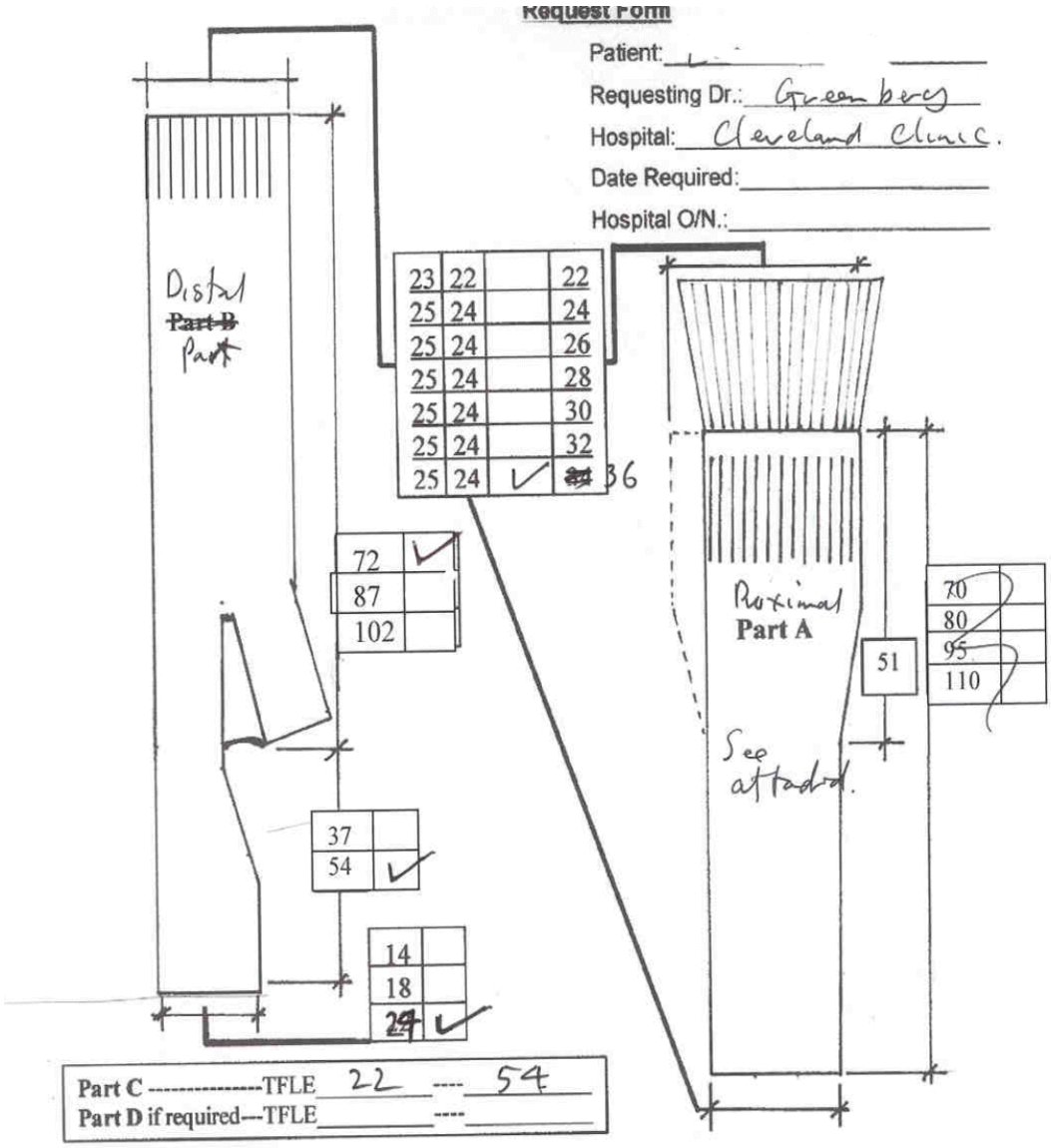


Figure 1.4: *Each stent is designed specifically for the patient it is intended for. The surgeon prepares a specification describing the geometry required. Courtesy of Dr. Roy K. Greenberg, Cleveland Clinic.*

user-intensiveness. The amount of measurement a human can perform on a 3-D scan is limited, so patients with complex and tortuous arterial geometry are poor candidates for endovascular repair. Currently stents are essentially cylindrical, or "Y"-shaped with cylindrical branches. Furthermore, determining the specifications and designing the stent involves a great deal of effort and time from those whose time is very valuable.

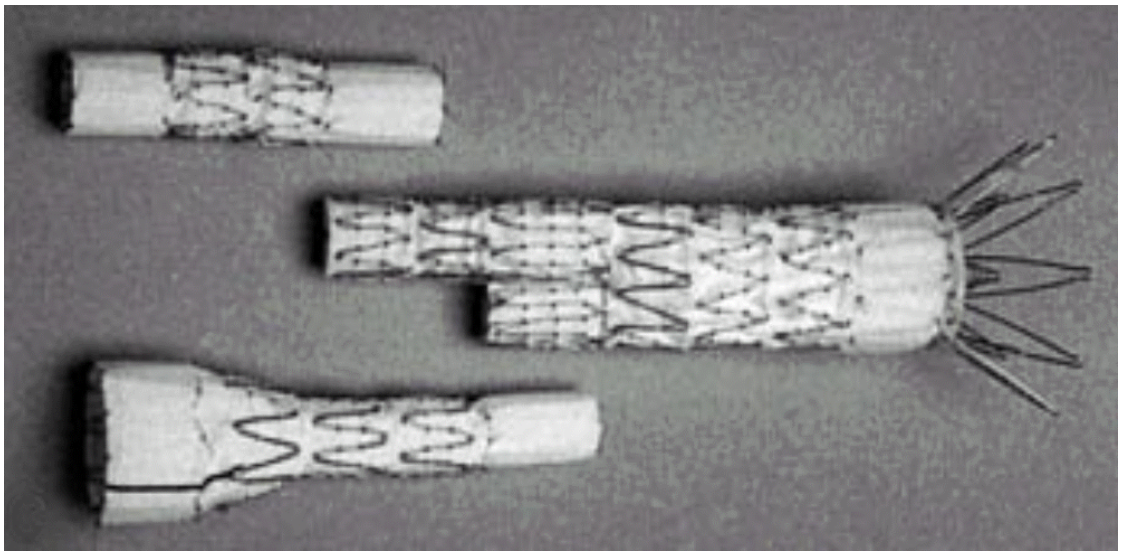


Figure 1.5: *Stents used for endovascular aneurysm repair. The design of the stent is such that it can be collapsed into a very narrow compact form which can be fed through narrow arteries. Note the relatively simple geometry of current stents.*

1.1.3 Convergence

Modern 3-D medical imaging techniques are rapidly improving, in both resolution and dynamic range, and can now collect far more data than can be usefully processed by a human operator. Computational analysis provides a means to make use of all this information.

In our work we have sought to leverage processing power in order to simultaneously reduce the amount of user effort required in the surgery planning process and improve the reach of minimally invasive aortic aneurysm repair to more patients.

The scanning step of the surgery planning process involves CT scans of two imaging modalities: the native scan representing the patient's anatomy directly, and the arterial scan performed with a dye injected into the bloodstream, increasing the contrast of the arterial lumen.

The convergence of these imaging modalities with our computational analysis techniques has allowed us to generate a fully analytical description of the complex geometry of a patient's aorta and its main branches, in a highly automated fashion. We have developed a centerline extraction procedure which involves minimal user interaction. The extracted centerline then provides the foundation for a surface fitting procedure.

The result of this procedure is a compact representation of the patient's arterial anatomy which can be analyzed directly or discretized with arbitrary precision. This description can serve as a powerful input for stent design and also for simulation models for stress analysis and computational fluid dynamics.

1.2 Organization

The remainder of this thesis is organized as follows: Chapter 2 briefly reviews medical imaging techniques and outlines previous work in centerline extraction. Chapter 3 describes our centerline extraction algorithm. Chapter 4 details our analytical curve and surface definitions and the fitting procedures. Results of our analyses are presented in Chapter 5, and we conclude with a discussion of our work and possible future research in Chapter 6.

Chapter 2

Previous Work

This chapter will first give a brief review of the medical imaging techniques and then outline the basic concept of centerline extraction. Furthermore it will discuss previous work relating to the centerline extraction task, enumerating the various approaches researchers have taken in the past.

2.1 A brief review of medical imaging modalities

The development of modern medical imaging began in 1895 when Wilhelm Roentgen discovered that a screen coated with barium platinocyanide would glow when exposed to the emissions of a Crookes tube. He found that the emissions could not be the cathode rays he expected because they went straight through physical objects and traveled far beyond the few centimeters a cathode ray can manage before absorption. It did not take him long to begin experimenting with a variety of objects placed in the path of this new type of radiation, which he dubbed the "X-ray." He found X-rays could travel through relatively lightweight material easily but were attenuated by dense material. The most shocking result was found when his hand fell into the path of the radiation: on the screen he saw the bones inside his fingers. He could see inside his body.

Before the advent of the X-ray radiograph, it was impossible for a doctor to see what was happening within a patient without exploratory surgery. It is difficult to imagine the relative ignorance of the inner workings of a living person, and the subsequent imprecision in diagnosis and treatment of injury and disease, prior to this technology.

Medical imaging has advanced greatly since the late 19th century. Paired with the powerful capabilities of modern computers, the magical abilities of medical imaging instrumentation have literally allowed a new dimension in analysis, three-dimensional imaging techniques which have the potential to provide accurate information about the geometry of the scanned subject. Early 3-D medical imaging techniques were iterative extensions of 2-D techniques; the images consisted of series of 2-D images. More advanced and sophisticated 3-D imaging is now capable of collecting and visualizing the patient's anatomy in three fully independent dimensions.

There are four commonly employed 3-D medical imaging modalities in use today: ultrasonography, computed tomography, magnetic resonance imaging, and positron emission tomography. It is important to understand that there exists great variety in implementations and that the technology is constantly advancing, but each of these four techniques will be briefly explained here.

2.1.1 Ultrasonography

Medical ultrasonography (often called ultrasound or simply US) was invented in 1953 by Inge Edler and Carl Hertz. Their intention was to image the interior of a human body using sound, and they developed an effective system.

The patient's inner anatomy is imaged using short-wavelength sound waves. The speed of sound through any particular medium is dependent on the density of that medium, and when a sound wave crosses an interface between media of differing density, a portion of the wave is reflected back. This is much like the behavior of light at an interface between media of differing indices of refraction. Ultrasonic receivers measure both the time offsets and the intensities of these

reflections and construct an image from these data.

Initially, this yielded a two-dimensional ultrasonograph. These 2-D sonograms are still very commonly used today. However, recent advances have allowed ultrasound imaging to reconstruct three-dimensional data. There exist several different methods for producing volumetric data from ultrasound scans, but they generally involve collecting multiple 2-D slices as samples of the desired 3-D dataset, and then extrapolating and interpolating any missing information.

Data collected from 3-D ultrasound is typically either cylindrically sampled or sampled along a helix. [1] It is often then resampled along a regular grid for processing and visualization.

2.1.2 CT

Computed tomography (abbreviated CT or sometimes CAT for computed axial tomography) is an x-ray technique that makes heavy use of computer processing. It was invented in 1972 by Godfrey Hounsfield. The concept is relatively straightforward: an x-ray detector placed opposite from an x-ray emitter can record the integral of x-ray opacity along the line between the detector and emitter. If enough data is collected about these integrals, a computer can reconstruct the x-ray opacity at individual three-dimensional points.

After reconstruction is complete, the resulting data is typically in the form of numerous parallel two-dimensional slices. These can then be processed as a

[1] It is important to understand that whether the samples are taken at regularly-spaced points in a cylindrical coordinate system or at regularly spaced points along a helix, neither dataset is ready for visualization on a rectangular display. Though each rotation of the helix closely approximates a plane, as does each slice of a cylinder, a proper resampling step must be performed in order to give the physician an accurate understanding of the patient's anatomy.

three-dimensional voxel grid.

2.1.3 MRI

Conceived in 1970 by Raymond Damadian, magnetic resonance imaging, or MRI, works on the concept of nuclear magnetic resonance. Electromagnetic radiation of the correct frequency can cause an atomic nucleus to absorb the incoming energy and enter an excited state. When the nucleus decays back to the stable state, it re-emits the captured energy. MRI scanners use radio frequency pulses that cause this interaction with hydrogen atoms.

Once the atoms are in the excited state, the scanner collects data through RF detectors that record how long the atoms took to decay and what the relative density is. This is indicative of the nature of the tissue. The resulting image is generally displayed as slices similar to CT scan data.

2.1.4 PET

The concept behind positron emission tomography, or PET, was first explored in the 1950s by Michael Ter-Pogossian. It slowly evolved to its current state, but the basic idea has remained the same: A dye containing a positron-emitting radioactive substance is injected into the patient. The positrons emitted by the dye annihilate with nearby electrons and emit photons. These photons are emitted in pairs and in opposite directions. Photon detectors pick up the photon emissions and then, much as in CT scanning, compute a 3-D image of the subject. Again, this data is generally examined in the form of 2-D slices of the 3-D dataset.

2.2 Centerline extraction

Our goal in this work is to extract the geometry of the arterial lumen from the processed output of a CT scanner. We need to select a method of 3D geometric representation appropriate for this goal. We have chosen to first extract the centerline of the lumen and represent this as a continuous B-spline. The surface of the lumen is then created using a lofting technique. This approach results in a concise representation of complex geometry which can be analyzed or discretized with arbitrary precision.

The concept of a centerline is intuitively simple. The centerline of an elongated three-dimensional volume is a one-dimensional curve outlining a path along the center of the volume. If the object's structure is "treelike," then the centerline should also take the form of a branching tree.

However, there is no generally accepted formal definition for a centerline. In fact, it would not be inaccurate to say that every proposed automated centerline extraction algorithm is in fact a definition of the term, much as a mathematical proof-by-construction is simultaneously an algorithm and a proof of existence.

There are, however, commonalities between centerline extraction approaches. Almost universally, they deal with data of the same form. The data are in the form of 2-D slices which make up a voxel set. The sampling density is generally somewhat higher within the slices than between slices, but the format of the data is basically that of a regular grid of scalar values, with anisotropic resolution.

The significance of each scalar value depends on the imaging modality; however, it is generally the case that tissues of different type have significantly different types of values. It is on this basis that the voxel dataset can be segmented. Once the desired object's volume has been segmented from the background, it has the form

of a set of voxels with a binary labeling – each voxel is either "in" or "out." It is within the "in" subset that the centerline is sought.

Existing automated centerline extraction algorithms can generally be grouped into three broad categories: peeling, shortest-path construction, and distance-based approaches.

2.2.1 Peeling

Peeling algorithms generally consider the centerline to be a subset of the set of voxels making up the object. Their goal is to remove all voxels which are not part of the centerline; what remains, then, is the centerline. They get their name from the analogy to peeling an onion. They are also often referred to as thinning algorithms [Bru01][GS99].

The iterative step of these algorithms is to remove the outermost layers of voxels, while simultaneously maintaining certain invariants. These invariants include the overall topology of the object, the presence of the voxels at the beginning and end of the line, and – depending on the particular implementation – continuity of the voxel set.

3-D peeling algorithms draw inspiration from 2-D peeling algorithms which were developed to determine skeletons of 2-D shapes. Unfortunately, the complexity rises dramatically when the algorithm is generalized to 3-D. A point whose removal would not affect the topology of the shape is called a *simple point*, and in the two-dimensional case, it is a relatively simple matter to examine each point's neighbors and determine the point's importance. In the three-dimensional case, the number of cases rises geometrically and one must use rules and classes of cases rather than tracking each case individually. Moreover, due to the sheer volume of

points and the requisite numerous iterations, peeling algorithms are computationally expensive.

There have been successful attempts to overcome these difficulties, however. A great deal of research has been done on how to quickly classify points and perform thinning in a parallel manner. Ma and Sonka developed what they call a fully parallel 3D thinning algorithm, which can be applied simultaneously to all voxels in the object [MS96]. It makes use of a set of *deleting templates*. If a point's neighborhood matches a generalized template after rotation and/or reflection, the point is judged simple. The templates are structured such that the deletion of simple points can be performed in parallel.

Ma and Sonka's algorithm works well on smooth computer-generated data sets, but falls somewhat short when subjected to noisy real-life input.

2.2.2 Shortest-path construction

Shortest-path construction approaches make use of graph algorithms, particularly Dijkstra's shortest-paths algorithm [Dij59]. Dijkstra's algorithm, given a graph and a node marked as the source, determines the distance from the source to every other node in the graph. This distance, measured at every node, is the *distance from source field*.

To begin, a graph is constructed with the same topology as the voxel set. Each node represents a voxel, and is connected by an edge to each node representing its physical neighbors. The length of each edge is chosen to reflect the physical distance between voxels.

A beginning voxel and an ending voxel are chosen, and the shortest path through the graph is computed through their nodes. The resulting series of nodes

is mapped back to a series of voxels, which represent the centerline through the object [ZT99].

Without other modifications, this technique has a clear weakness dealing with objects of high curvature; the shortest path through the object tends to hug the inner edge of corners, taking the "racing line" through the object, as demonstrated in Figure 2.1.

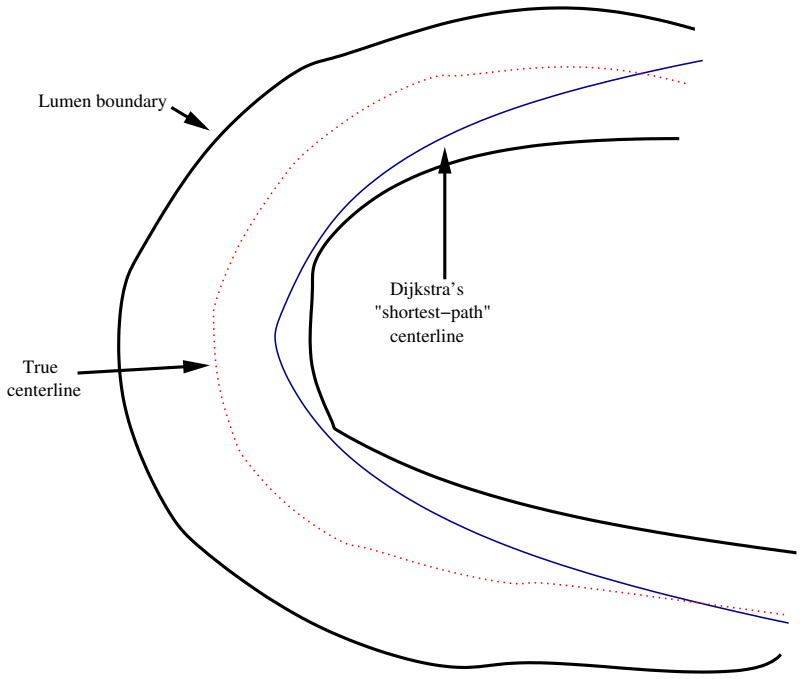


Figure 2.1: *Simple Dijkstra-based centerline algorithms tend to choose centerlines that get very close to the object's boundaries in areas of high curvature. This "cutting-corners" effect is not surprising given that the purpose of Dijkstra's algorithm is to identify the shortest path. The solid blue curve shows an example of what a shortest-path construction might yield. Note the significant deviation from the dotted red line representing the true centerline.*

Another obvious shortcoming of simple Dijkstra-based approaches is that they do not offer a clear extension to allow for bifurcating shapes, or any shape without

a cylindrical topology. This is, however, often not a material weakness, as there are many useful cases where the object in question is indeed cylindrical in topology.

There are several ways of dealing with this difficulty. Samara et. al. explored clustering voxels of similar distance from the source, in hopes of exploiting the longitudinal shape of a human colon imaged with a CT scan [SFD+99]. The centerline extracted this way was refined with the volumetric data, attempting to pull towards the center of the lumen. This helped somewhat, but their results still showed errors in regions of sharp curvature.

Bitter et. al. also experimented with methods of introducing "penalties" to Dijkstra-based algorithms, so as to encourage the resulting centerline to avoid voxels close to the boundary of the object [BKS01]. They increase the complexity of the graph by splitting each edge into three edges, with two dummy nodes added in between. The middle of the three new edges carries the same weight as the original edge, while the other two are available to provide extra weight to any path that may traverse them. They called this the *penalized distance from end field*, and compute the centerline using this more complex graph with good results. See Figure 2.2 for a visual explanation. Still, it cannot capture tree-like topologies.

2.2.3 Distance-based approaches

Distance-based approaches make use of the distance between a voxel and the boundaries of the object. There are a wide variety of algorithms which fall into this category.

Some proposed approaches are based on the *medial axis transformation* [SPB95]. The medial axis transformation of a volume is the locus of the largest spheres which fit within that volume. The centers of these spheres are then presumed to lie along

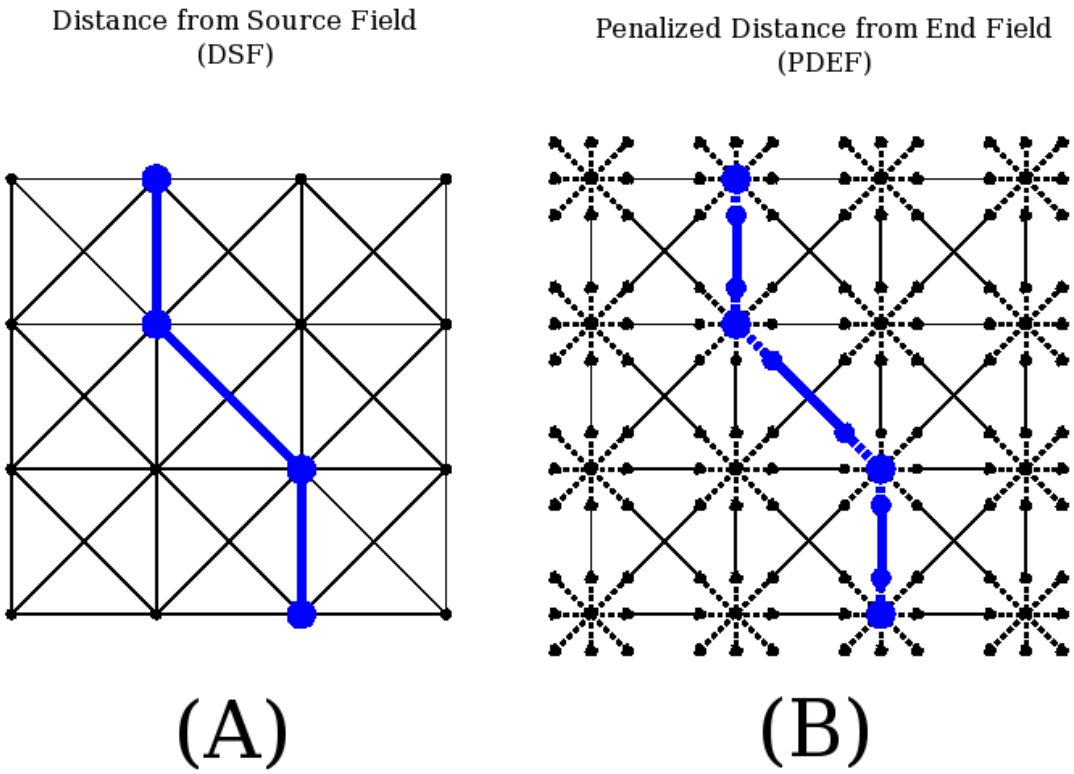


Figure 2.2: *Bitter et. al.'s approach to improving graph-based centerline extraction involves increasing the number of nodes in the graph by a factor of 27 and the number of edges by a factor of three. In (A) we see the original distance-from-source field. Each voxel is represented by a node, connected by an edge to each of its neighbors. Each edge's weight represents the distance between two voxels. The overall cost of a path between two voxels is equal to the distance between them. In (B) we see the penalized distance from end field, constructed by splitting each edge of the DSF into three edges, with dummy nodes spliced in. The new edges, drawn with dashed lines, have their weights set to penalty values chosen by the algorithm. Thus the overall cost of any particular path is equal to the distance between the endpoints plus the penalty costs of the new edges.*

the centerline of the object.

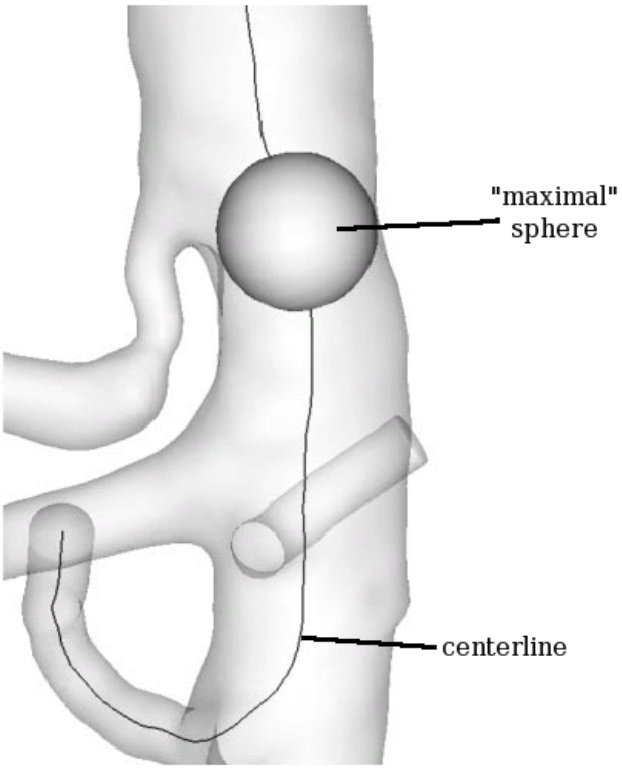


Figure 2.3: *An example of a maximal sphere. The medial axis transformation consists of the locus of all such maximal spheres. Figure from [AEIR03].*

Like peeling algorithms, medial axis-based algorithms were originally developed in 2-D, and suffer from problems which are much more easily solved in 2-D than in 3-D. Difficulty arises due to the fact that the number of spheres in the medial axis transformation is relatively small, and therefore the centerline is sparsely sampled. There are good ways to resolve the lack of connectivity in 2-D situations, but ambiguities arise in 3-D which are only worsened by the presence of noise.

Dey and Zhao approximate the medial axis transformation of a 3-D volume using a *Voronoi diagram* [DZ02]. A Voronoi diagram is a partitioning of a space containing marked points, such that each partition contains exactly one marked

point, and that marked point is the closest marked point to all points within the partition. Figure 2.4 shows a 2-D Voronoi diagram; in 3-D it would be a partition of 3-space rather than of a plane.

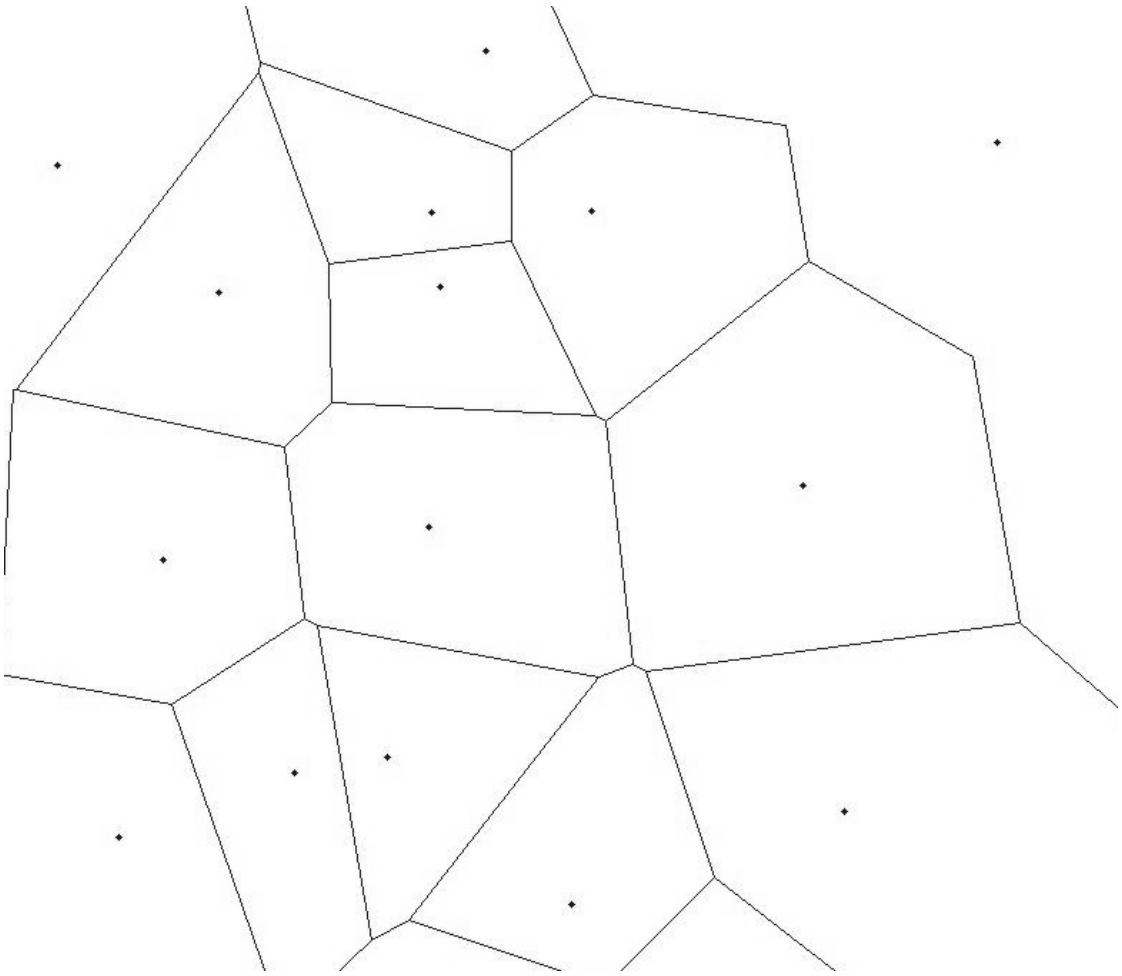


Figure 2.4: *An example of a 2-D Voronoi diagram. Each dot represents a marked point. The plane is partitioned such that each partition contains exactly one dot, and that dot is the closest dot to all points within the partition.*

They sparsely marked points along the boundary of the object and then filtered through the results of the Voronoi diagram computation. While they did not take the step of deriving a centerline from it, they did achieve some success in computing an approximate medial axis transformation. However, their algorithm was rather

sensitive to the values of a few certain parameters, and susceptible to noise. A post-processing smoothing step was required to reduce this problem.

Other distance-based centerline extraction approaches make use of the *distance transform* of an edge-detected data set [BSB+00]. The distance transform of a boolean array (be it a 3-D voxel array, a 2-D pixel array, or even a linear array) is a labeling of each element with the distance to the nearest true element.

Telea and Wijk [Tv02] developed a centerline extraction algorithm which they call an augmented fast marching method. It was developed as a 2-D algorithm. First, the boundary of the object is parameterized, so that the value of a smoothly varying parameter can be used to signify a particular point along the edge of the object. Then, the distance transform of the edge-detected object is computed by iteratively marching inwards from the boundaries. As the algorithm marches inwards, at each point it records the mean parameter value of the boundaries from which it originated. This provides a record of the "source" of the distance transform number at each point. The centerline is then simply the result of a simple edge detection operation on the image of these parameter values.

Telea and Wijk then extend the algorithm to produce 3-D centerlines in a different way: rather than attempting to generalize the algorithm to 3-D, they compute 2-D centerlines for every slice of constant x, y, or z. These slices are then all intersected to yield a 3-D centerline. This approach works fairly well, although there is no guarantee of connectedness in the resulting centerline.

2.3 Summary

The problem of centerline extraction has been attacked many times, resulting in many different approaches. While they can generally be placed into one of three

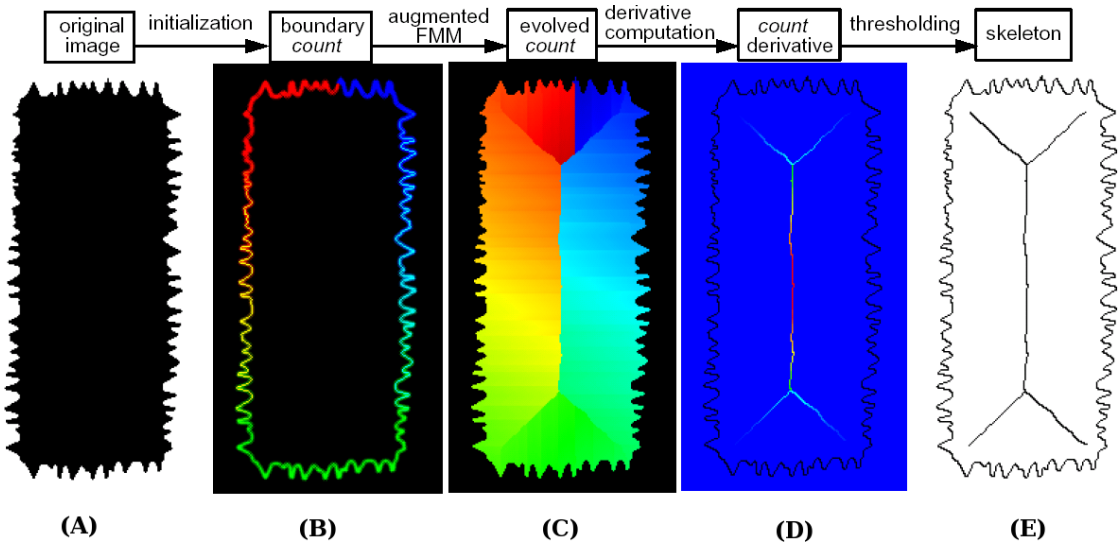


Figure 2.5: *An overview of Telea and Wijk's augmented fast marching method. The procedure begins with an input image (A). First the image is edge-detected and parameterized (B). In this diagram, the color of the border pixel represents the value of the parameter. Then the marching process takes place, carrying parameter values inwards (C). The derivative is computed (D), and finally the derivative is thresholded to provide an edge detection (E). Image from [Tv02].*

broad categories, each centerline extraction algorithm has specific goals and a particular form of input for which it is designed. In our case, our goal is to extract the centerline of the human aorta and main arteries from CT scans, and therefore our algorithm is geared towards handling bifurcations and tortuous curvatures. Meanwhile, we can rest assured that the source data is collected in a controlled manner, that the scan is performed with the subject in a controlled orientation, that the samples contain as little unnecessary noise as possible, and that the voxels are sampled regularly in a well-defined way. While all the outlined approaches provided a foundation for our work, we specifically drew inspiration from the distance-based methods in developing our algorithm.

Chapter 3

Centerline extraction

The overall goal of our work is to extract geometric information from biological CT scan data. The specific goal of this research is to determine the centerlines of a patient's aorta and major arteries, and to use that information to construct a preliminary model of the arterial lumen's surface.

This chapter will first explain the type of data we receive. It will then provide an overview of the entire centerline extraction process we have developed, and then explain each step in detail.

3.1 Our data

The initial data is collected by a CT scan machine. The CT scans are performed in anticipation of abdominal aortic aneurysm repair. Proprietary processing algorithms are employed within the CT scan machine in order to interpret and sample the data, and the output is in the form of a three-dimensional regular grid, presented as a series of horizontal slices perpendicular to the patient's axis.

Within each slice, sample density is fixed at approximately 1.595 linear samples per millimeter; the samples are roughly 0.627 millimeters apart. The axial sample density, however, can vary. We make use of both "3mm" scans and "1mm" scans. In a "3mm" scan, slices are sampled 3 millimeters apart. The higher-resolution "1mm" scans are misnamed – in actuality slices are sampled approximately every 0.8 millimeters.

Each scan can also be performed in one of two modalities. A standard scan performed on a patient who has not been otherwise prepared is known as a *native*

scan. In a native scan, the recorded intensity value for each sample is representative of the x-ray opacity of the corresponding portion of the patient's tissue. An *arterial* scan can also be performed. In an arterial scan, an x-ray opaque dye is administered to the patient, typically via intravenous injection. This dye flows through the blood vessels and highlights them in the scan by greatly increasing their opacities. In an arterial scan, the recorded intensity of each sample does not directly represent the opacity of the inside of the patient's arterial lumen. Examples of data from each modality are shown in Figure 3.1.

For each patient, both a native and an arterial scan are performed. Naturally the arterial scan is performed after the native one. Because the scans are intended primarily for human use rather than for computerized processing, the native scan is only a 3mm scan, while the arterial scan is performed at the higher 1mm resolution.

3.2 Overview of procedure

There are seven steps to our centerline extraction process. First is *preprocessing*, which conditions the data in preparation for the rest of the process. Next is *segmentation*: isolating just the portions of the voxel dataset which are part of the arterial lumen. After that, the next step is *edge detection*, in which the voxels at the borders of the lumen are identified. Then the *distance transformation* of the edge detected dataset is computed. From there, the *vector gradient* of the distance transform, and then a novel *scalar derivative* of the vector gradient field, is calculated. The final step is *identification of the centerline* of the lumen. The resulting data is ready for the analytical fitting stage described in the next chapter. These steps are shown in Figure 3.2.

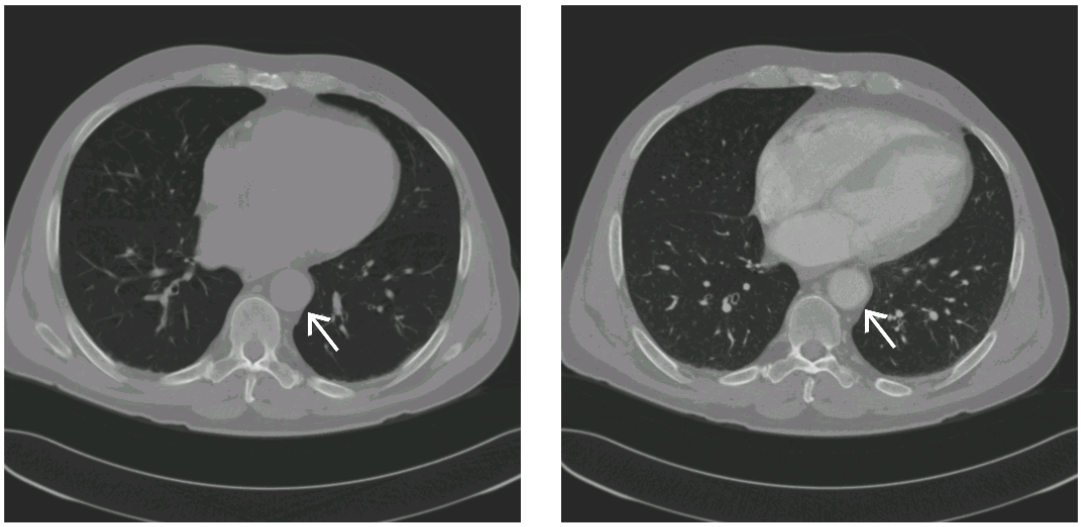


Figure 3.1: *To improve visualization, CT scans often make use of a contrast dye. This dye is generally either injected intravenously or taken orally, and is opaque to x-rays. A scan performed without this dye is called a native scan, and a scan performed with it is called an arterial scan. Above are examples of native (A) and arterial (B) CT scan slices. Note the difference in the arterial lumen's appearance. Note that these images have both been contrast-enhanced to demonstrate the difference.*

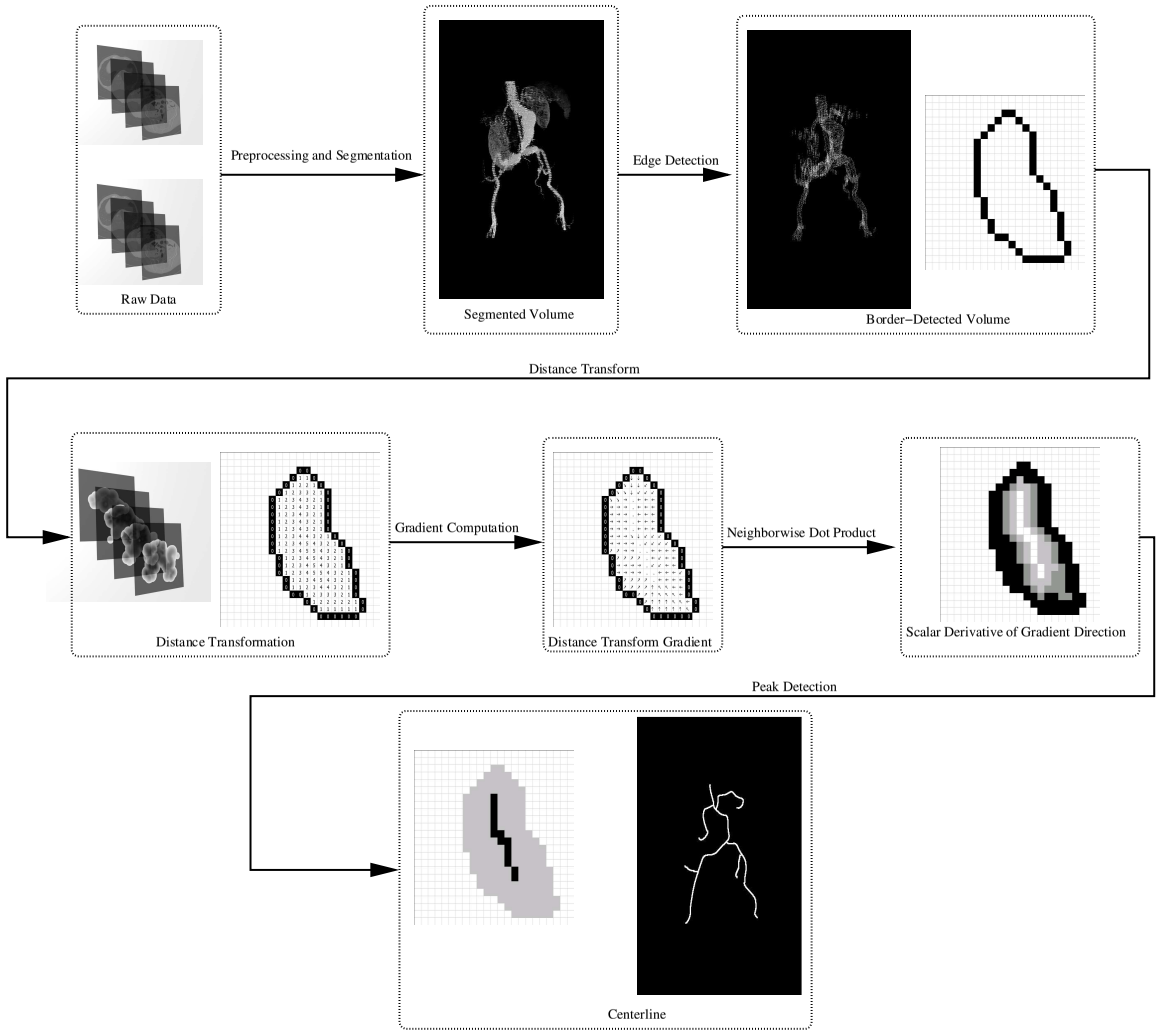


Figure 3.2: *A diagrammatical overview of the centerline extraction procedure detailed in this chapter.*

3.3 Preprocessing

As explained above, the two scans we begin with are performed at different resolutions. Naturally, they are also performed at different times. The purpose of the preprocessing step is to make these two datasets compatible.

Preprocessing involves two substeps. First, the low-resolution native scan must be upsampled to match the higher resolution of the arterial scan. Second, the two datasets must be registered to one another. This way, a particular voxel in one dataset correctly corresponds to the same voxel coordinates in the other dataset.

3.3.1 Voxel resampling

The first preprocessing substep is voxel resampling. The low-resolution native scan is upsampled to the resolution of the arterial scan. When the dataset is resampled this way, interpolation is needed to fill in the missing information.

There are many ways this interpolation can be performed. The simplest is "nearest-neighbor" sampling, where the value of the closest sample is chosen. Such a simplistic approach – which cannot even be realistically considered to be interpolation – tends to result in heavy aliasing artifacts.

A more sophisticated method of upsampling involves using linear interpolation to estimate the value of the image at points between samples. With a 1-D image, linear interpolation is easy to understand and compute. In 2-D, however, a naive approach to interpolation requires computation of Euclidean distances to determine the appropriate weights for the four samples of interest. This problem is compounded in 3-D. Fortunately, there is a computationally efficient way to perform multi-dimensional linear interpolation. Bilinear (in 2-D) or trilinear (in

3-D) interpolation performs the interpolation along one dimension at a time, as indicated in Figure 3.3.

More sophisticated than linear interpolation is quadratic or cubic interpolation, performed in a similar manner as bilinear or trilinear interpolation. In our tests, however, quadratic and cubic interpolation yielded very nearly the same results as linear interpolation (see Figure 3.4), so the extra processing was not deemed worthwhile.

3.3.2 Voxel registration

The second preprocessing step is registration. After the native scan is upsampled, the two voxel datasets have the same sample density; but because they were collected at different times, the coordinates do not necessarily correspond directly. This mismatch can be due to external movement of the patient, internal movement of the patient's organs, or movement of the scanner's sensing array.

In order to make use of both the native and arterial datasets, it is necessary that we register them to one another, so that correspondence between points is known. We perform this registration using a simple maximization algorithm. We specify the registration quality Q of a particular pair of datasets $A(x, y, z)$ and $B(x, y, z)$ to represent the closeness of the match:

$$Q = \sum_{x,y,z} \sqrt{A(x)B(x)} \tag{3.1}$$

This sum is essentially a convolution of the two datasets, except that the square root is taken in order to avoid excessively large sums. This matching metric rewards peaks matching other peaks. It also indirectly penalizes valleys not matching other

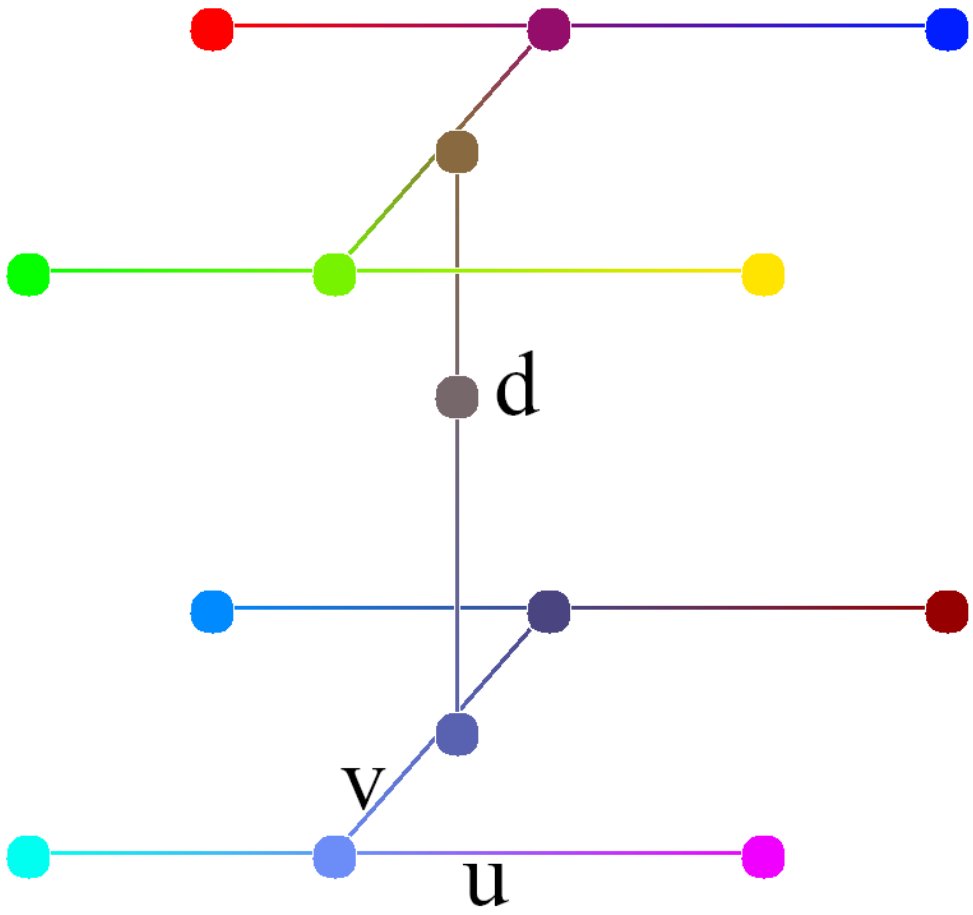


Figure 3.3: *Linear interpolation between voxels must be performed in a trilinear fashion. First, interpolation is performed in the x direction according to the u parameter (the distance to the nearest sampled x coordinate). This yields four interpolated samples at the correct x coordinate. Interpolation is then performed along the y direction between these two samples, according to the v parameter. This yields two interpolated samples at the correct x and y coordinates. A third linear interpolation is performed along the z direction according to the d parameter.*

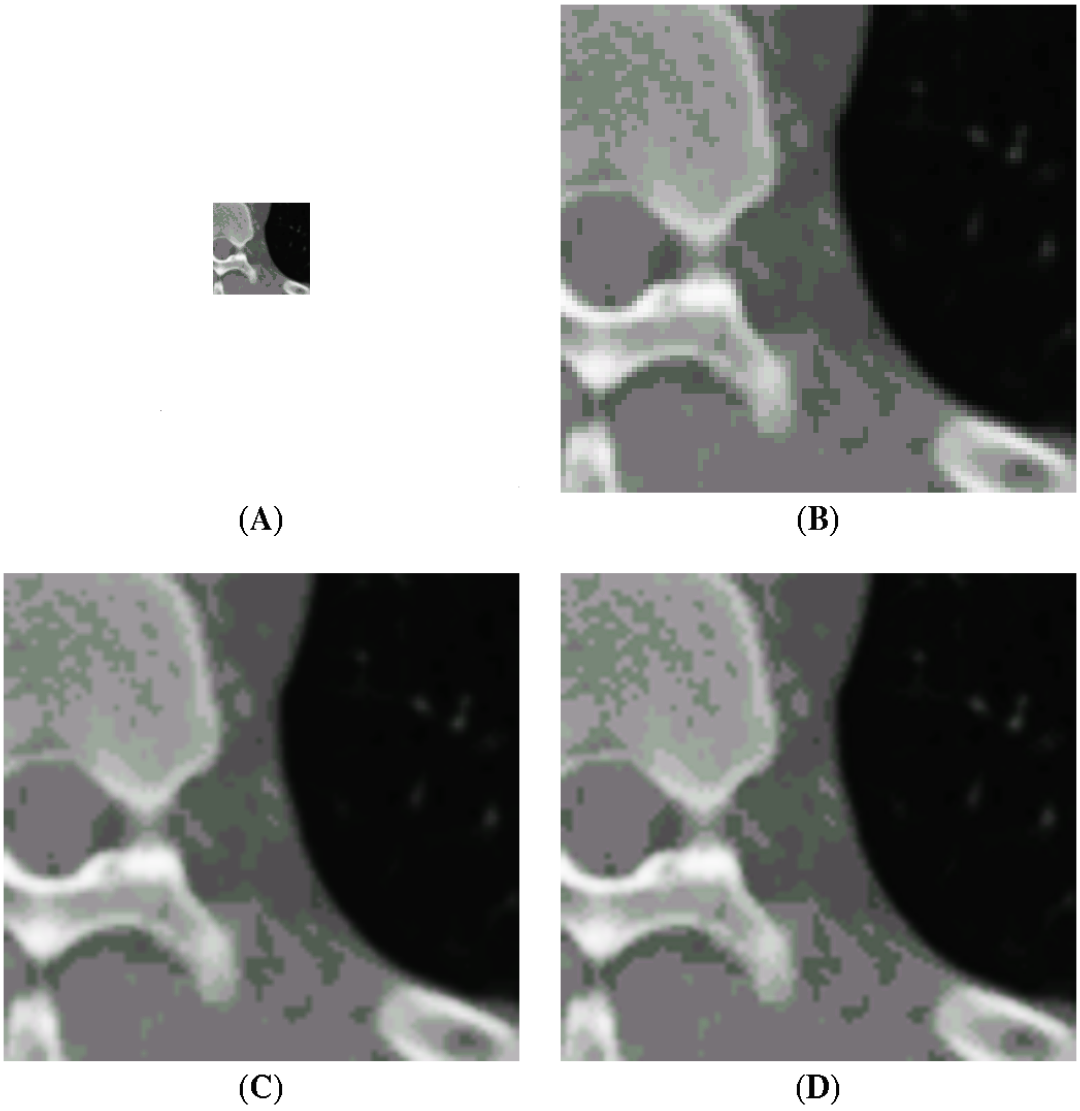


Figure 3.4: *Interpolation yields less aliased results than nearest-neighbor sampling. In (A) we see our original low-resolution image which we wish to upsample to a higher resolution. In (B) we see the results of a simple nearest-neighbor sampling. Notice the aliasing artifacts apparent as jagged boundaries. In (C) and (D) we see the results of bilinear and bicubic interpolation respectively. Notice the greatly improved output, and also the very small difference between linear and cubic interpolation.*

valleys. [1]

A simple search is performed through various reasonable translations to identify the one which maximizes the quality function Q. The translated native dataset is then used for the rest of the centerline extraction process.

3.4 Segmentation

The second step of the procedure is segmentation. The voxel data we begin with contains samples over the entire abdomen of the patient. The goal of segmentation is to isolate only the voxels contained in the region of interest. In our case, these are voxels within the arterial lumen.

It is generally the case that computers are good at tasks at which humans are poor, and vice versa. This is no exception; image segmentation is performed continuously and automatically in the human visual system, but difficult to encode computationally.

We approach segmentation using three separate substeps: intensity cropping, intersection, and connected components. These substeps are explained below.

3.4.1 Intensity cropping

Each sample in a CT scan has an intensity representative of that volume's opacity to x-rays. Since every type of material has a characteristic opacity, a preliminary segmentation can be performed simply by cropping according to intensity.

We perform this intensity cropping procedure on both input datasets. It is an interactive step requiring user input. The user is presented with the dataset in

[1] Consider the case where a peak is matched with a valley. The small value of the product minimizes the peak's potential contribution to the overall registration quality number.

slice form, as no usable volumetric rendering is yet possible. Simultaneously, a histogram representing intensity frequency in the slice is displayed.

The user must select upper and lower intensity bounds on the histogram. The user is provided with realtime feedback demonstrating the results of such a cropping operation. Because of the feedback, and because each type of tissue is represented as a local peak in the histogram, this operation only takes a few seconds for the user to perform. Once the two bounds have been inputted by the user, the intensity cropping computation is performed very quickly. These bounds are quickly and easily entered by clicking on an image of the intensity histogram for one slice, and the same bounds are applicable to the remainder of the voxel dataset.

Unfortunately, simply performing intensity cropping does not yield a properly segmented volume in either the native or arterial scans. In the native scan, we found that the abdominal viscera tended to have similar intensities to the arterial lumen. Meanwhile, in the contrast scan, we found that the more opaque lumen actually closely matched the intensity of the patient's bones.

3.4.2 Intersection

In order to further improve on the segmentation achieved with simple intensity cropping, we then make use of the additional information we have. The cropped native scan consists primarily of a union of the abdominal organs and the arterial lumen. The cropped arterial scan, meanwhile, consists primarily of a union of the skeletal structure and the arterial lumen. The intersection of these two volumes, then, represents a volume much closer to the isolated arterial lumen.

In the intersection step, we take the intensity-cropped arterial scan and remove all voxels not present in the intensity-cropped native scan. As shown in Figure 3.7,

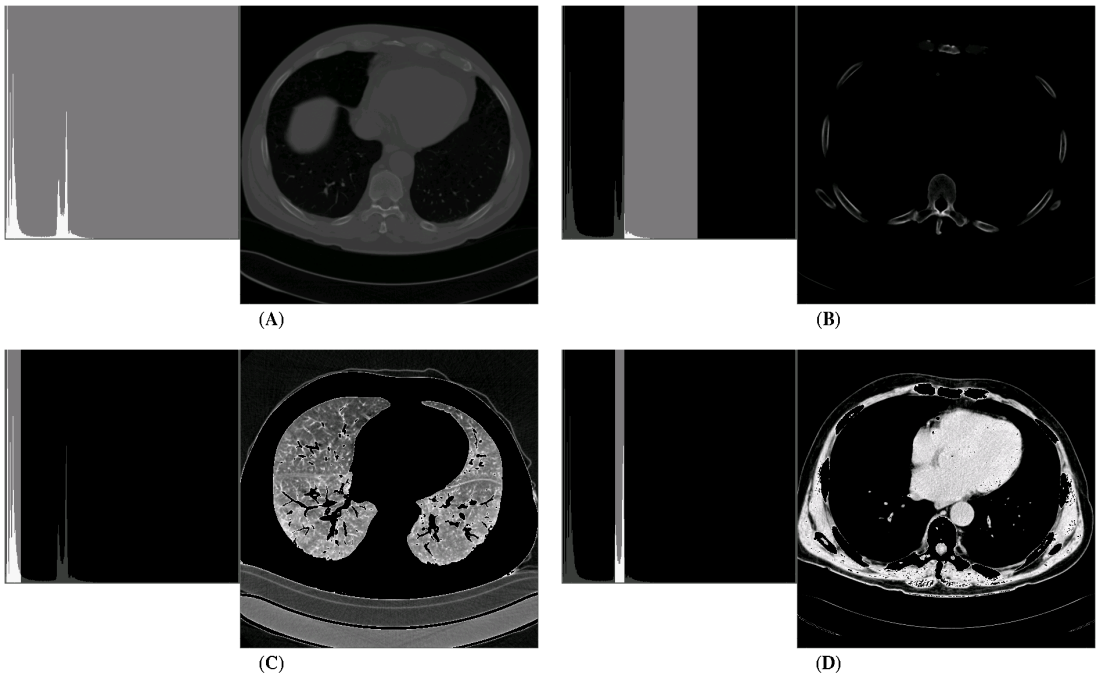


Figure 3.5: *Pictured above are four examples of intensity cropping. In each case, the intensity histogram is shown next to the image of a slice. The portion of the histogram within the user-specified intensity bounds is highlighted in light gray. In (A) we see a slice of the original dataset and the corresponding histogram of intensities. In (B) we see the patient's bones are clearly revealed by cropping a particular range of intensities. In (C) we segment the air inside the patient's lungs. In (D) we segment the water inside the body. Note that the arterial lumen shows up clearly in this image, but there still remains much other tissue.*

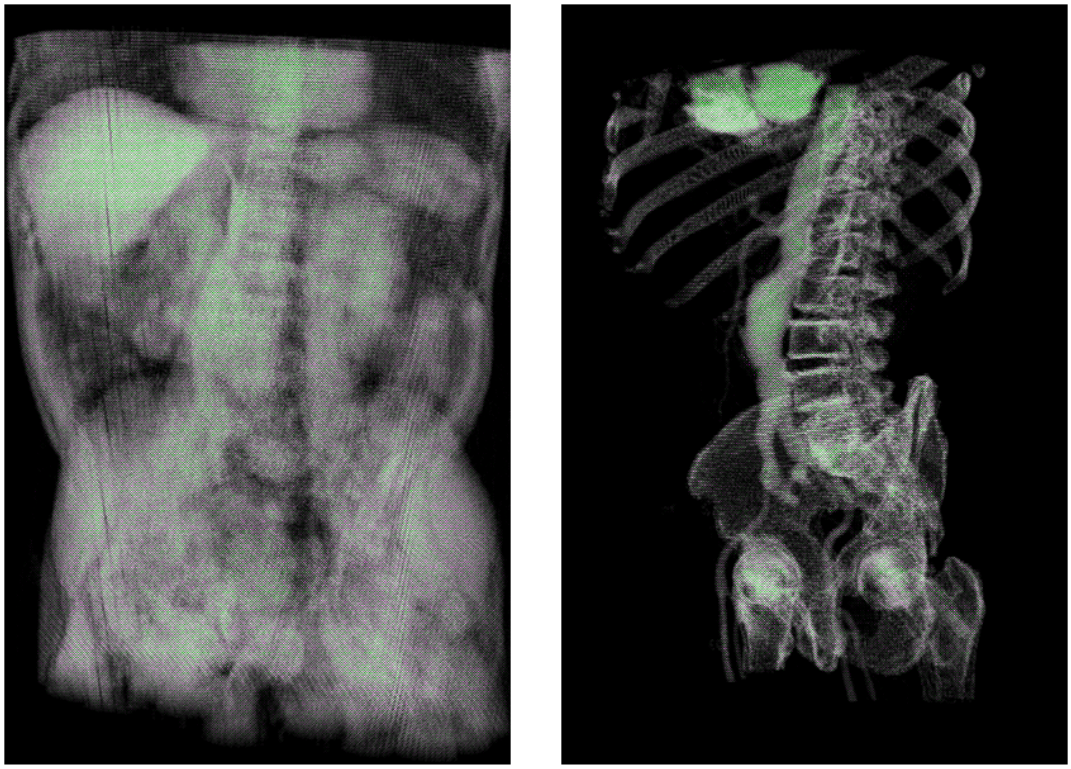


Figure 3.6: *Intensity cropping alone provides insufficiently precise segmentation. In the native scan (A), the abdominal viscera provide too much distraction. In the contrast scan (B), the situation is better, but still there is much distraction due mainly to the patient's bones.*

the result is a voxel dataset without a great many of the distractions noted in the previous subsection.

3.4.3 Connected component

The third and final substep in the segmentation process is computing a connected component. In order to eliminate the distractions in the intersected dataset, we note that the distractions are primarily spatially separated from the region of interest. Thus, if we define a neighbor relation on the voxel set and use it to isolate connected components, one such component will contain the closely isolated arterial lumen.

In practice there is no reason to run a complete "blob-finding" algorithm. Instead, we seed the processing with user input. The user specifies one point which is within the arterial lumen, and then our procedure computes the connected component containing that point using an optimized flood-filling algorithm.

In empirical testing, we found that using 18-connected neighborhoods, where two voxels must share either a face or an edge in order to be considered neighbors, worked best. We also tested 6-connected neighborhoods (where neighbors must share a face) and 26-connected neighborhoods (where neighbors must share a face, an edge, or a point); the former proved insufficient for our data and the latter showed no improvement over 18-connected neighborhoods despite the increased computation.

Our optimized flood-filling algorithm exploits the fact that the overall shape of the region of interest is largely cylindrical, with the axis in the z direction. Ordinarily, 2-D flood filling is much more efficient than 3-D flood filling due to the greatly reduced degree of recursion. We make use of this fact by requiring

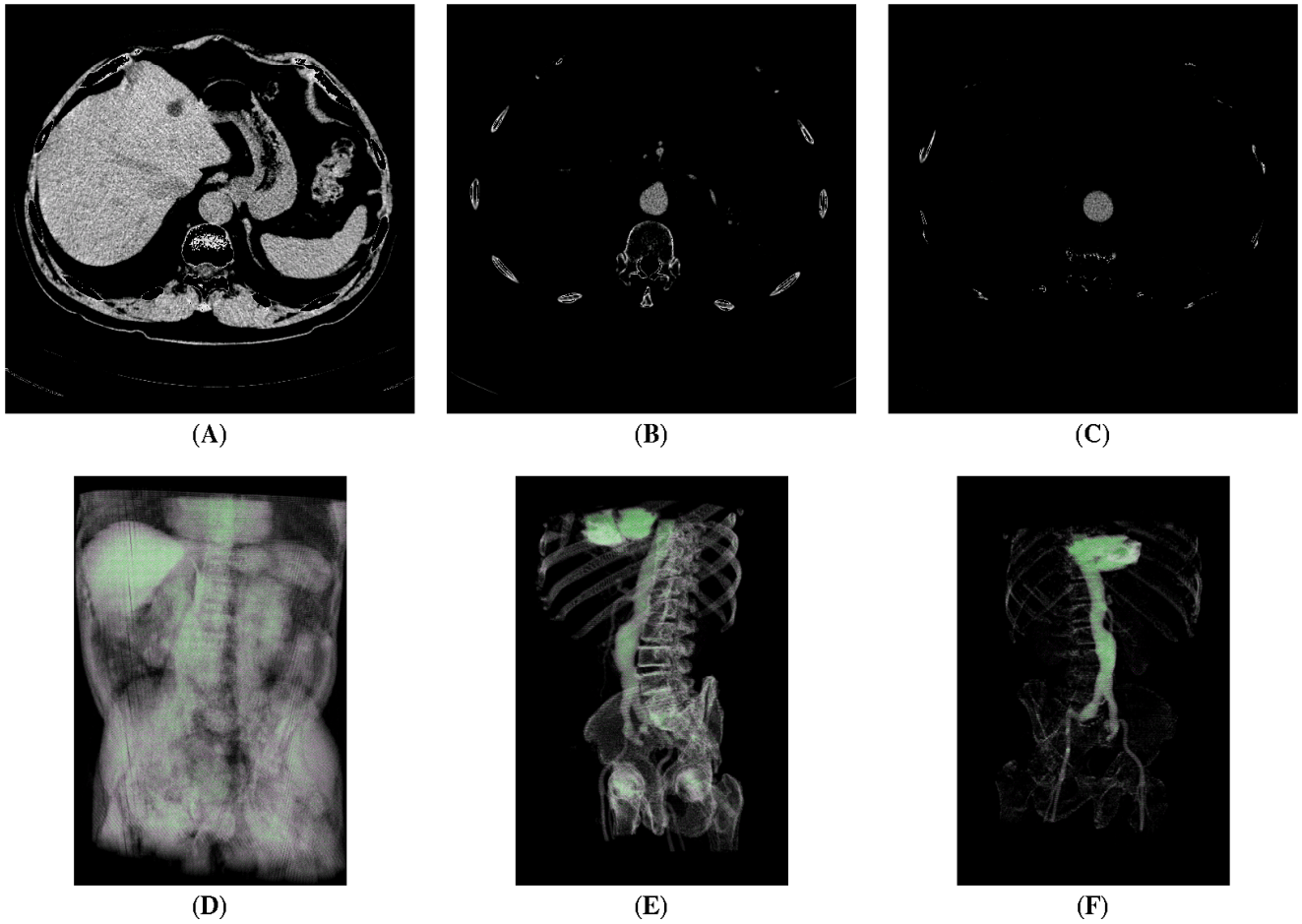
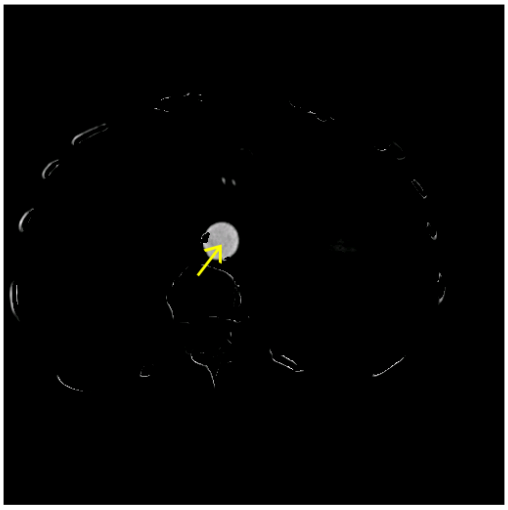


Figure 3.7: *We improve upon the segmentation accomplished with intensity cropping by intersecting the native and arterial volumes. In (A), (B), and (C) we respectively see the cropped native and arterial scans, and their intersection. In (D), (E), and (F) we see 3-D renderings of these voxel sets. The segmentation achieved in (F) is quite good, although we still notice some small distractions, primarily due to the patient's bones.*



(A)

(B)

Figure 3.8: *The final segmentation step is computing the connected component. We must collect one piece of information from the user: the coordinates of a voxel within the lumen. In (A) we see an example of how this data might be inputted; the user is shown a slice of the intersected dataset, and must select a point in the interior of the lumen. The arrow shows an example of a point the user might select. In (B) we see the resulting connected component. Notice that the lungs partially remain, but nearly all other distractions have been eliminated.*

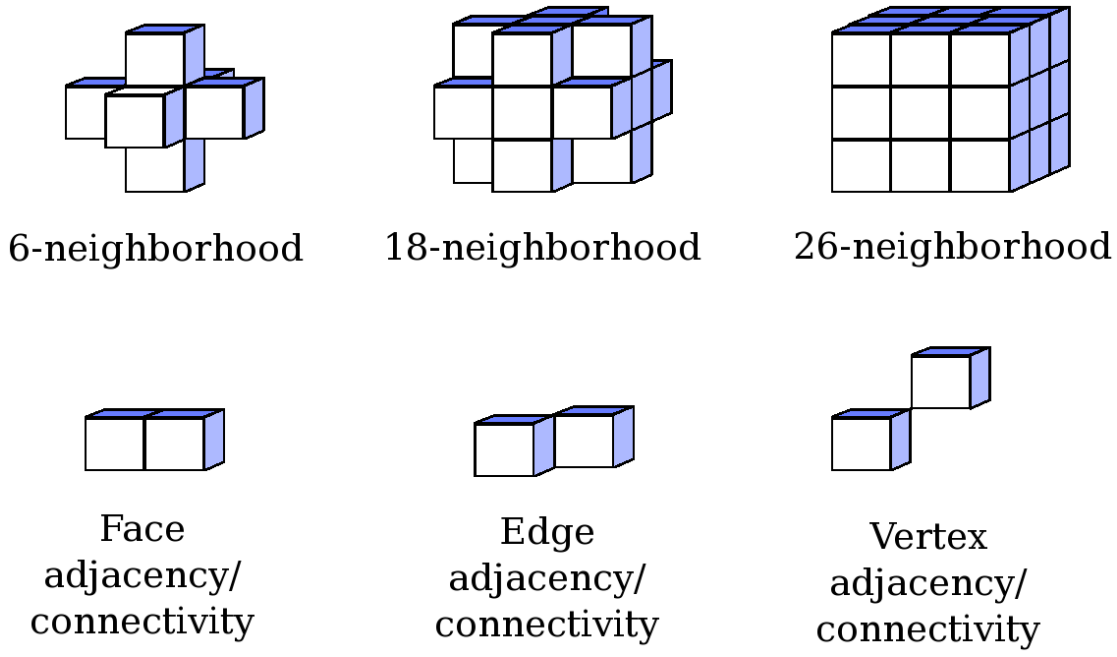


Figure 3.9: *The three common neighboring relations used with voxel datasets. The simplest, the 6-neighborhood, simply consists of voxels which share a face with the voxel in question, or in other words has only one coordinate which differs by one. The 18-neighborhood consists of voxels which share either a face or an edge, or differ by one in one or two coordinates. The 26-neighborhood consists of voxels which share a face, an edge, or a point; they can differ by one in any or all three coordinates.*

the submitted interior point to lie on the topmost slice, and then by performing a two-pass flood fill.

In the first pass, the flood fill is performed slice-by-slice, from smallest to largest z. To begin, the topmost slice is filled using the user-inputted point as a seed. Then, we iterate through the rest of the slices. Each slice's flood fill is seeded using those points whose neighbors on the previous slice were part of the connected component. The result is a subset of the actual connected component, because this pass cannot follow an artery in the upwards direction.

The second pass is a more traditional iterative flood fill. Because of the extremely large number of samples, a recursive algorithm is not appropriate. Instead, we pass over the entire voxel array repeatedly, expanding on the connected component, until a pass occurs in which no changes are made. The results can be seen in Figure 3.8.

3.5 Edge detection

At this point in the process we have a segmented voxel dataset. The next step is to perform an edge detection, identifying the voxels at the borders of the volume. This identifies the contours which characterize the shape of the arterial lumen.

There are a great many edge-detection algorithms which have been developed by the machine vision community, primarily for use on 2-D images. These range from extremely simple thresholded spatial derivatives to complex iterative procedures with derivations based in statistical analysis.

These algorithms are intended for use on photographs or other complex input images. We have an advantage in this situation, as our data is more structured than arbitrary photographs. We have already performed a complete binary seg-

mentation, and therefore have a simpler problem to solve – we need only to mark voxels at the borders of the segmented lumen.

This is straightforwardly done by iterating over each lumen voxel's neighbors. If any of them are not within the lumen, it is marked as a border. This process is performed efficiently in linear time. The results are shown in Figure 3.10.

3.6 Distance transformation

3.6.1 Definition

The *distance transformation* is a very useful tool in image processing in general. It transforms a boolean image B into a gray-level image DT. Mathematically, the 3-D distance transform DT is a scalar field which satisfies the Eikonal equation:

$$|\nabla DT| = 1 \tag{3.2}$$

With the boundary conditions:

$$B(i,j,k) = 1 \implies DT(i,j,k) = 0 \tag{3.3}$$

B is generally an image where the majority is marked as false, and a few special image elements are marked as true. These true elements are called features. Essentially, the value of each voxel in DT contains the answer to the question, "How far from this image element is the nearest feature?"

3.6.2 Efficient computation

As the distance transformation is a global operation, it is generally quite resource-intensive to compute. A naïve implementation runs in time quadratic in the num-

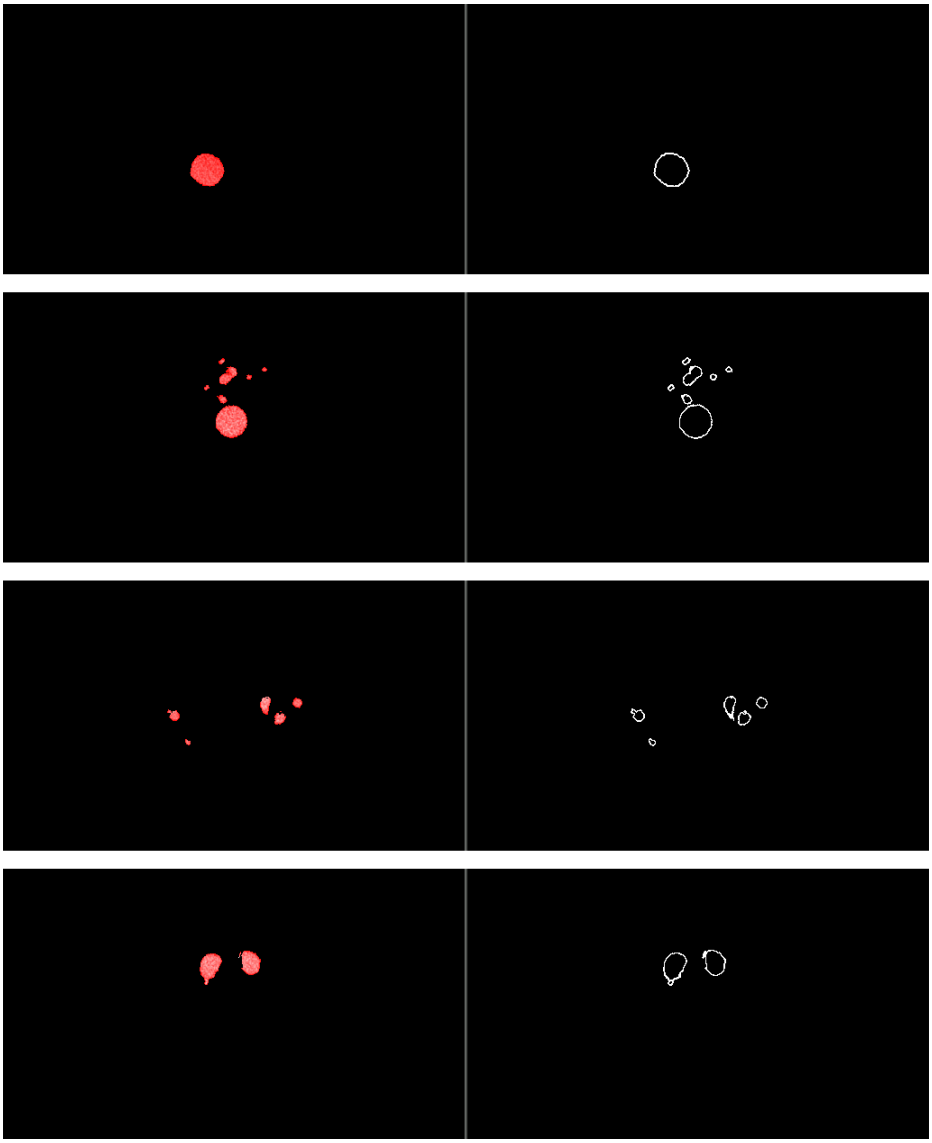


Figure 3.10: *Shown here are slices from the connected-component dataset adjacent to the corresponding slices of the edge-detected dataset. Notice that the borders of the lumen are well marked with a minimum of noise.*

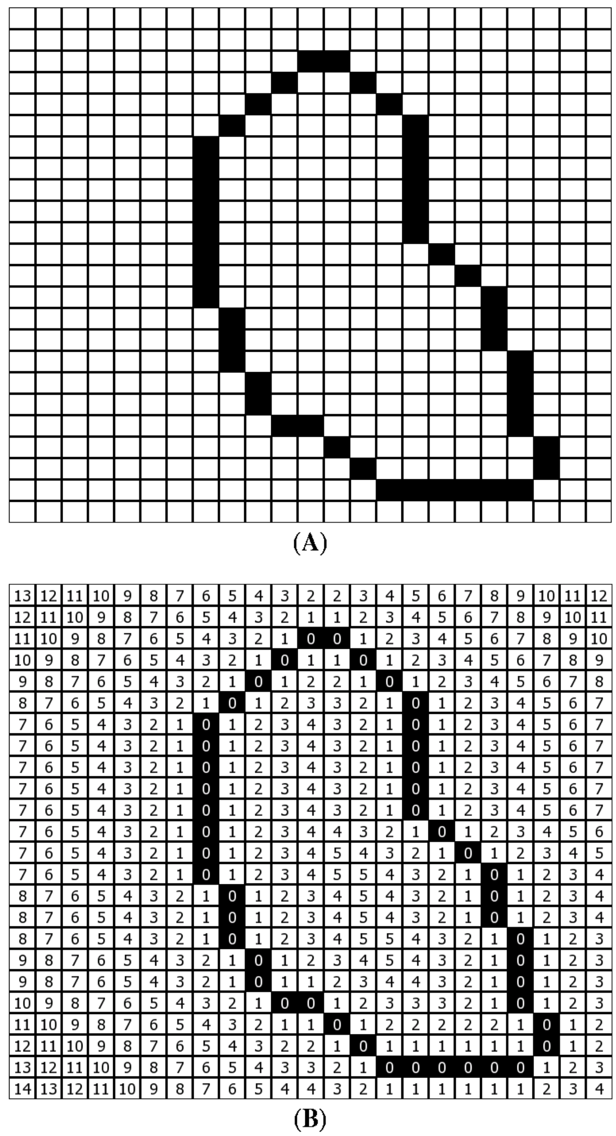


Figure 3.11: *The distance transformation labels every voxel with a number indicating the distance to the nearest feature. In (A) we see an example of an image with features marked in black. In (B) we see the distance transformation overlaid. This particular distance transformation was computed using the Manhattan distance metric.*

ber of image elements. This complexity is simply unacceptable in most 3-D cases. In our case, with hundreds of millions of samples, we needed a much more efficient algorithm.

We implemented a multiple-pass distance transformation algorithm. The algorithm is based on the observation that an n-dimensional distance transformation can be efficiently computed from multiple $n-1$-dimensional distance transformations.

For the sake of simplifying the explanation, let us consider the case where a Manhattan distance metric is being used in the computation of a 2-D distance transformation.

We begin with the 2-D image, and compute the distance transformation for each row separately. For every pixel, we then know the distance to the nearest feature on the same row. To determine the distance from a pixel to the nearest feature overall, we simply scan the column. Each pixel in the column represents a distance to a feature – the value of the pixel plus the distance along the column. Simply choosing the minimal sum yields the correct distance transformation value for that pixel.

By extending this concept to 3-D and to Euclidean distances, we can efficiently compute the distance transformation of our edge-detected voxel dataset. With an $m \times n \times p$ array, computational complexity is $O(mnp \times (n+p))$. Assuming a cube array of n elements, distance transform computation is performed in $O(n^{\frac{4}{3}})$ time.

3.7 Vector gradient field

The next step of the centerline extraction process is very straightforward: the gradient field of the distance transformation is computed.

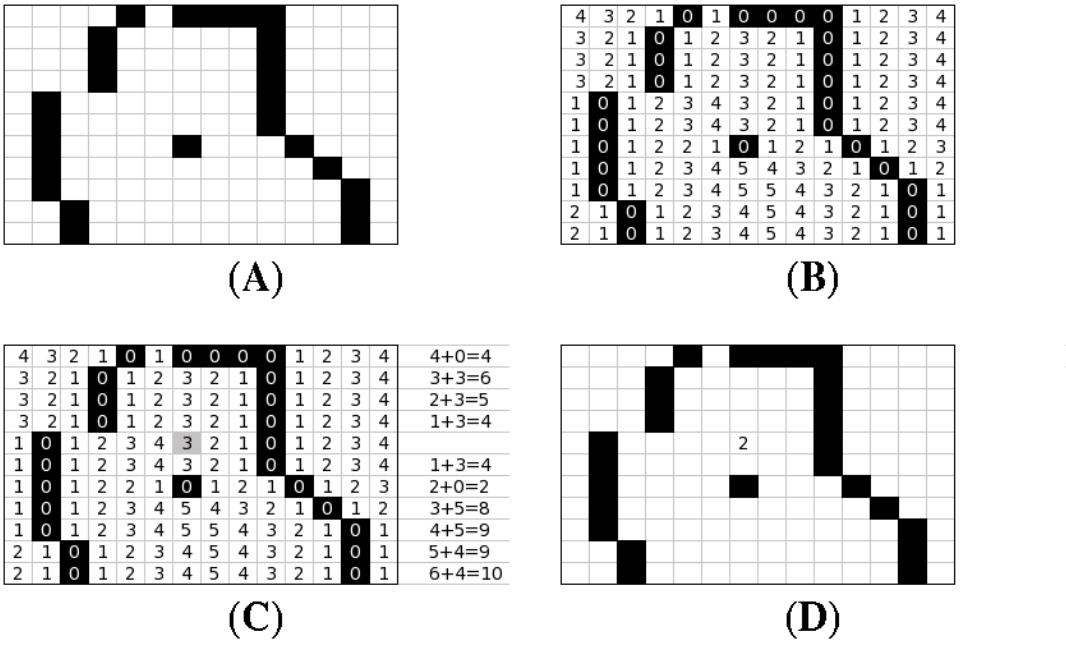


Figure 3.12: *An example of multiple-pass Manhattan distance transformation computation. In (A) we see the original input image. Features are marked with black. In (B) we see a 1-dimensional distance transformation, computed only horizontally. This can be computed very efficiently by simply scanning each row once. In (C) we compute the 2-D distance transformation value for a particular pixel marked with gray. Every element in the pixel's column is considered; the potential distance transformation value is the sum of the distance to the element being considered and the value of the element being considered. Clearly, the choice with the minimum sum is two elements downwards, with a distance of 2 and a value of 0. Thus, we mark the final distance transform value as 2 in (D). This process is repeated for each pixel. An analogous process computes a 3-D distance transform from multiple 2-D distance transforms.*

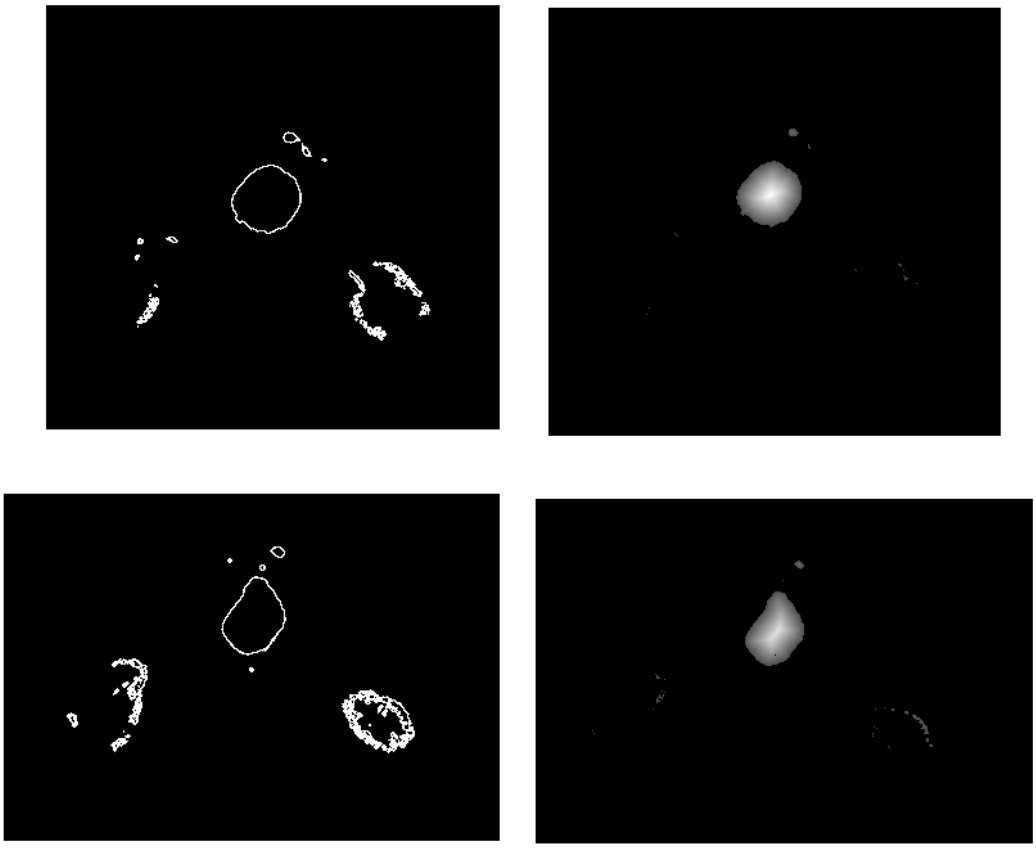


Figure 3.13: *Slices from the edge-detected dataset shown with the corresponding slices of the distance transformation. Values outside the lumen are omitted for clarity. Brighter values indicate higher distance transform values. Note that the intensities in these images are not opacities, but rather gray scales representing values of the distance transformation. Brighter intensities represent greater distances.*

The mathematical definition of the gradient operator ∇ of a 3-D scalar field F is as follows:

$$\nabla F = \frac{\delta F}{\delta x}\hat{x} + \frac{\delta F}{\delta y}\hat{y} + \frac{\delta F}{\delta z}\hat{z} \tag{3.4}$$

Difficulty arises when computing a 3-D numerical gradient, however. In 1-D a numerical derivative is easy to compute:

$$\nabla F(i) = \frac{\delta F}{\delta x}\hat{x} \simeq F(i+1) - F(i) \tag{3.5}$$

This computes the 1-D derivative at the boundaries between samples, since it is symmetric about the point $i + \frac{1}{2}$.

In 2-D, neighborwise differences can only yield such symmetry about the corners of samples, so the 2-D derivative must be computed along a coordinate system rotated 45 degrees relative to the original one, as shown in Figure 3.14.

In three dimensions, there is no such convenient transformed coordinate system that allows the gradient to be computed neighborwise while maintaining symmetry. For this reason, we chose to sacrifice the neighborwise computation in favor of maintaining symmetry. The 3-D gradient is approximated as:

$$\begin{aligned} \nabla F(i,j,k) \simeq & \\ & (F(i+1,j,k) - F(i-1,j,k))\hat{x} + \\ & (F(i,j+1,k) - F(i,j-1,k))\hat{y} + \\ & (F(i,j,k+1) - F(i,j,k-1))\hat{z} \end{aligned} \tag{3.6}$$

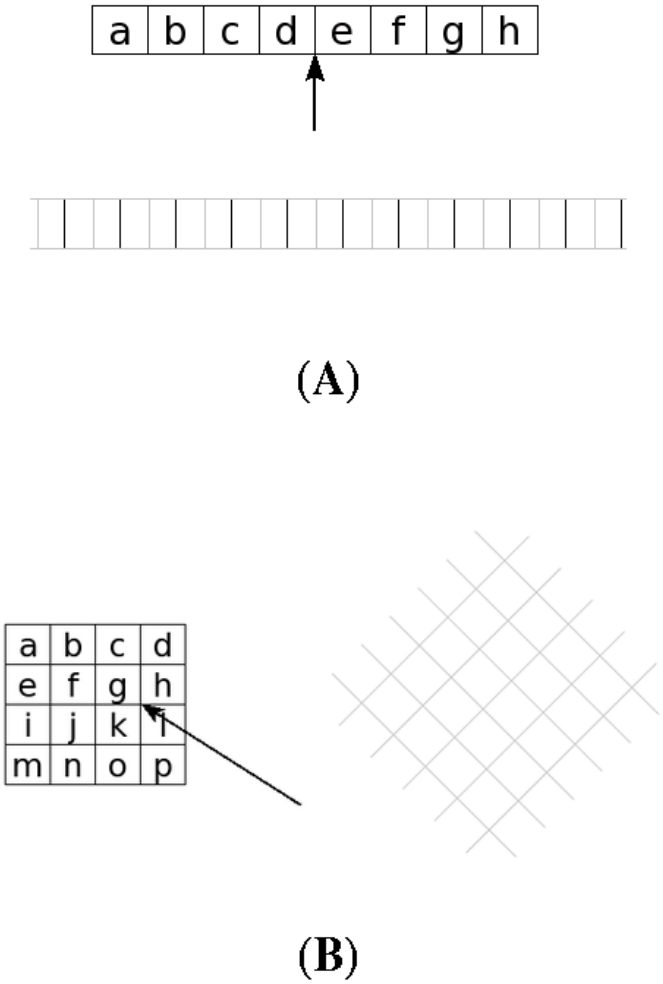


Figure 3.14: *The 1-D gradient at the point indicated in (A) can be taken to be the value $e - d$. The gradient is thus measured between points, and so is measured along a coordinate system translated half a sample relative to the original dataset. The 2-D gradient measured at the point indicated in (B) can be taken to be the value $(l - g)\hat{u} + (h - k)\hat{v}$, where $\hat{u}$ and $\hat{v}$ are unit vectors forming an orthonormal basis for a coordinate system 45 degrees rotated relative to the original dataset.*

3.8 Scalar derivative field

We now wish to compute a derivative field of this vector gradient field. This derivative computation is a means to the end of identifying voxels where the gradient of the distance transform changes rapidly, as these voxels comprise the centerline.

There are several standard methods for differentiating a vector field However, our case is special due to the structure of the vector field. The distance transform satisfies the Eikonal equation, which stipulates that the gradient must always be of unit magnitude. Our vector field is thus rigidly structured; neighboring vectors can differ only in direction.

Our problem is thus reduced to the identifications of sharp changes in direction of the vector field. This allows us to conveniently represent the derivative field as a scalar field, permitting efficient storage and visualization.

Noting that the dot product of two unit vectors is equal to the cosine of the angle between them, we compute this scalar derivative of the vector field using neighborwise dot products. This novel approach of using the dot product as a scalar derivative of a unit vector field gives us quite good results.

3.9 Centerline identification

At this point we have a scalar derivative field, represented as a voxel array of the same dimensions as the original input. Let us revisit what each processing step computes.

The first two steps are very straightforward. The segmentation step simply eliminated distractions from the dataset, leaving only the arterial lumen. The edge detection step identified the borders of the lumen.

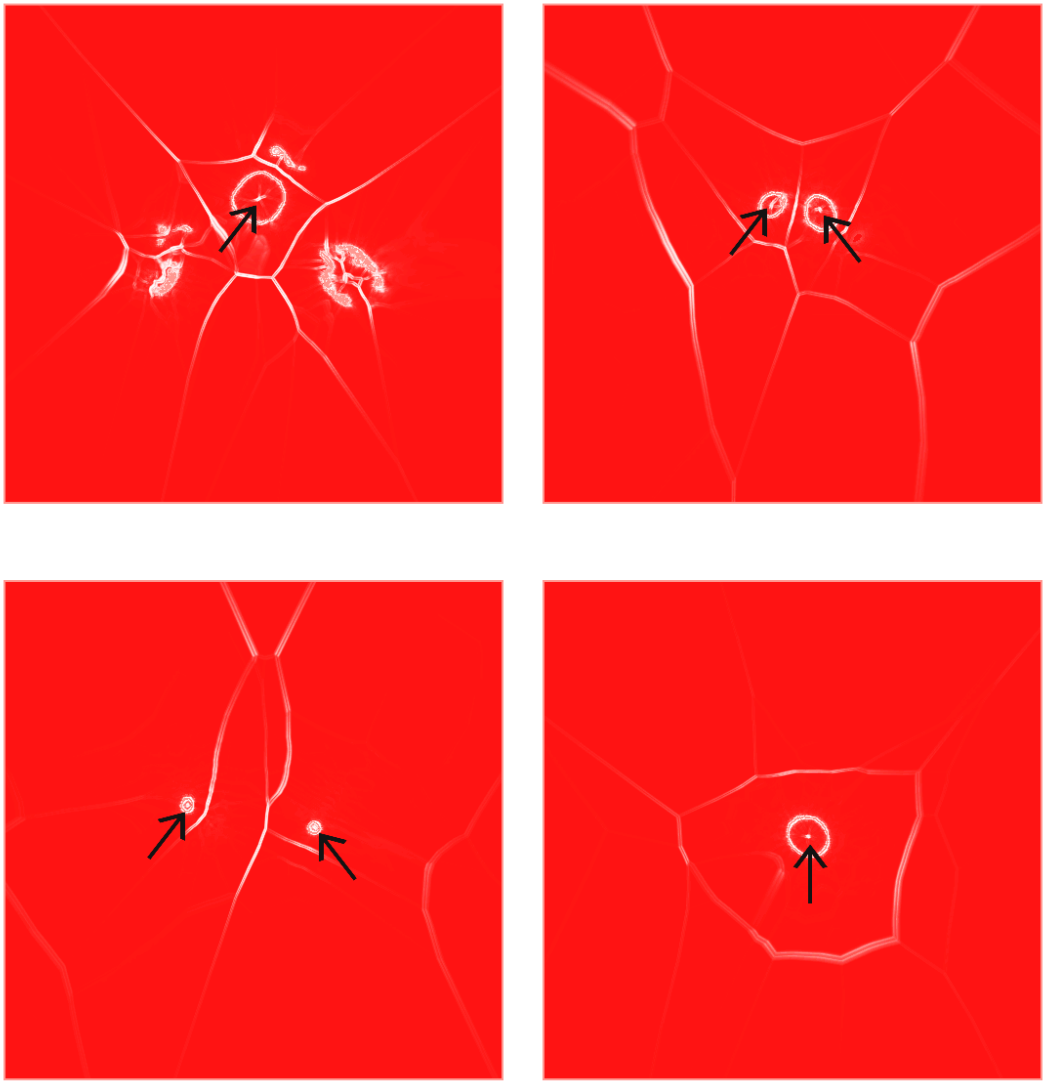


Figure 3.15: *Here we see the scalar derivative of the gradient field. Four slices from the dataset are shown. Lighter shades indicate a higher derivative value. Notice that the center of the arterial lumen is clearly marked with a high derivative.*

The remaining steps' results carry an unobvious significance. The distance transformation encodes how close each voxel was to the wall of the lumen. The gradient of this field, then, encodes the direction of steepest ascent, which represents the direction from any voxel to the nearest portion of the lumen wall. The scalar derivative field measures heterogeneity of this direction, which is high only at the borders and at the centerline.

From the results of the edge detection step, we already have records of the locations of the borders of the arterial lumen. Thus, a very simple thresholding procedure performed on the scalar derivative field, followed by a boolean subtraction of the borders, yields an accurate reconstruction of the arterial centerline, as shown in Figure 3.17.

3.10 Summary

We have developed an efficient centerline extraction procedure. It is highly automated, requiring minimal user input and a relatively low level of user expertise. The user input is collected prior to performing the computationally intensive steps, so the process can be easily run in the background after data collection. The end result is an accurate centerline well-suited for analytical fitting.

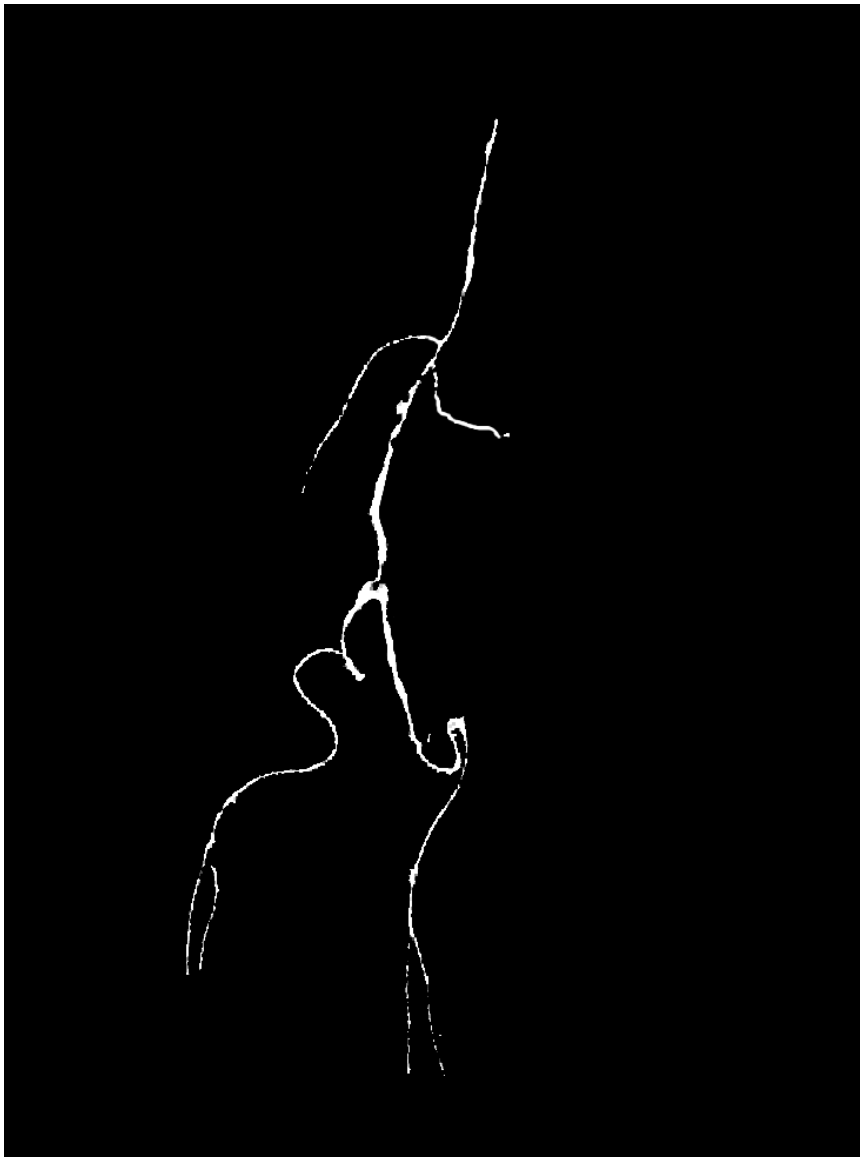


Figure 3.16: *A simple thresholding of the scalar derivative field yields an accurate reconstruction of the arterial centerline.*

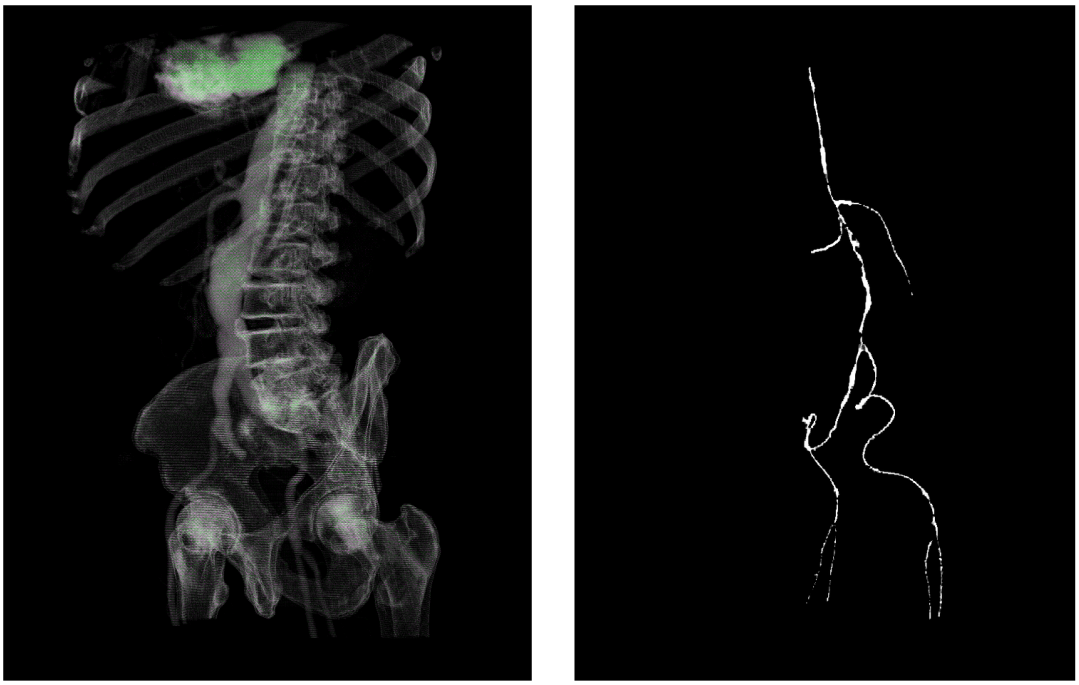


Figure 3.17: *The results of our highly automated centerline extraction procedure. In (A) we see the original contrast dataset, and in (B) we see the identified arterial centerline.*

Chapter 4

Analytical fitting

At this point in our data analysis, we have extracted the arterial centerline from abdominal CT scan data. However, our overall goal is to determine the complete geometry of the aortic lumen, including its surface and that of major branches. In order to make this information as useful as possible, we fit the lumen to an analytical model. Such a model compactly represents a large amount of geometry, and can be discretized with arbitrary precision for the purposes of rendering and simulation.

This chapter presents the analytical fitting processes. It first presents a brief overview of parametric curve and surface representation. It will then detail the procedure for fitting an analytical model to the identified centerline from the previous chapter. It concludes with a description of our surface-fitting procedure which yields us an efficient and useful model of the arterial lumen.

4.1 Centerline fitting

The first step towards an analytical model of the lumen is producing an analytical expression of the lumen's centerline. This can then be used to trace and loft an analytical surface.

The centerline data which results from our final centerline identification step is in the form of a voxel array. Voxels which make up the arterial centerline are marked as such. One desirable property which this centerline does not exhibit is thinness; an ideal centerline is only a single voxel thick. Due to our centerline extraction algorithm, our centerline is generally several voxels thick. Our centerline

fitting algorithm must take this into account.

Our approach to centerline fitting involves a "marching" procedure. We begin at the top of the aorta and iterate downwards, using a flood-fill type of algorithm. Because of the highly structured narrow form of the centerline voxel dataset, this type of approach reliably iterates along the direction of the lumen, and accurately determines the geometric properties of the centerline.

4.1.1 Centerline marching

The concept behind the centerline marching algorithm is best visualized as a propagating wave front. The front begins in the topmost slice and at every iteration propagates along the centerline. As it travels down the centerline, it clearly maintains a cohesive structure. In fact, the structure of the advancing wave front follows the centerline structure closely enough that a split of the front into more than one connected subcomponent happens at the point where the centerline splits due to a branch.

We exploit this predictable behavior of the marching procedure to extract the geometric knots from which we can produce a B-spline model of the centerline. We simply monitor the center-of-mass of the marching wave front. Each recorded center-of-mass is a geometric knot. When the wave front divides, the largest portion is kept and the remaining connected subcomponents begin new branches.

In keeping with the desire for a compact representation, we do not use every recorded center-of-mass in subsequent computations; such a model would be little more than a numerical record. Instead, we select key knots periodically which are then used in the remainder of the fitting procedure. This selection of key knots is adjusted if it causes difficulties, but it is always a relatively small amount of

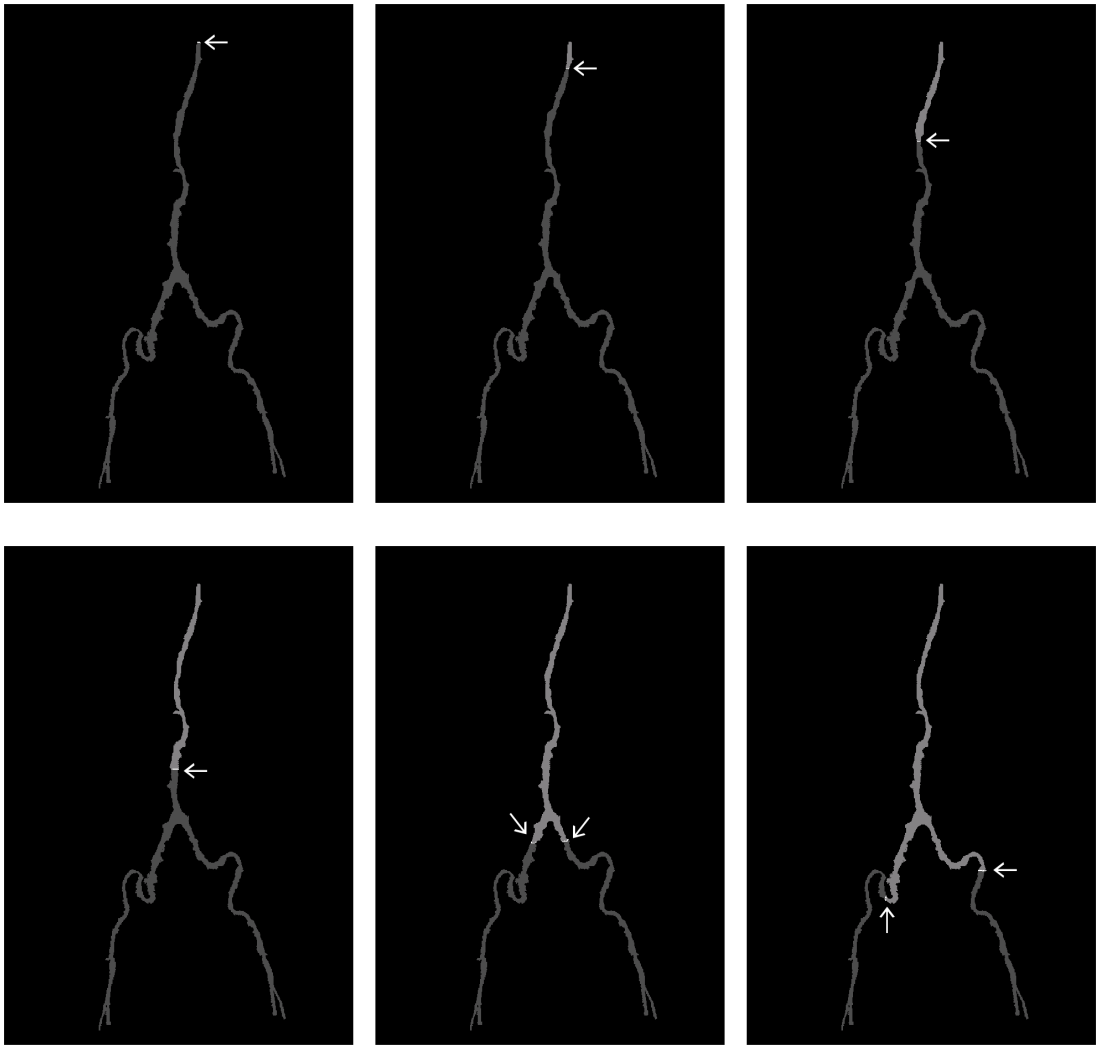


Figure 4.1: *The wave front during centerline marching clearly follows the structure of the centerline.*

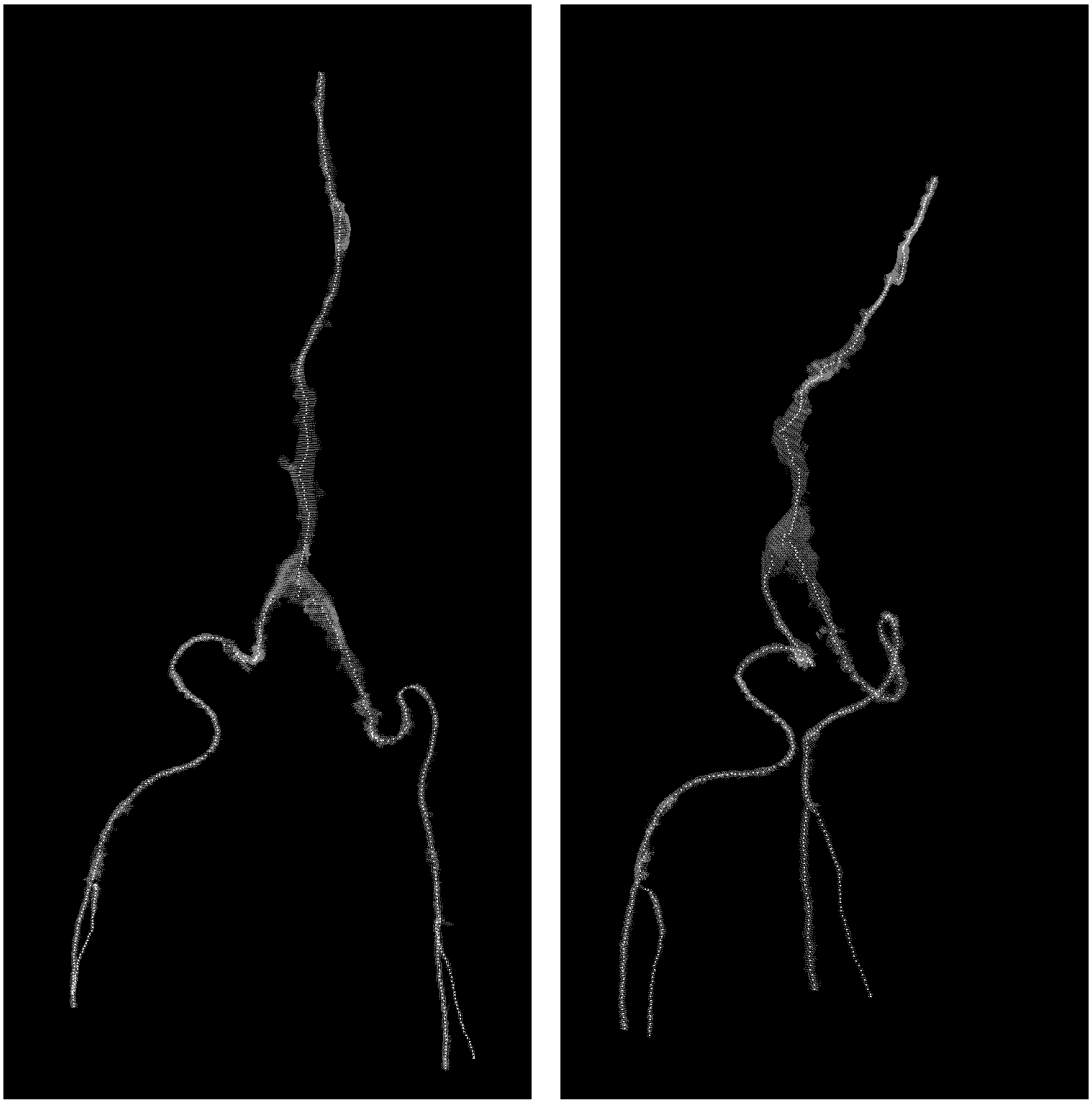


Figure 4.2: *By monitoring the center-of-mass (indicated with white dots) of the advancing wavefront, we derive a series of geometric knots around which to build a spline-based centerline model.*

information.

4.1.2 Spline fitting

We have chosen to use uniform cubic B-splines to represent the arterial geometry. This type of analytical curve adequately satisfies our requirements of conciseness and precision. Mathematical details and characteristics of B-splines are described in Appendix A.

Fitting an existing set of geometric knots to a B-spline curve is not as simple as applying the B-spline basis functions to the geometric knots as if they were control points. A B-spline does not function as an interpolant when used this way. While the knot vector of a nonperiodic B-spline will ensure the curve passes through the first and last control points, in general the curve will not pass through any other control points. For an example, see Figure 4.3.

To fit a B-spline to a set of existing geometric knots, we must solve for a series of control points that generate a curve passing through the geometric knots. We implement a B-spline curve-fitting algorithm based on the observation that the curve's location is always a linear combination of the relevant control points. Thus, if we apply constraints specifying the value of the parameter u at each geometric knot, we can write out an easily solved linear system relating the control points to the geometric knots. This fitting process is described in more detail in Appendix B.

The resulting analytical centerline is shown in Figure 4.4, overlaid on the original data from the intensity-cropped arterial scan. This analytical centerline provides a good basis for the construction of an analytical surface model.

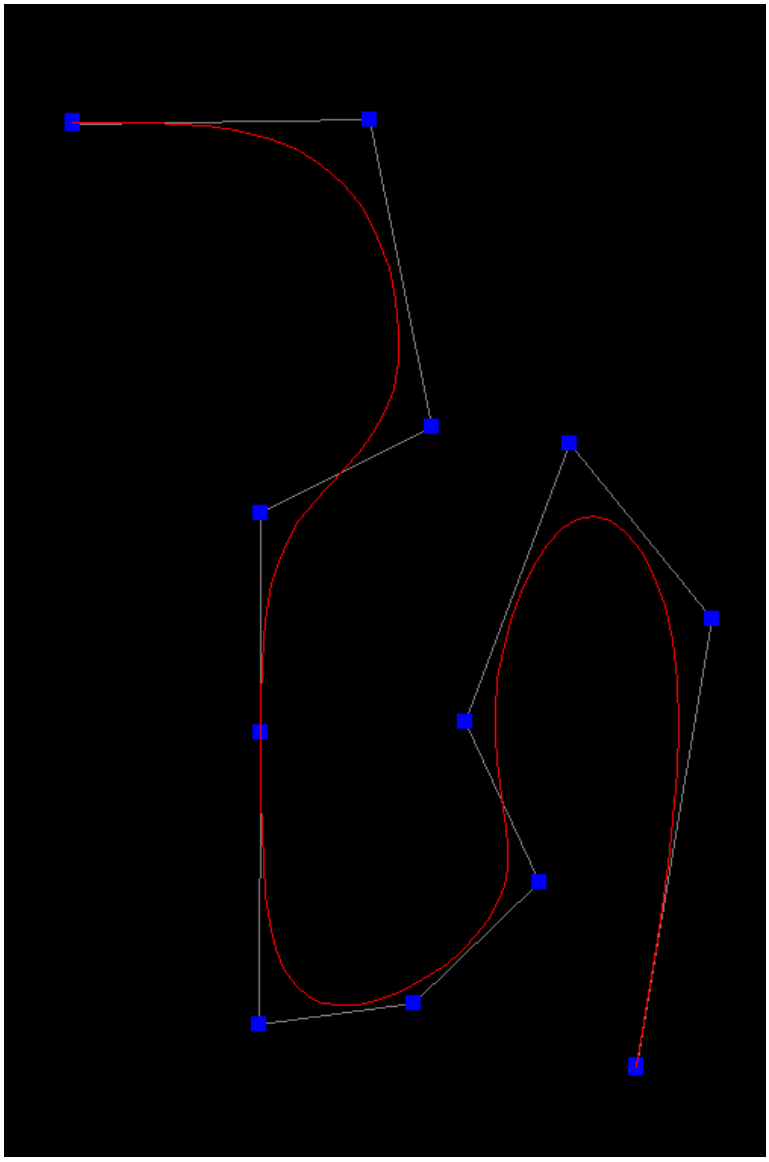


Figure 4.3: *This nonperiodic cubic B-spline does not pass through any control points (indicated by squares) aside from the first and last. This means that one cannot use geometric knots as control points and hope for an adequate representation of the source shape. Instead, one must solve for a set of control points which yields a spline of the correct shape.*

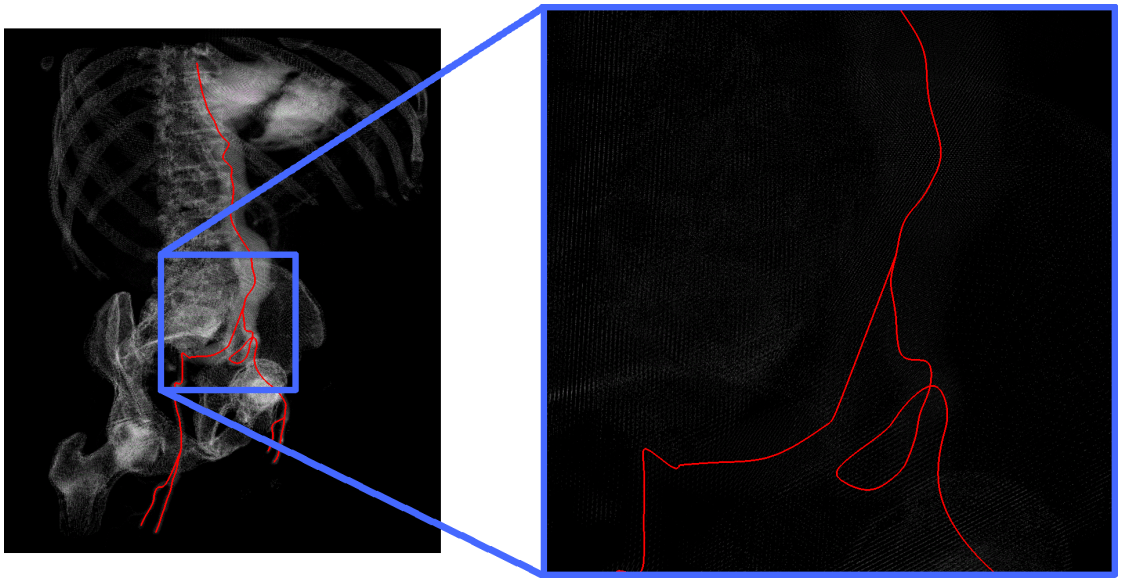


Figure 4.4: *B-spline curves fitted to the centerline, displayed against the intensity-cropped arterial dataset. Notice that the curves closely follow the arterial lumen. Even when examined closely as on the right, because of the spline representation, the curve is smooth, as would actually occur in the real arterial structure.*

4.1.3 Branches

Each branch of the arterial lumen is represented as a separate list of recorded centroids, forming a separate set of geometric knots. To ensure that the centerline of the entire lumen is representative of the correct arterial geometry, it is important that the centerlines of all branches form a single continuous structure without gaps.

In order to achieve this continuity, we use a simple procedure to "stitch" child branches to their parents. Each branch (except for the first) is given one additional geometric knot placed ahead of the rest. This knot is selected from the parent's centroid list.

The stitching procedure selects the additional knot based on two parameters: the distance of the potential knot from the head of the branch, and the direction from the head of the branch to the potential knot. The first parameter is fairly clear; one would not want to add a knot which is far from the rest of the branch.

The second parameter is intended to avoid the situation where the new knot drastically changes the final geometry of the branch. It is encoded as the dot product of the vector from the head of the branch to the new knot with the initial tangent vector of the branch.

These two parameters are balanced against one another using an empirically-determined regularization parameter. See Figure 4.5 for an example.

4.2 Surface fitting

While the focus of our work has been on centerline extraction, we also developed a preliminary surface fitting procedure. The goal of this procedure is to produce an analytical representation of the surface of the arterial lumen. Such a surface

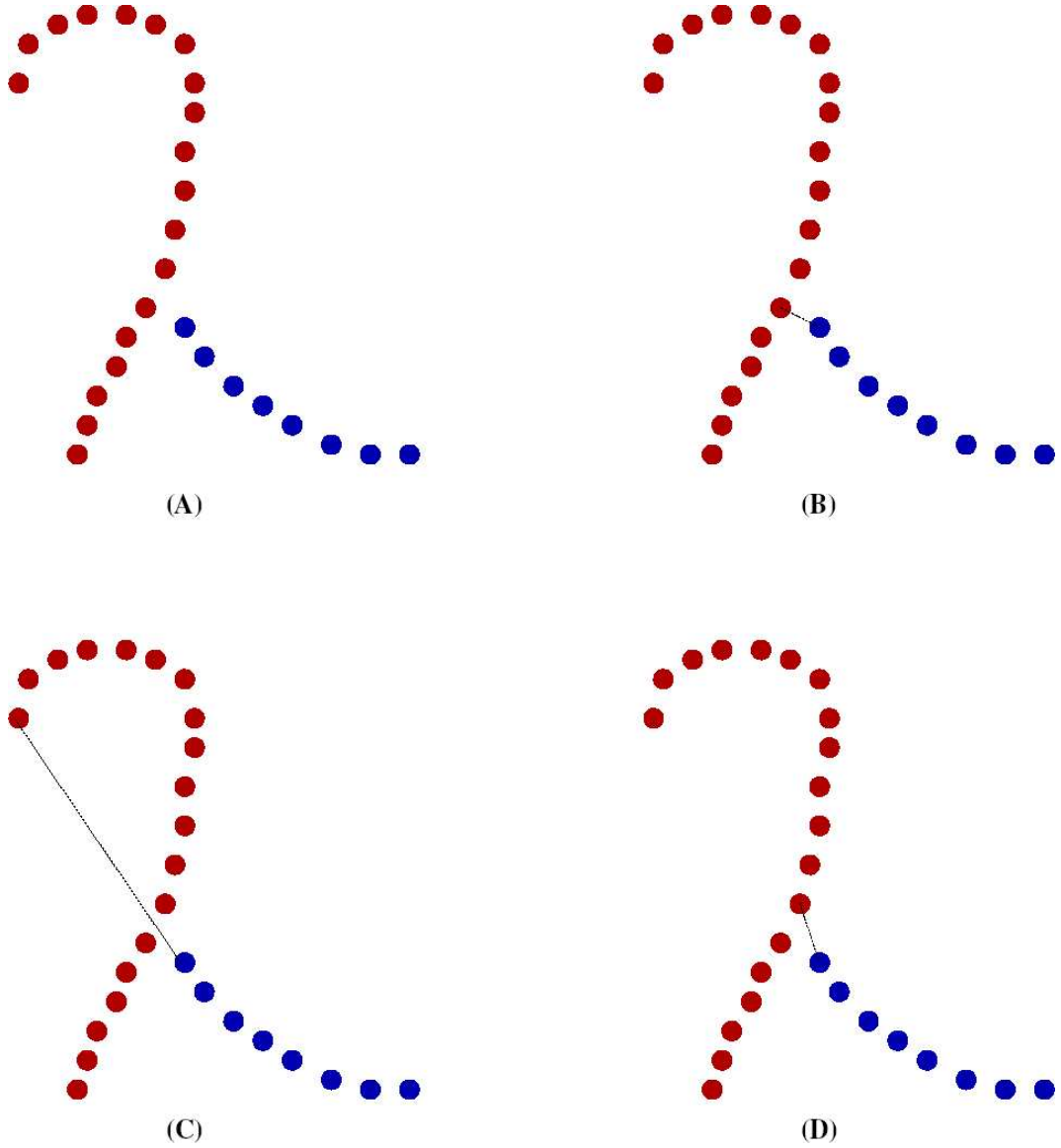


Figure 4.5: *(A) The blue branch needs to be stitched to the red branch. A point on the red branch must be chosen to place at the head of the blue branch's list of knots. (B) Selecting the knot based solely on closeness can give the child branch an unnatural bend at the end. (C) Selecting the knot based solely on direction can cause topologically incorrect results. (D) Balancing both requirements yields an appropriate joint.*

model can be of great utility in applications involving visualization, simulation, and prosthetic design.

Our surface fitting procedure leverages the accurate analytical centerline to compute an analytical surface appropriate for the topology of the lumen. Each arterial branch is represented as a lofted B-spline surface. The surface is defined by a series of two-dimensional slices, each encoded as a periodic B-spline. These slices are analogous to a set of geometric knots of a curve; the surface is rendered by interpolating between the slices with B-splines. Again, we employ an inverse-spline algorithm, as described in Appendix B, in order to derive appropriate control geometry to achieve interpolation.

We compute these slices by constructing coordinate systems at key points periodically placed along the centerline. This coordinate system is then used to drive a marching algorithm which locates the surface boundary.

4.2.1 Generating a local orthonormal basis

At each desired slice location, we generate a local orthonormal basis. For a good representation of the surface, the slice would ideally exist in the plane orthogonal to the axis of the lumen. Fortunately, we have the centerline of the lumen in an analytical form, and it is a straightforward matter to generate an orthonormal coordinate system basis that will place the slice orthogonal to the centerline.

We begin with a unit vector tangent to the centerline. This vector is easily determined from the analytical centerline representation. We compute this vector and label it $\hat{w}$.

To compute the remaining two basis vectors, $\hat{u}$ and $\hat{v}$, we first choose an arbitrary reference vector; in our implementation we use $\hat{x} = \langle 1, 0, 0 \rangle$ as this reference

vector. The cross product $\hat{w} \times \hat{x}$ gives us a vector guaranteed to be orthogonal to the centerline; we normalize this vector and label it $\hat{u}$. Then, the cross product $\hat{w} \times \hat{x}$ yields $\hat{v}$. This process is shown in Figure 4.6.

Note that there is a potential weakness in this approach. Always using $\langle 1, 0, 0 \rangle$ as our reference vector has the potential to cause difficulty if the centerline tangent vector $\hat{w}$ is close to parallel or antiparallel to $\langle 1, 0, 0 \rangle$. The magnitude of the cross product is very small. Because of the limits of the floating point processor's registers, this means that the direction is not specified with high precision. If $\hat{w}$ is equal to either $\langle 1, 0, 0 \rangle$ or $\langle -1, 0, 0 \rangle$, the situation is worse; the cross product has a magnitude of zero and hence no direction.

Fortunately, this latter situation is so highly improbable as to never occur in practice. The former situation is encountered occasionally, but the loss in precision is acceptable for our purposes. Moreover, there is an advantage to using the same reference vector when generating all the local bases: it simplifies the lofting procedure because it ensures that the local $\hat{u}$ and $\hat{v}$ vectors would be continuous if computed continuously along the centerline. This makes interpolation between slices easier because each slice's local basis vectors correspond to its neighbors' basis vectors. Figure 4.7 shows the results of the orthonormal basis generation procedure.

4.2.2 Slice marching

Once a local orthonormal basis has been computed, we determine the boundaries of the slice by making use of data collected during the segmentation step of centerline extraction.

Our algorithm simply marches radially outward from the centerline of the lumen

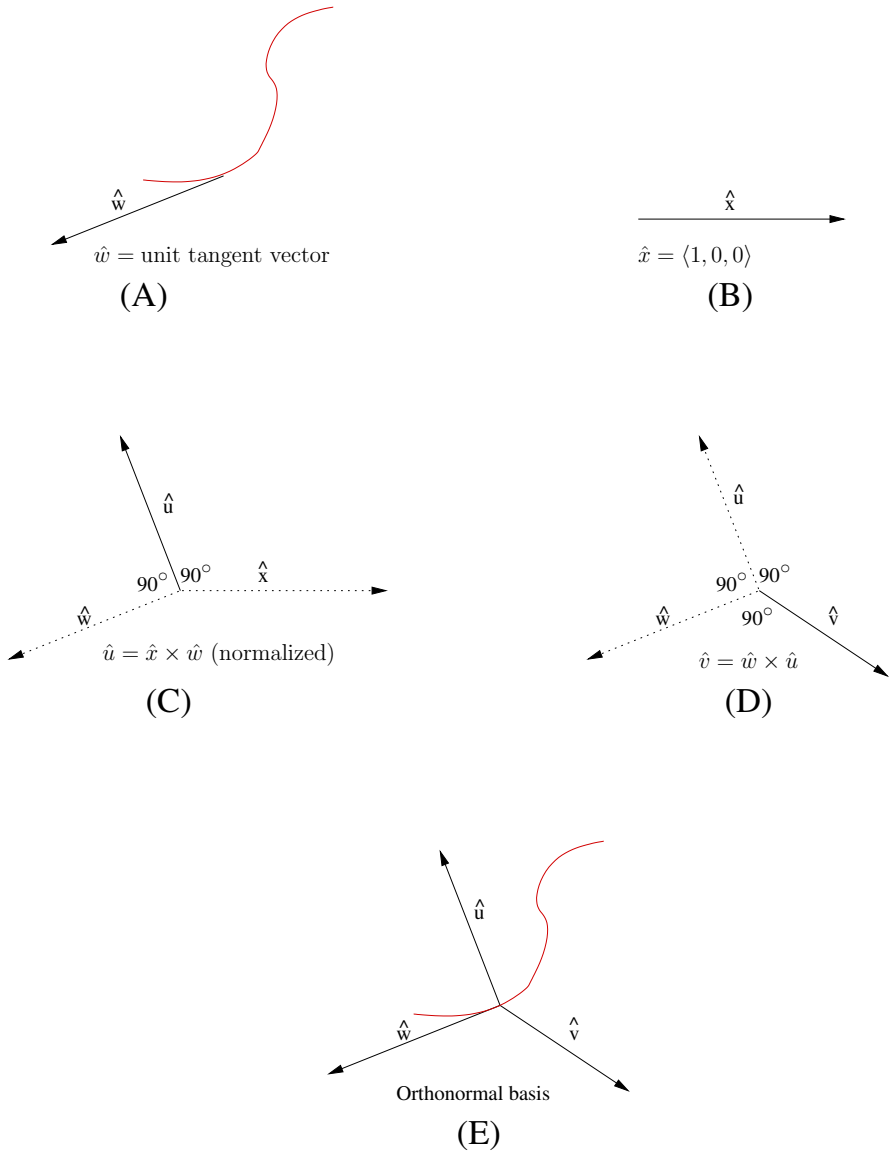
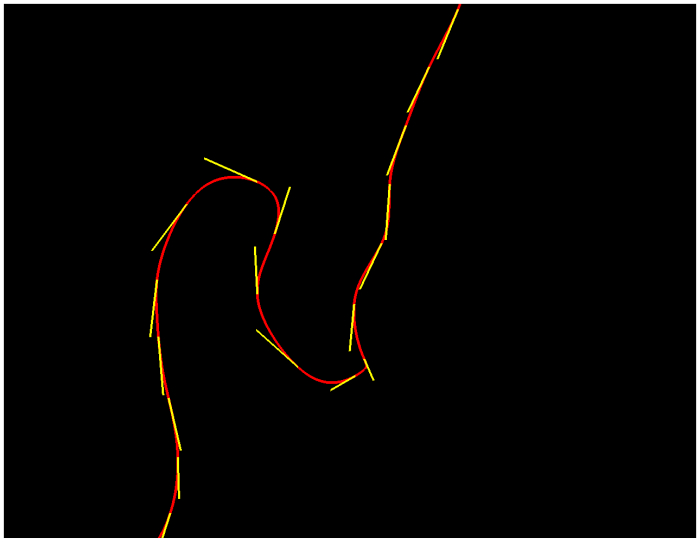
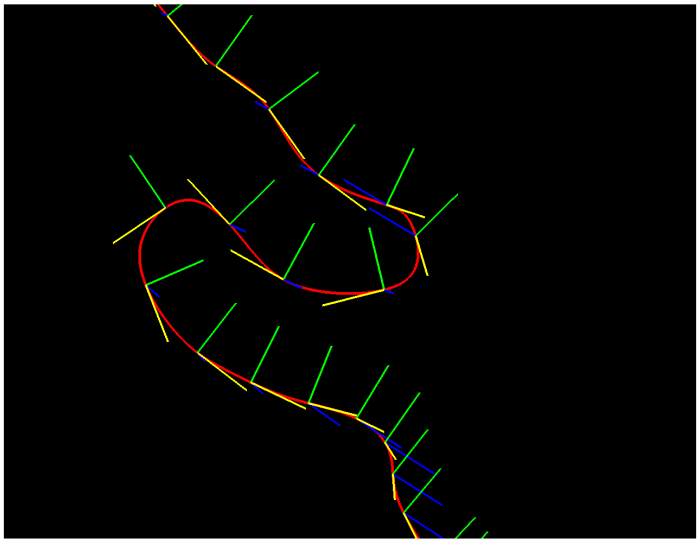


Figure 4.6: *An orthonormal basis is generated around the centerline. (A) A unit vector tangent to the centerline (shown in red) is computed and labeled* $\hat{w}$ *The unit vector in the x direction is chosen as a reference. (C) The cross product of* $\hat{w}$ *and* $\hat{x}$*, once normalized, gives us the local basis vector* $\hat{u}$*. (D) The cross product of* $\hat{w}$ *and* $\hat{u}$ *gives us the local basis vector* $\hat{v}$*. (E) The result is an orthonormal basis consisting of the three vectors* $\hat{u}$*,* $\hat{v}$*, and* $\hat{w}$*.*



(A)



(B)

Figure 4.7: *Orthonormal bases generated along the centerline. In (A) we see the analytical centerline shown in red, with the tangent vectors displayed in yellow. These are used as the local $\hat{w}$ vectors. In (B) we see the orthonormal bases built around these tangent vectors. Blue represents the $\hat{u}$ vector, and green represents the $\hat{v}$ vector. Note that the orientation of the local bases is continuous along the curve. This will simplify the lofting process later on.*

until it reaches the end of the segmented volume. The boundary between the final internal voxel and the first external voxel is then marked as a geometric knot.

This marching step is performed several times in several directions, each evenly spaced around the unit circle. Each direction is chosen as an angle θ, and $\hat{d}(\theta)$, the unit vector in the marching direction, is computed with the simple formula:

$$\hat{d}(\theta) = \hat{u}\cos\theta + \hat{v}\sin\theta \tag{4.1}$$

This direction is expressed as a linear combination of $\hat{u}$ and $\hat{v}$, and so is guaranteed to lie in the plane orthogonal to the centerline. Figure 4.8 shows an example of eight evenly-spaced marching directions.

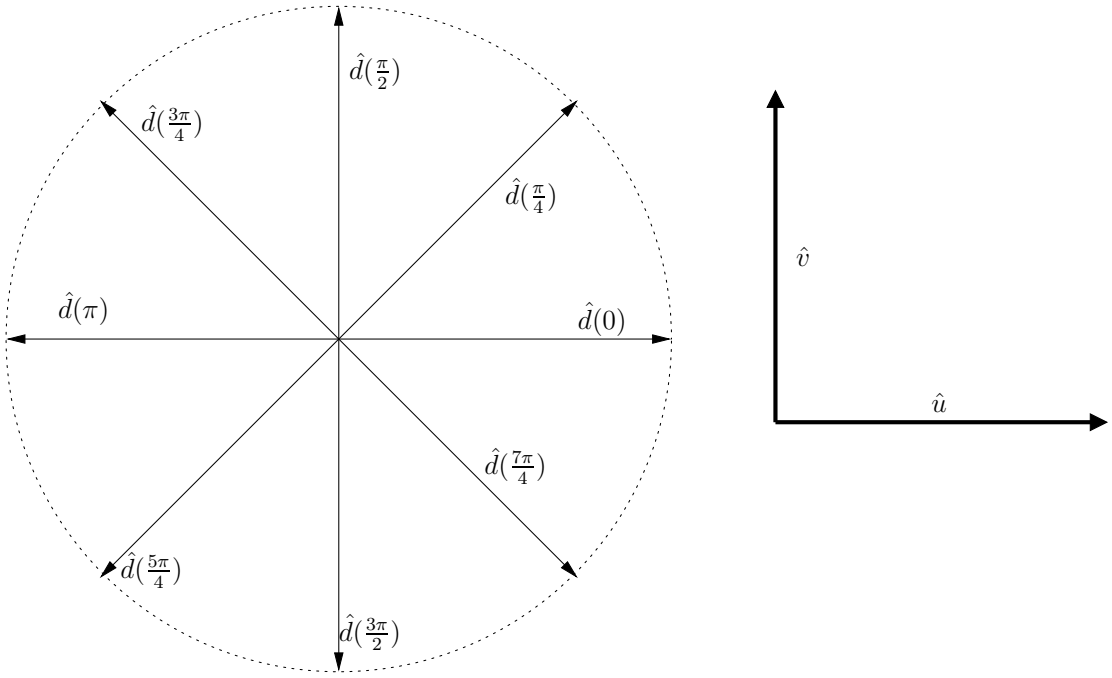


Figure 4.8: *With the given $\hat{u}$ and $\hat{v}$ vectors, the marching direction vector $\hat{d}(\theta)$ is computed for eight values of θ.*

4.2.3 Lofting

With all the slices computed, we have a complete analytical description of the arterial lumen's surface. The slices define the lofted surface. To construct the entire surface, we merely interpolate between these slices using nonperiodic B-splines. The interpolation technique is very similar to that used within each cross-sectional slice.

To evaluate the lofted surface for rendering, each point on each cross-sectional curve is treated as a geometric knot on a longitudinal curve. One such longitudinal curve exists for every value of the section curves' parameter u. The longitudinal curve is one-dimensional in terms of its parameter v, so the result is a two-dimensional surface indexed by parameters u and v. The lofting process is described in more detail in Appendix C. By efficiently solving for the longitudinal B-splines in realtime we can render the surface of the arterial lumen.

4.2.4 Slice placement refinement

When constructing the cross-sectional slices for lofting, the placement of slices affects the geometry of the resulting lofted surface. A sufficiently pathological placement of slices can create a completely unusable surface. The biggest problem arises when the geometry of the lumen is such that two cross-sectional slices intersect, as shown in Figure 4.9. The intersecting slices create a malformed lofted surface which does not correctly represent the arterial geometry.

In order to prevent this problematic situation, we implemented a simple refinement step. To simplify computation, we replace each slice with the smallest disc which fully contains it. We then perform pairwise intersection tests on all pairs of circles, identifying pairs which intersect. One member of each intersecting pair is

selected for recomputation.

To recompute a slice, we first relocate it, selecting a nearby position on the centerline. Then we repeat the local orthonormal basis generation and marching steps with the new slice placement. This process is repeated until all slice intersections are eliminated.

Figure 4.10 demonstrates the effect of this slice placement refinement procedure.

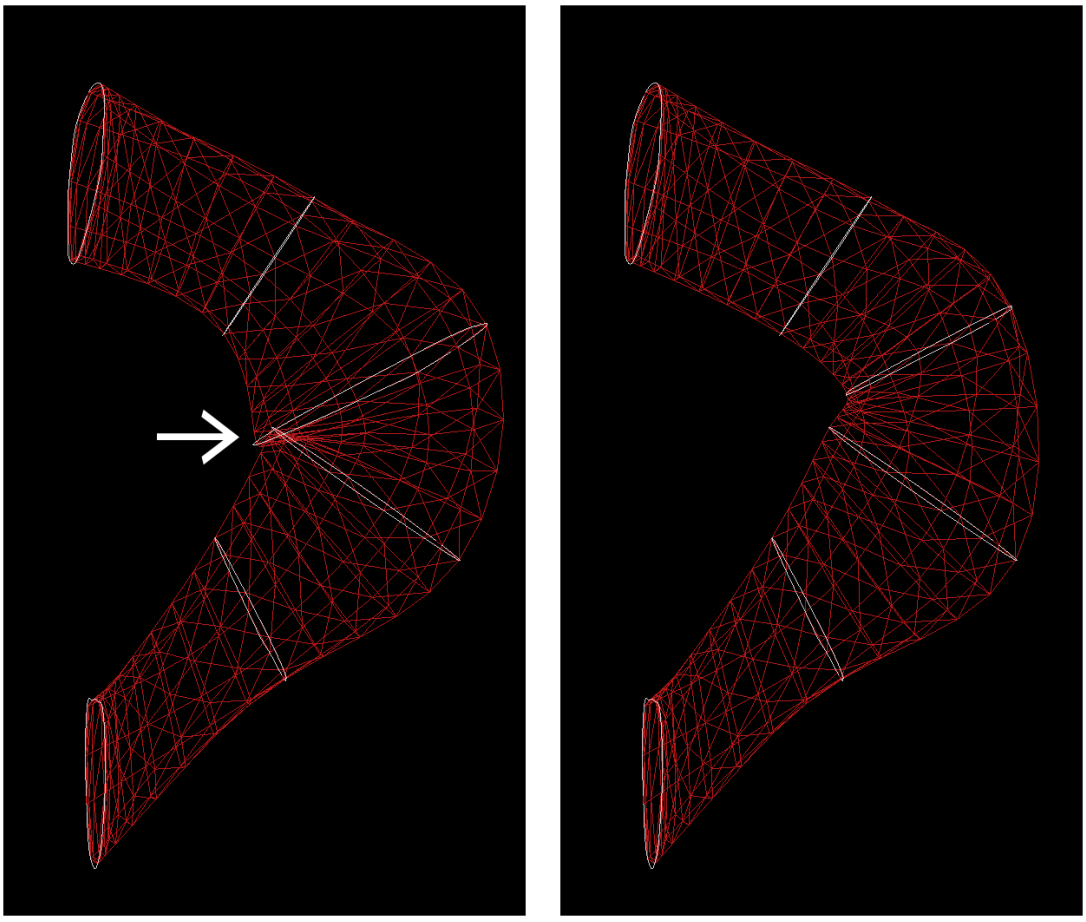


Figure 4.9: *When two cross-sectional slices intersect (A), they create a malformed lofted surface which cannot represent the true shape of the lumen. If slices do not cross (B), the correct shape is represented.*

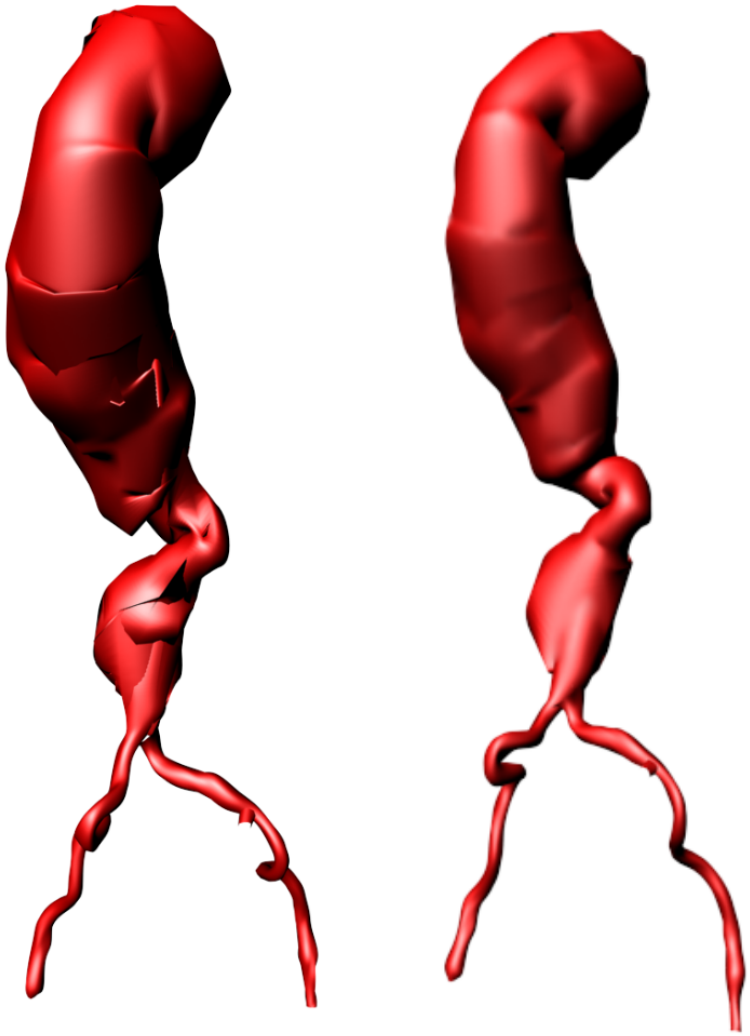


Figure 4.10: *Prior to refining the placement of slices, a tortuous centerline often results in incorrect surface geometry due to intersecting cross-sectional slices, as shown on the left. Slice placement refinement selects slices which do not interfere, allowing the fitted surface to have the correct smooth topology, as shown on the right. These surfaces are rendered with a Blinn-Phong reflection model.*

4.3 Summary

We have presented a robust procedure to fit one-dimensional spline curves to the centerline. The results of the centerline fitting procedure are shown in Figure 4.11. This curve alone is a potentially very useful construct. It provides an accurate way to measure the length of a portion of the patient's anatomy. It also provides a powerful basis for the fitting of a surface model to the arterial lumen. We have exploited this and developed a surface fitting procedure which yields a fully analytical model of the surface of the lumen. As described in the conclusion, these results can then serve as input for the design of the endovascular stents, as well as simulation models for stress analysis and computational fluid dynamics. The geometric results are shown in Figure 4.12.

Figure 4.11: *The final analytical description of the centerline.*

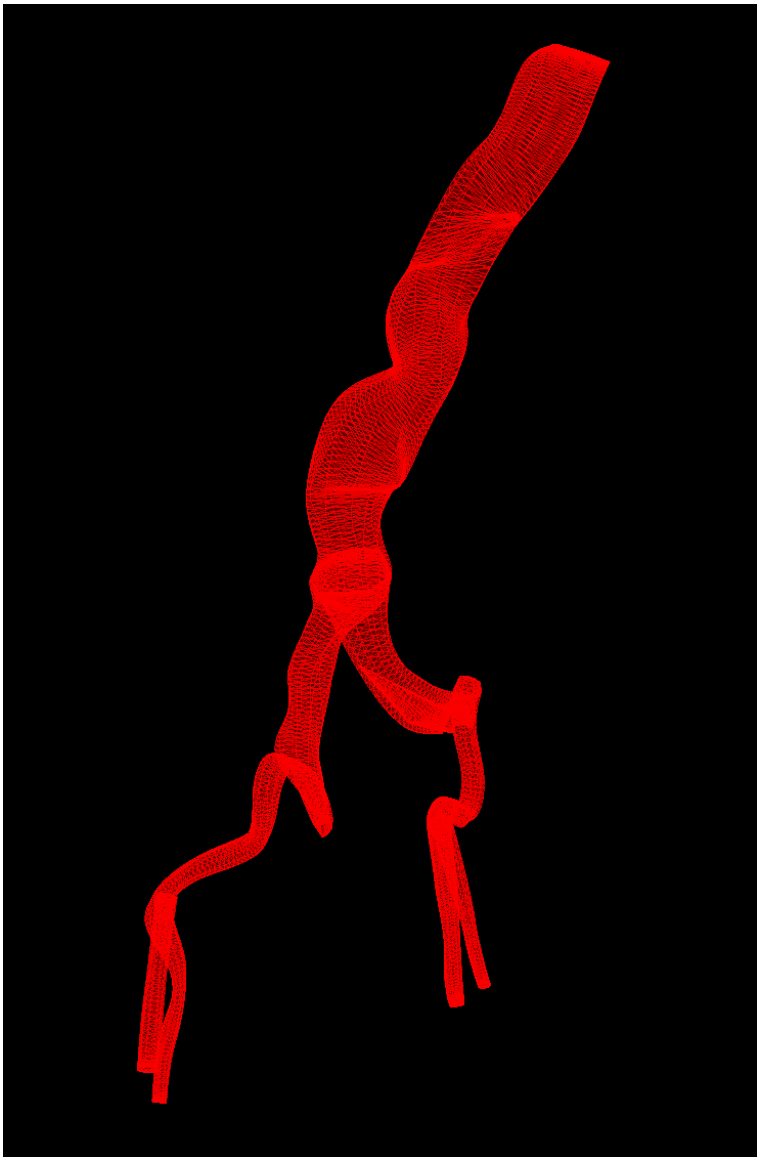


Figure 4.12: *The lofted surface which analytically represents the arterial geometry.*

Chapter 5

Results

In cooperation with Doctor Roy K. Greenberg and his staff at the Cleveland Clinic, we were able to test our centerline extraction and analytical fitting procedures on CT scans performed on actual patients. This chapter will discuss our results and present a summary of the work conducted.

5.1 Results

We tested the system described in this thesis on data from three aortic aneurysm patients. In an anonymizing step, all patient-identifying information is removed from the data; we identify our three datasets simply by a one-letter code name. Here we show results from Patient J, Patient M, and Patient B.

5.1.1 Patient J

In applying our procedures to Patient J's CT scans, we met with a good deal of success. The analytical centerline is well-behaved and closely follows the aorta and its major branches. The centerline is shown superimposed on the intensity-cropped arterial CT scan in Figure 5.1.

The analytical surface is also well-behaved and closely fits the actual geometry of Patient J's arteries. The surface is shown rendered in wireframe in Figure 5.2 and rendered as a surface with a shading model in Figure 5.3.

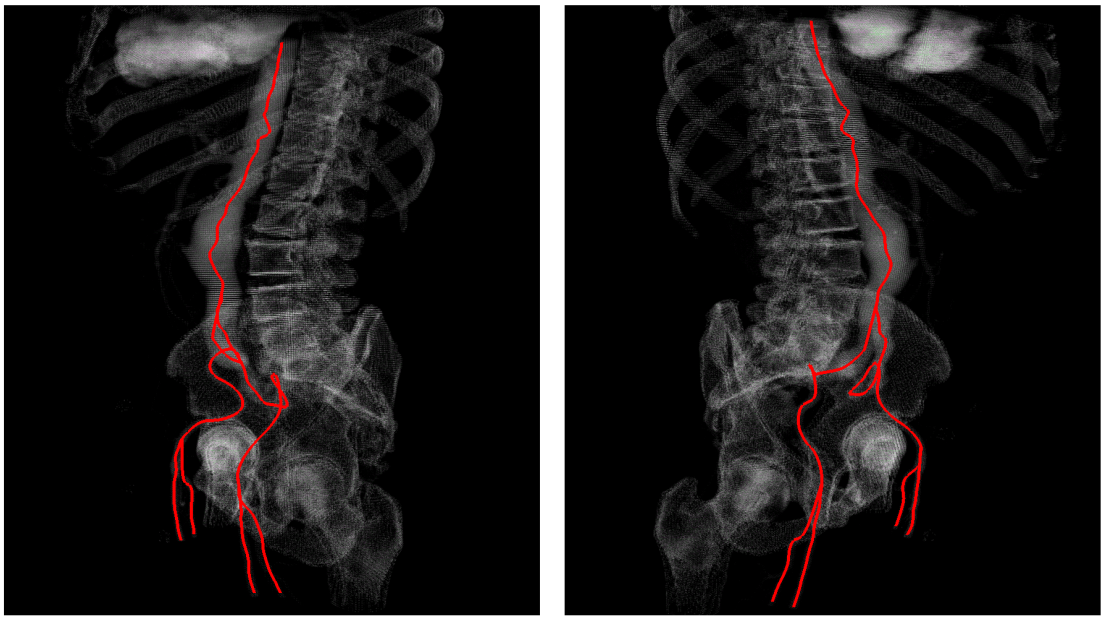


Figure 5.1: *The analytical centerline computed for Patient J, superimposed on the intensity-cropped arterial CT scan.*

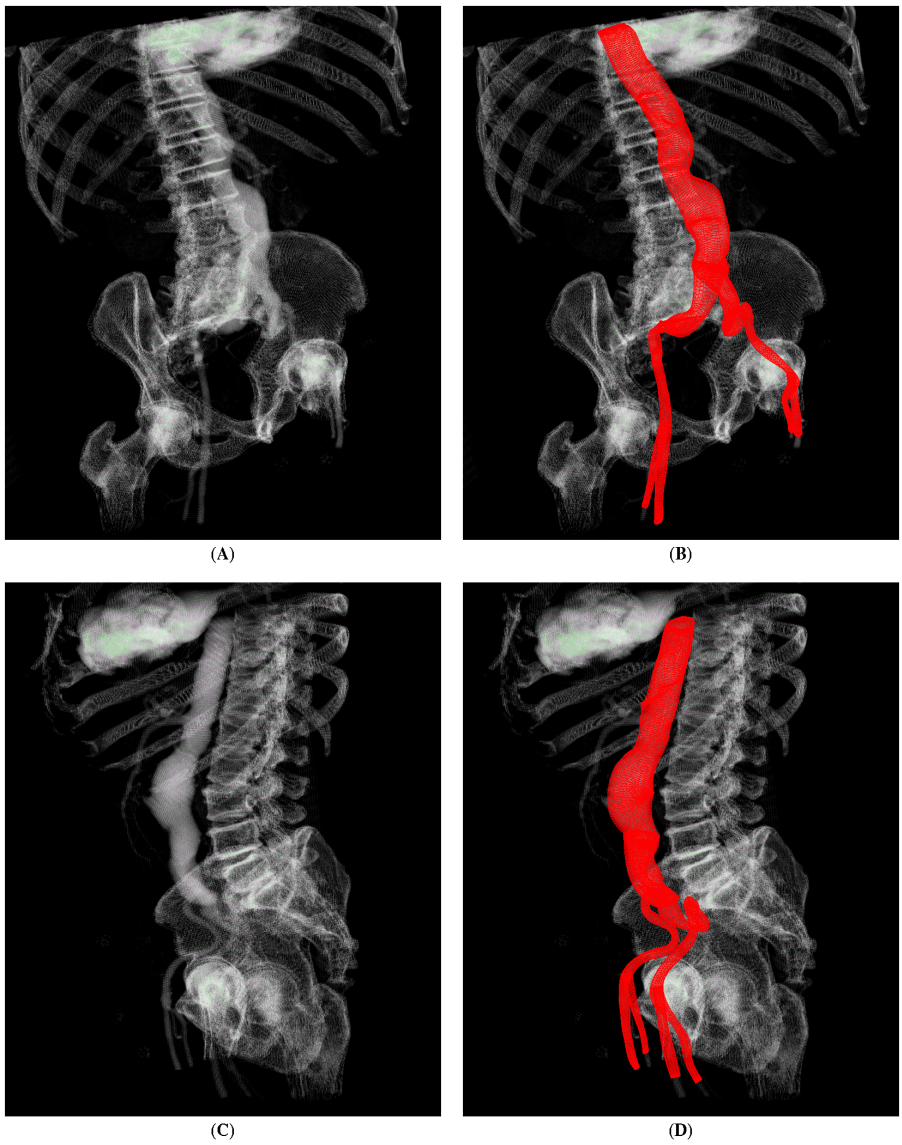


Figure 5.2: *The final analytical surface computed for Patient J. In (A) we see a view of the intensity-cropped arterial scan. In (B) we see the same view with the analytical surface superimposed. In (C) and (D) we see the same from a different viewing angle.*

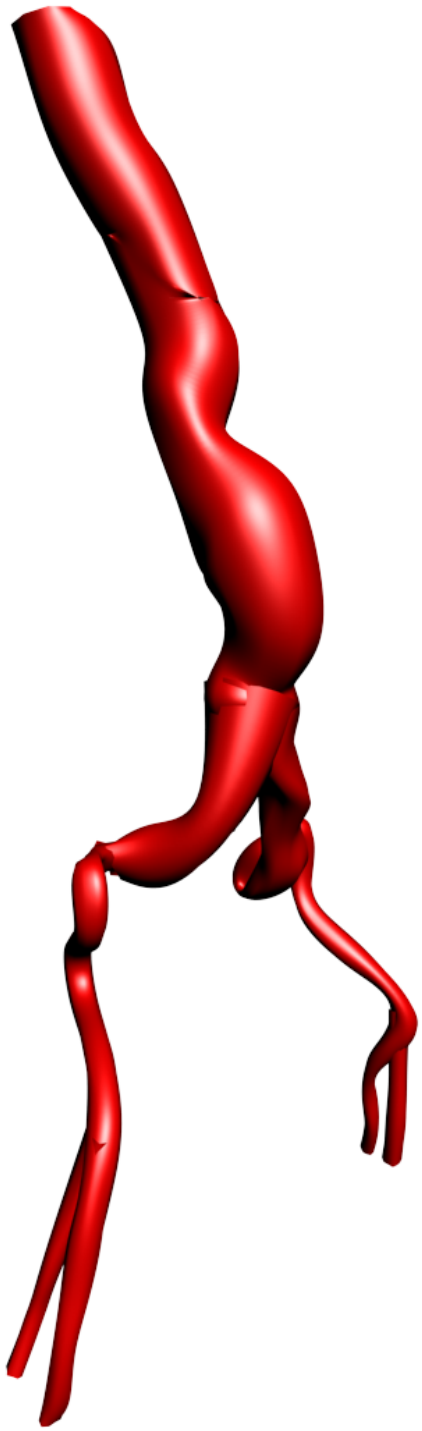


Figure 5.3: *The analytical surface representing Patient J's arterial lumen, rendered with a Blinn-Phong material to show the 3-D geometry.*

5.1.2 Patient B

Our initial results with the data from Patient B were not nearly as encouraging. At the end of the segmentation step, we found that the segmented volume appeared to only contain the superior portion of the aorta. Upon close comparison between the segmented volume and the intensity-cropped arterial scan, we further found that the segmented volume did not contain the complete volume of even the portion of the aorta which it did represent.

The culprit was poor registration. The segmentation step described in Chapter 3 depends on a registration between the native and arterial data sets, and this was difficult to attain. Even the best registration we were able to achieve left a great many mismatched voxels. This meant that the narrow inferior arterial branches simply disappeared during the voxel intersection processing, as the disparity between the two scans was larger than the blood vessels themselves.

In order to continue with our testing, we performed a segmentation of the intensity-cropped arterial scan by hand, which we then used in place of the intersected volume. To this we applied the remainder of our system, beginning with connected-component computation and proceeding all the way through analytical surface fitting. Our success was similar to that we had with data from Patient J. The results are shown in Figures 5.4, 5.5 and 5.6.

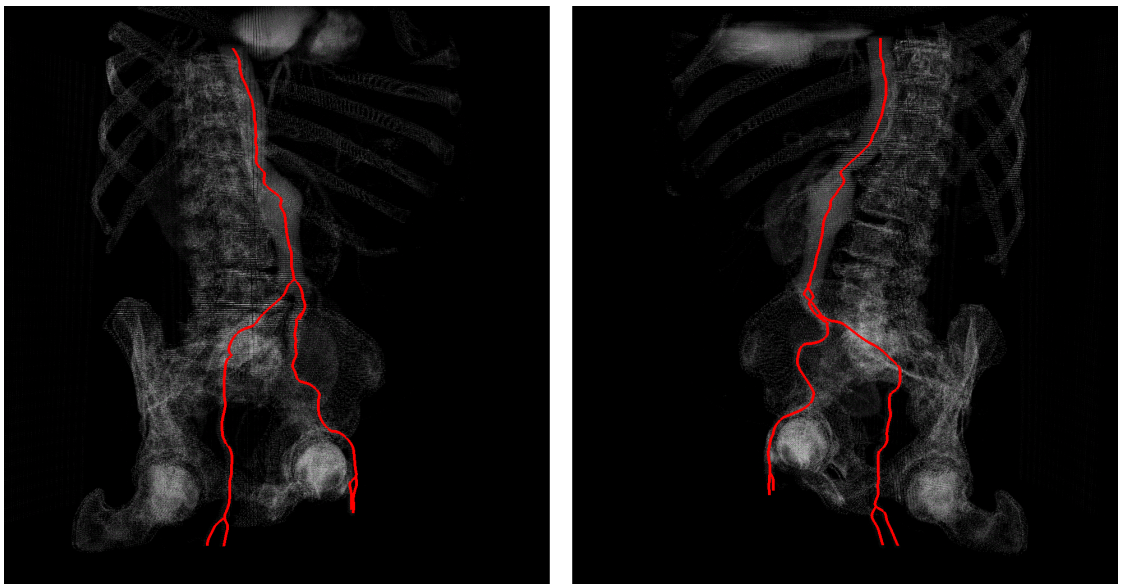


Figure 5.4: *The analytical centerline computed for Patient B, superimposed on the intensity-cropped arterial CT scan.*

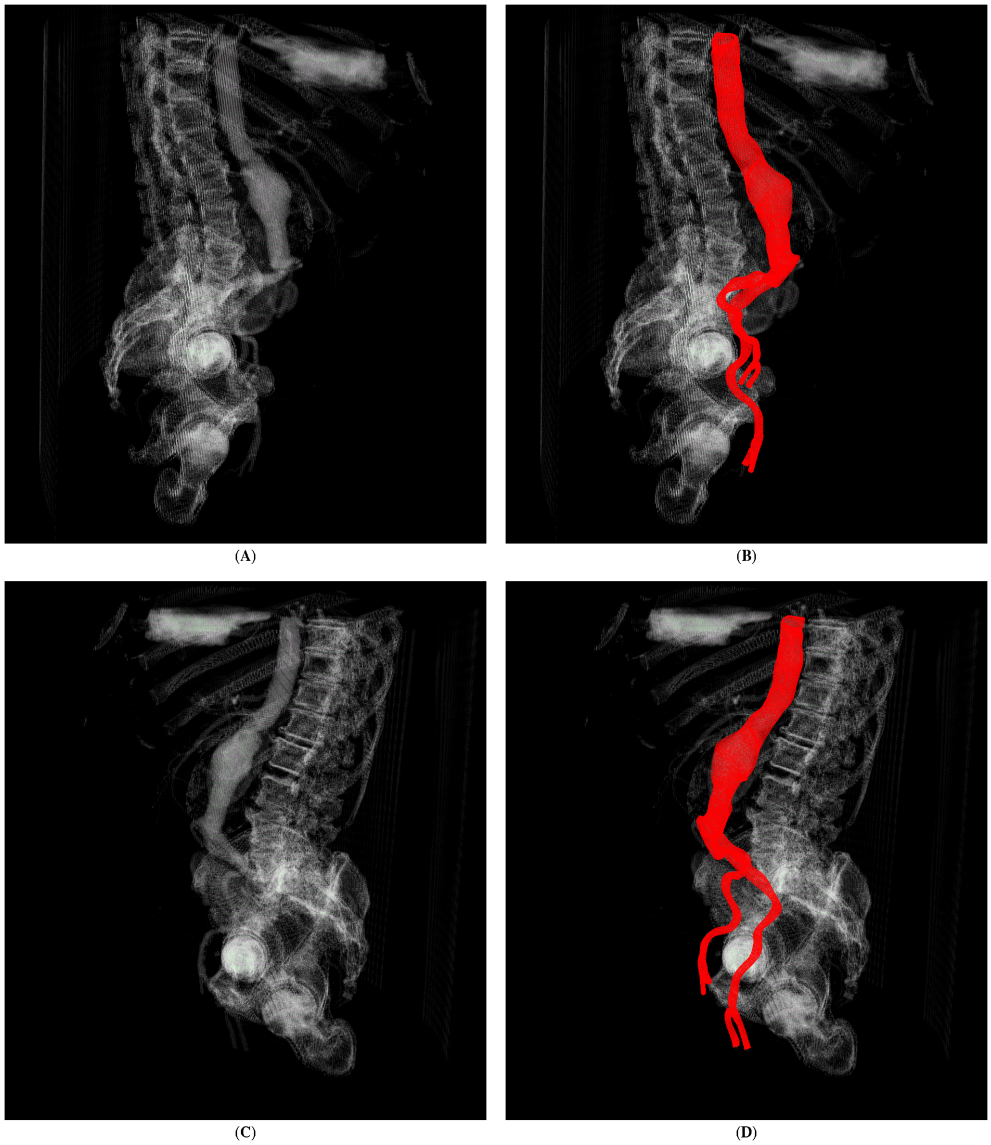


Figure 5.5: *The final analytical surface computed for Patient B. In (A) we see a view of the intensity-cropped arterial scan. In (B) we see the same view with the analytical surface superimposed. In (C) and (D) we see the same from a different viewing angle.*

Figure 5.6: *The analytical surface representing Patient B's arterial lumen, rendered with a Blinn-Phong material to show the 3-D geometry.*

5.1.3 Patient M

We also experienced segmentation difficulties with the data from Patient M. As with Patient B, the intersected volume proved unusable. Again, we performed a manual segmentation of the intensity-cropped arterial scan and used the result in place of the intersected volume.

Despite Patient M's especially tortuous arterial geometry, the centerline extraction procedure worked quite well, as shown in Figure 5.7. The surface fitting procedure also yielded good results, due in large part to the slice placement refinement step described in Chapter 4. The results are shown in Figures 5.8 and 5.9.

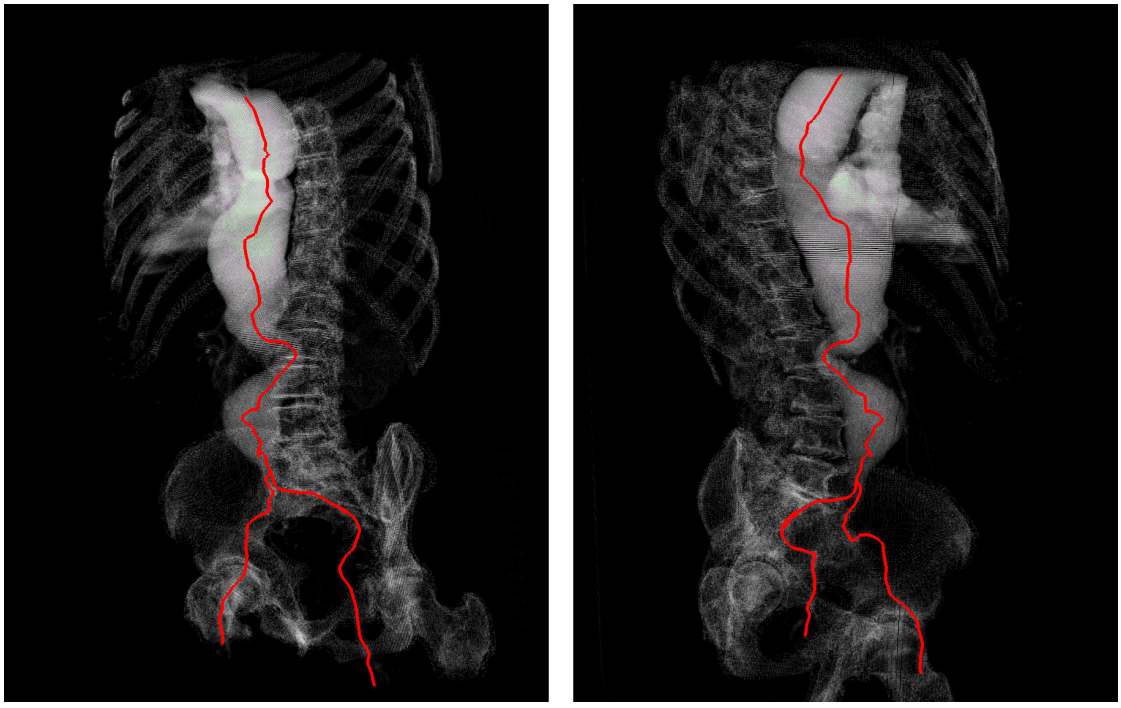


Figure 5.7: *The analytical centerline computed for Patient M, superimposed on the intensity-cropped arterial CT scan. Note the especially tortuous path the centerline must take.*

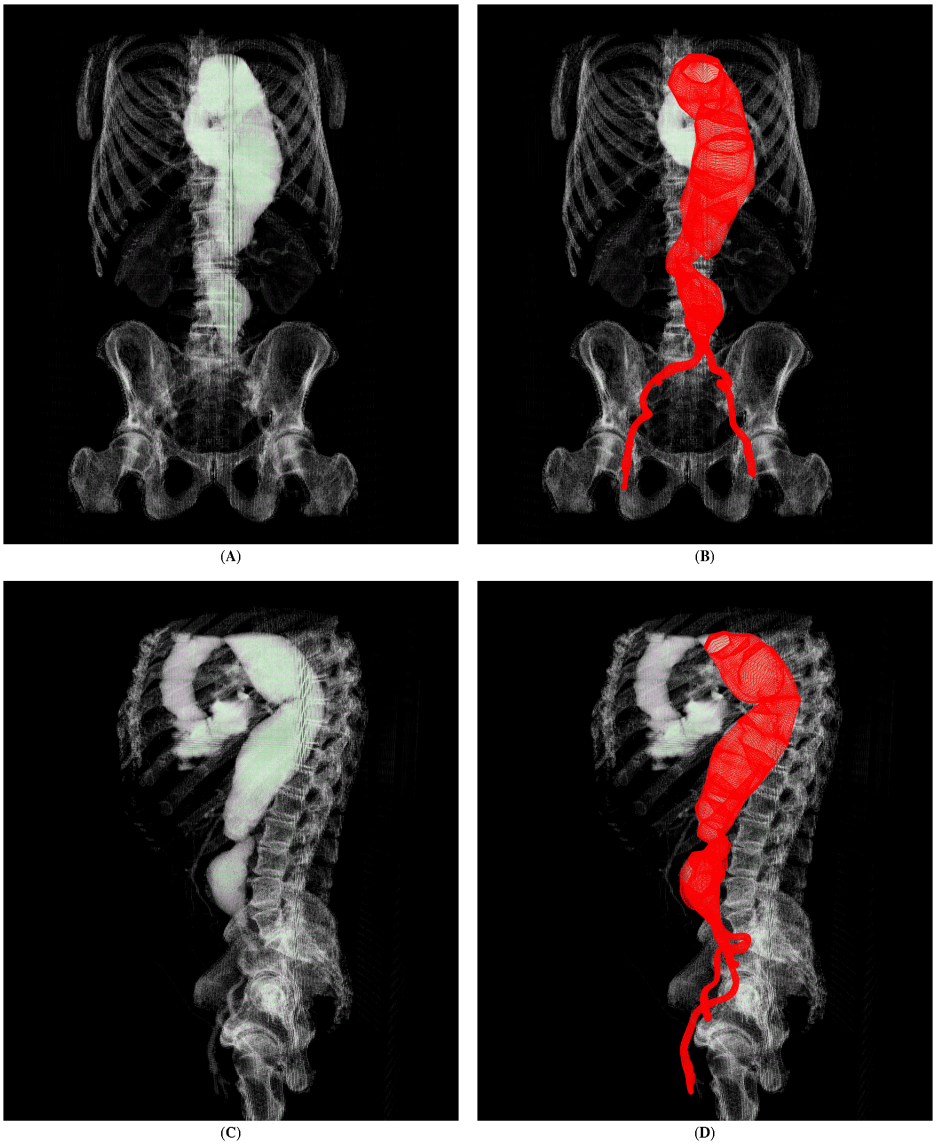


Figure 5.8: *The final analytical surface computed for Patient M. In (A) we see a view of the intensity-cropped arterial scan. In (B) we see the same view with the analytical surface superimposed. In (C) and (D) we see the same from a different viewing angle.*

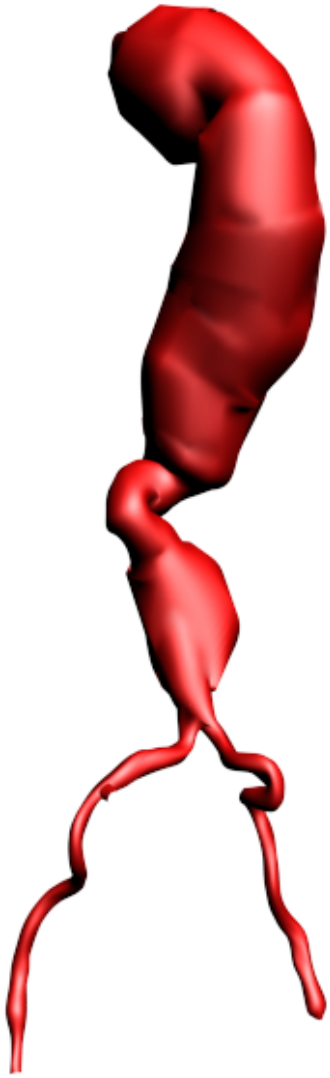


Figure 5.9: *The analytical surface representing Patient M's arterial lumen, rendered with a Blinn-Phong material to show the 3-D geometry.*

Chapter 6

Conclusion and future work

6.1 Discussion

In this thesis we have presented a well-structured efficient procedure for the construction of a geometric representation of the arterial anatomy of a human patient. We begin with CT scan data which is already commonly collected for patients with aortic aneurysms, and we produce a fully analytical model of the arterial lumen's centerline and surface.

We have presented an efficient, robust, and highly automated centerline extraction procedure. It leverages two imaging modalities in order to simultaneously reduce error and reduce the amount of user intervention necessary. In fact, the user input is limited to specifying four scalar values and one two-dimensional vector, all of which require little thought or skill to determine. The result is a procedure which neatly balances the capabilities of a human with the capabilities of a computer, and generates an accurate centerline well-suited for analytical fitting.

We then defined a procedure for performing this analytical centerline fitting. Our fully automated fitting process generates an accurate, smooth, concise analytical description of the centerline in B-spline form. This analytical centerline is useful on its own for purposes such as visualization and measurement, but primarily serves as the basis for our analytical surface fitting procedure.

This surface fitting procedure makes use of the analytical centerline description to produce a lofted spline surface description which is appropriate for the elongated topology of the arterial lumen. It produces a fully analytical representation of the lumen's surface. This can either be used in its analytical form, or can be

rendered or otherwise discretized at any desired resolution. These qualities make this analytical description useful as input for future CAD systems for the design of arterial stents, and for future simulation systems which perform stress analysis and computational fluid dynamics to predict the behavior of the stent and vascular tissue.

6.2 Future work

There are several clear avenues along which our work can be improved. We briefly discuss four potential improvements.

6.2.1 Registration and radiologist cooperation

In testing our procedure on different datasets, we found that the most significant difficulty appeared at the beginning of the process, during segmentation. In particular, problems often appear during the resampling and registration steps in which the native and arterial datasets are matched. This weak point completely failed on two of our three sets of input data.

Because the native scan is performed at a lower resolution than the arterial scan, the native scan must be resampled to match the arterial scan. This upsampling step by necessity produces more information than it is given as input. Though interpolation is done in an intelligent manner, there is no way to guarantee the accuracy of the interpolation, and thus the interpolated values cannot be said to precisely represent reality.

Naturally, the error inherent in resampling causes difficulty in registration. There is an additional source of registration difficulty, however; the bodily movement of the patient between the two scans can be so extreme as to completely

prevent any reasonable registration.

The data we worked with was collected prior to the development of our geometric analysis procedures. In the future, it would be advantageous to involve the radiologist responsible for collecting the data. For example, if the native scan could be performed in the same high-resolution "1mm" mode used for the arterial scan, the resampling step could be eliminated. Furthermore, if the arterial scan could be performed immediately after the native scan, with the patient instructed to limit bodily motion as much as practical, the voxel registration step would be much more likely to provide good results. X-ray-opaque inks or markers could also be used to place registration marks on the patient's anatomy.

We envision that each of these improvements in scanning will become standard procedure in the future. Obviously by improving the foundation of the entire geometric analysis, one could improve the results of every subsequent processing step.

6.2.2 Hole robustness

Our analysis methods are generally resistant to noise in the input data set. By employing two imaging modalities and utilizing their intersection, we reduce the effect of noise in a single scan. Also, by analyzing sets of points rather than individual points, we allow for noisy geometry to be smoothed out.

However, the effect of sampling noise cannot always be eliminated. There is one type of artifact to which our analysis is particularly sensitive. If the flood-filled volume contains any missing voxels – "holes" – the centerline loses its one-dimensional structure around the hole. The distance transformation peak must avoid the border around the hole, and the only way in which this can be done

uniformly is to form a "balloon" surrounding the hole. This causes the centerline to take on the form of a two-dimensional surface around the hole (Figure 6.1).

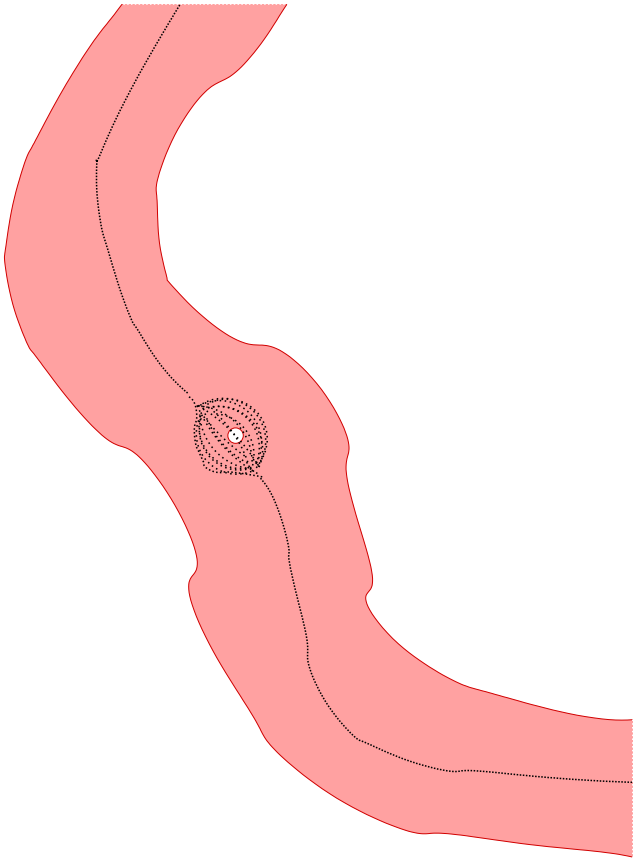


Figure 6.1: *The ballooning effect produced due to a hole in the segmented lumen volume. The centerline must avoid the hole and so cannot maintain its simple one-dimensional structure. Instead, the centerline travels around the hole in all directions, forming a two-dimensional surface.*

If the ballooning artifact is small, it does not materially affect the analytical centerline; during the marching step, the wavefront tends to remain continuous and the centroid stays in the middle of the lumen. If it is large, however, this cannot be relied upon.

While improving the segmentation can help reduce the number of holes, it would still be worthwhile pursuing some modifications to the centerline extraction procedure to make it more robust when holes are present.

6.2.3 Slice placement

The selection of cross-sectional slice locations along the centerline materially affects the geometry of the resulting lofted surface. In our work we have chosen an extremely simplistic approach in which we place slices at regular intervals along the parameter of the centerline, and then adjust these positions if the cross-sectional slices are at risk of intersecting. The results are acceptable but not ideal. It would be worthwhile to explore a more sophisticated slice placement method based on the curvature of the centerline and the shape of the lumen's cross section.

6.2.4 Bifurcations

Properly modeling the branching of arteries is essential to the goal of analytically representing the arterial geometry of a patient. Currently we model each branch separately as a lofted surface and simply consider their boolean union to represent the arterial geometry. This is a good beginning but it must be improved upon.

A more sophisticated bifurcation model would take into account the observation that the juncture between two arteries exhibits continuity to at least the first derivative along a portion of the interface, while exhibiting creasing behavior along the remainder of the interface, as shown in Figure 6.2.

The need to describe the shape of the lumen precisely must be balanced with the need to do so concisely and in a manner resistant to noise. Perhaps artificial slices could be generated at the beginning of each branch to enforce the correctly-

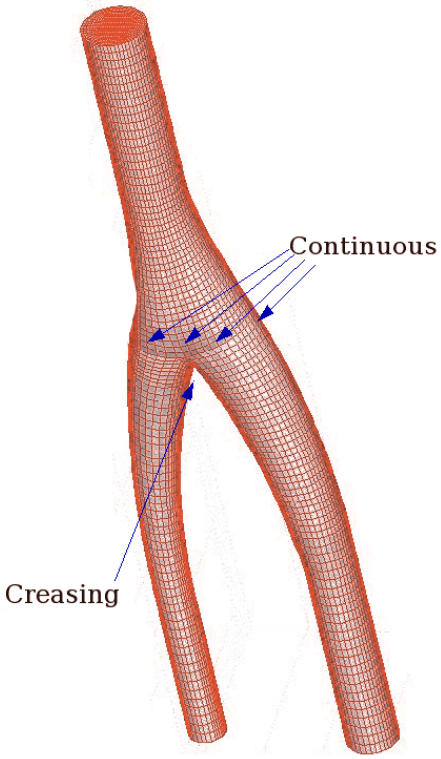


Figure 6.2: *The the junction of two blood vessels, a portion of the interface exhibits continuity of at least the first derivative, while the remainder exhibits creasing behavior.*

shaped joint. Such an approach could yield surfaces which share a portion of their geometry, allowing each branch to be stored separately while still modeling the connection.

6.3 Summary

As discussed in Chapter 1, endovascular aortic aneurysm repair is rapidly becoming a strong and attractive alternative to traditional open surgery. In order to fully realize the benefits of this type of surgery, several aspects of the procedures must be improved. Surgery planning must become a less intensive process for the surgeon.

Methods for accurately representing complex anatomy must be robust. Lastly, using this information, accurate predictions of the future behavior of the stent and arterial tissue must be made.

We have presented a highly automated geometric analysis procedure which begins to address these needs. We begin with CT scan data collected from aortic aneurysm patients and produce a fully analytical model of the patient's arterial geometry. This analytical model is a compact representation of complex geometry, and can be mathematically analyzed or discretized for rendering or finite element analysis with arbitrarily high precision.

Appendix A

A brief review of B-splines

B-splines are commonly used both in numerical analysis and in geometric modeling, and are therefore well-described by the literature. Here we present a very brief review of the mathematical formulation of the B-spline. For more information, the reader is referred to a text such as [BBB87].

A B-spline is a univariate parametric curve. The curve itself can exist in any number of dimensions, but it is a function of a single parameter, typically designated u. The curve's shape is determined primarily by a control polygon comprising a series of control points. The curve's position at any value of the parameter is a weighted average of these control points. See Figure A.1 for an example.

The curve is defined as:

$$P(u) = \sum_{i=0}^{n} V_i N_{i,k}(u) \tag{A.1}$$

V_0 through V_n are the control points. k is the order of the curve (one greater than the degree). $N_{i,k}$ is the blending function, defined as:

$$N_{i,1}(u) = \begin{cases} 1 \text{ if } t_i \leq u < 1_{i+1} \\ 0 \text{ otherwise} \end{cases} \tag{A.2}$$

$$N_{i,k}(u) = \frac{(u-t_i)N_{i,k-1}(u)}{t_{i+k-1}-t_i} + \frac{(t_{i+k}-u)N_{i+1,k-1}(u)}{t_{i+k}-t_{i+1}}$$

t_0 through t_{k+n} are the elements of the knot vector, a nondecreasing series of real numbers. The knot vector matches the series of control points to the parameter, and also defines the range of the parameter u. If the knots are evenly spaced, the curve is called a uniform B-spline, and can be much more efficient to compute due to all blending functions being translations of each other.

The structure of these blending functions is such that they are nonzero only over a span of k knots, that they always sum to unity, and that they are polynomials of degree $k-1$ and are therefore C^{k-2} continuous.

This mathematical formulation yields several desirable properties. The curve remains within a convex hull defined by the control polygon and the order (Figure A.2). The curve is smooth and exhibits C^{k-2} continuity unless knot or control point multiplicities collapse the convex hull. Since each control point has local control, affecting only the nearby portion of the curve (Figure A.3), this formulation is frequently used for design purposes.

In order to represent a closed curve, a uniform B-spline can be made periodic simply by treating the control point indices as a periodic series. And in order to more intuitively represent an open curve, a B-spline's knot vector can include multiplicities at the beginning and end, forcing the curve to pass through the first and last control points.

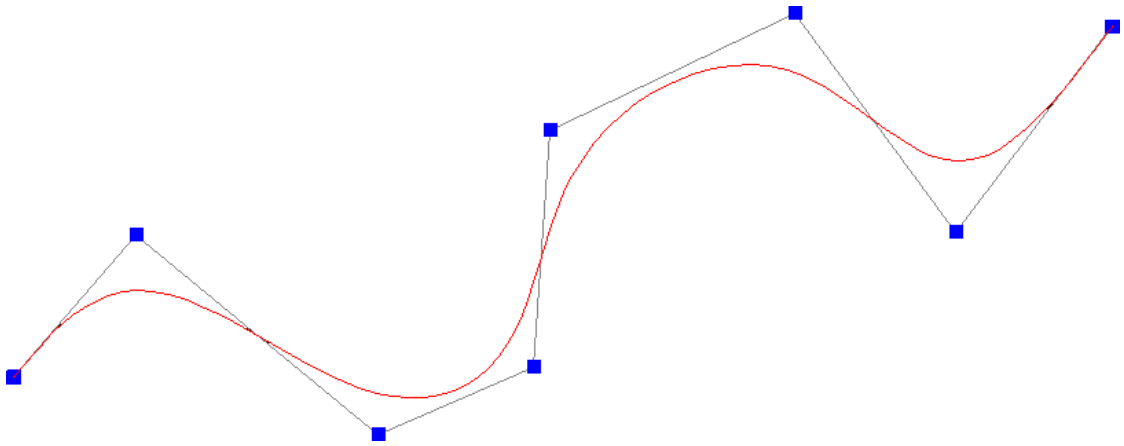


Figure A.1: *An example of a cubic nonperiodic B-spline. The control points are indicated by blue squares. The control polygon is drawn in gray, and the spline curve is drawn in red.*

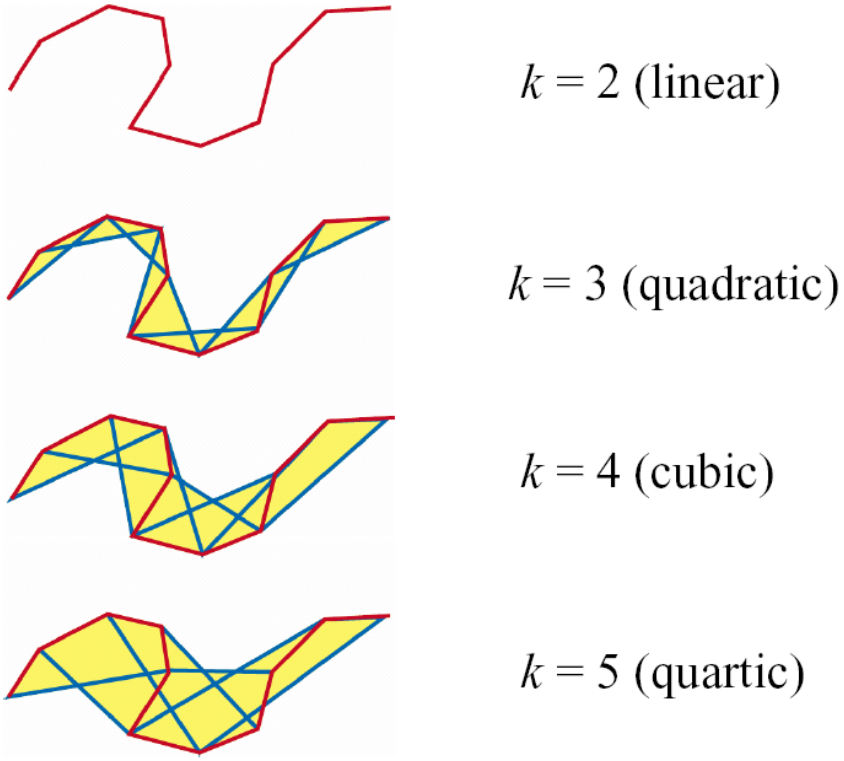


Figure A.2: *A B-spline remains within a convex hull defined by the control polygon and the order of the curve. The convex hull is shown in yellow for four different orders of B-spline. Notice that the region of local control grows as order increases.*

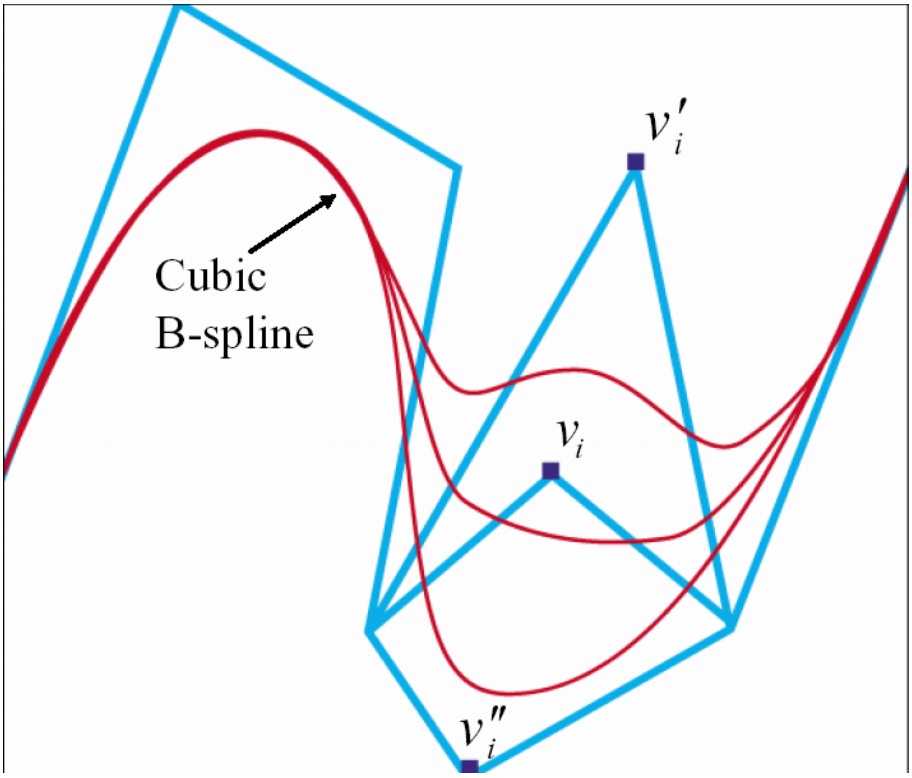


Figure A.3: *A B-spline provides local control; moving a single control point only affects nearby portions of the curve.*

Appendix B

B-spline fitting

As shown in the previous appendix, a B-spline is not a reliable interpolant of its control points. In general, the curve does not pass through the control points unless required to by the structure of the knot vector. B-splines do have advantages over traditional interpolants, however, so it is worthwhile to implement a technique which permits the use of B-splines to interpolate between collected data points.

This technique would begin with a series of geometric input points, and compute the locations of control points which guarantee that the B-spline will pass through the given input points. This task is, in a sense, the inverse of the task of rendering a spline, and is therefore often called the spline inversion problem.

It should be noted that a simple set of geometric points does not fully describe a B-spline. Therefore, a spline inversion procedure must make further assumptions in order to produce a stable result. A fair assumption to make is that each input point is a geometric knot, and therefore falls at the beginning of a span of the curve.

The spline inversion procedure used in our work begins with the observation that the position of each geometric knot depends only on the positions of $k-1$ control points. If one assumes that each geometric knot occurs at the beginning of a span, the weights of the control points are fixed.

For example, for a closed cubic uniform periodic B-spline, the position of the curve at the beginning of span i is equal to:

$$P_i = \frac{1}{6}V_{i-1} + \frac{2}{3}V_i + \frac{1}{6}V_(i+1) \tag{B.1}$$

From this observation, one can construct a linear system representing the spline inversion problem using the geometric knots P:

$$\begin{bmatrix} \frac{2}{3} & \frac{1}{6} & & & & & \frac{1}{6} \\ \frac{1}{6} & \frac{2}{3} & \frac{1}{6} & & & & \\ & \ddots & \ddots & \ddots & & & \\ & & & \frac{1}{6} & \frac{2}{3} & \frac{1}{6} & \\ \frac{1}{6} & & & & \frac{1}{6} & \frac{2}{3} & \end{bmatrix} \begin{bmatrix} V_0 \\ V_1 \\ \vdots \\ \vdots \\ V_m \end{bmatrix} = \begin{bmatrix} P_0 \\ P_1 \\ \vdots \\ \vdots \\ V_m \end{bmatrix} \tag{B.2}$$

This nearly tridiagonal system is easily solved for the positions of the control points V. With slight modifications, this approach can be used to invert other types of splines. For more information, the reader is referred to [WAG77].

Appendix C

Lofted B-spline surfaces

Splines are one-dimensional curves. An analytical surface is function of two parameters. There are numerous methods of representing surfaces using splines. In this work, we use a technique called lofting. A 3-D lofted surface is defined by a series of control curves. The lofting process interpolates between these curves to produce an entire surface.

This is a simple but powerful method of describing a surface analytically. The first parameter dictates the position along the control curves, and the second parameter dictates the position between control curves. For an elongated structure such as those dealt with in our work, such a parameterization is elegant, as it allows one to independently address the position around a cross-sectional slice and the position along the centerline.

The method of interpolation can vary according to the application. In our work, it made good logical sense to use the B-spline inversion technique outlined in the previous appendix. Each cross-sectional slice is used as a control curve, and the values of all slices at a particular value of the parameter u are used as control points for a nonperiodic axial spline indexed by the parameter v.

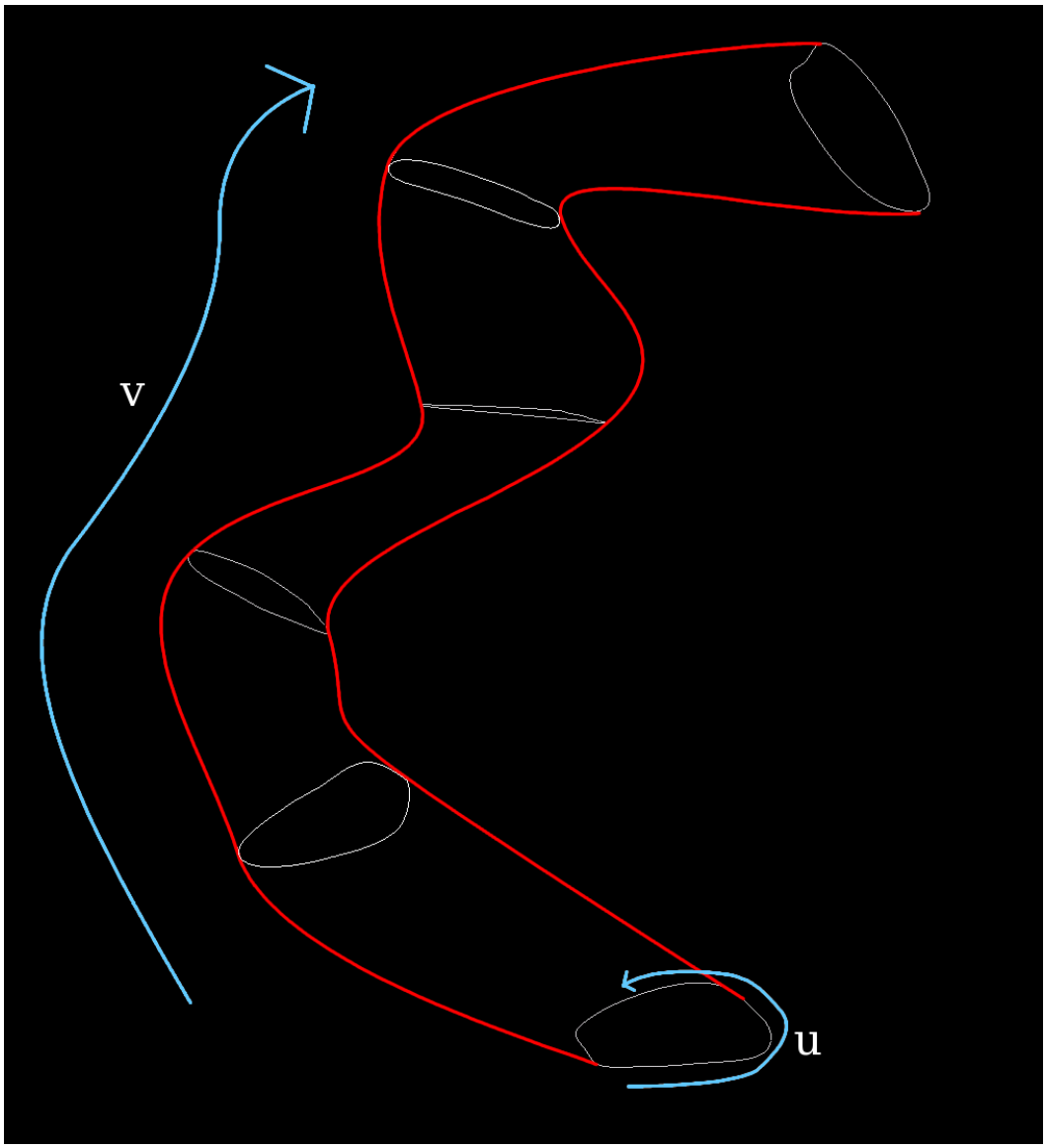


Figure C.1: *An example of a lofted surface. The border of the surface drawn in red interpolates the six white control curves. Though in this example they are, in general the control curves do not need to be planar. The lofted surface is parameterized in two dimensions by u and v.*

BIBLIOGRAPHY

[AEIR03] Luca Antiga, Bogdan Ene-Iordache, and Andrea Remuzzi. Computational geometry for patient-specific reconstruction and meshing of blood vessels from mr and ct angiography. *IEEE Transactions on Medical Imaging*, 5:674–684, May 2003.

[BBB87] Richard H. Bartels, John C. Beatty, and Brian A. Barsky. *An introduction to splines for use in computer graphics & geometric modeling*. Morgan Kaufmann Publishers Inc., 1987.

[BJMR01] Frederic Banégas, Marc Jaeger, Dominique Michelucci, and M. Roelens. The ellipsoidal skeleton in medical applications. In *Proceedings of the sixth ACM symposium on Solid modeling and applications*, pages 30–38. ACM Press, 2001.

[BKS01] Ingmar Bitter, Arie E. Kaufman, and Mie Sato. Penalized-distance volumetric skeleton algorithm. *IEEE Transactions on Visualization and Computer Graphics*, 7(3):195–206, 2001.

[Bru01] Jan Bruijns. Fully-automatic branch labelling of voxel vessel structures. In *VMV*, pages 341–350, 2001.

[BSB+00] Ingmar Bitter, Mie Sato, Michael Bender, Kevin T. McDonnell, Arie Kaufman, and Ming Wan. Ceasar: a smooth, accurate and robust centerline extraction algorithm. In *Proceedings of the conference on Visualization '00*, pages 45–52. IEEE Computer Society Press, 2000.

[Dij59] Edsger. W. Dijkstra. A note on two problems in connexion with graphs. *Numerische Mathematik*, 1:269–271, 1959.

[DZ02] Tamal K. Dey and Wulue Zhao. Approximate medial axis as a voronoi subcomplex. In *Proceedings of the seventh ACM symposium on Solid modeling and applications*, pages 356–366. ACM Press, 2002.

[FB89] Daniel J. Filip and Thomas W. Ball. Procedurally representing lofted surfaces. *IEEE Comput. Graph. Appl.*, 9(6):27–33, 1989.

[GS99] Nikhil Gagvani and Deborah Silver. Parameter-controlled volume thinning. *CVGIP: Graph. Models Image Process.*, 61(3):149–164, 1999.

[GSV96] Yaorong Ge, David R. Stelts, and David J. Vining. 3d skeleton for virtual colonoscopy. In *VBC '96: Proceedings of the 4th International Conference on Visualization in Biomedical Computing*, pages 449–454. Springer-Verlag, 1996.

[LC87] William E. Lorensen and Harvey E. Cline. Marching cubes: A high resolution 3d surface construction algorithm. In *Proceedings of the 14th annual conference on Computer graphics and interactive techniques*, pages 163–169. ACM Press, 1987.

[LLT+01] Patrick Lermusiaux, Cécile Leroux, Jean Claude Tasse, Lucien Castellani, and Robert Martinez. Aortic aneurysm: Construction of a life-size model by rapid prototyping. *Annals of Vascular Surgery*, 15(2):131–135, 2001.

[MAKSZ02] Mahnaz Maddah, Ali Afzali-Kusha, and Hamid Soltanian-Zadeh. Fast center-line extraction for quantification of vessels in confocal microscopy images. In *Proceedings of the 2002 IEEE International Symposium on Biomedical Imaging*, pages 461–464. IEEE Computer Society Press, 2002.

[MS96] C. Min Ma and Milan Sonka. A fully parallel 3d thinning algorithm and its applications. *Computer Vision and Image Understanding*, 64(3):320–433, 1996.

[SFD+99] Y. Samara, M. Fiebich, A.H. Dachman, J.K. Kuniyoshi, K. Doi, and K.R. Hoffmann. Automated calculation of the centerline of the human colon on ct images. *Academic Radiology*, 6:352–359, 1999.

[SPB95] Evan C. Sherbrooke, Nicholas M. Patrikalakis, and Erik Brisson. Computation of the medial axis transform of 3-d polyhedra. In *SMA '95: Proceedings of the third ACM symposium on Solid modeling and applications*, pages 187–200. ACM Press, 1995.

[Tv02] Alexandru Telea and Jarke J. van Wijk. An augmented fast marching method for computing skeletons and centerlines. In *Proceedings of the symposium on Data Visualisation 2002*, pages 251–ff. Eurographics Association, 2002.

[TV03] Alexandru Telea and Anna Vilanova. A robust level-set algorithm for centerline extraction. In *Proceedings of the symposium on Data visualisation 2003*, pages 185–194. Eurographics Association, 2003.

[WAG77] Sheng-Chuan Wu, John F. Abel, and Donald P. Greenberg. An interactive computer graphics approach to surface representation. *Communications of the ACM*, 20(10):703–712, 1977.

[ZT99] Yong Zhou and Arthur W. Toga. Efficient skeletonization of volumetric objects. *IEEE Transactions on Visualization and Computer Graphics*, 5(3):196–209, 1999.